Proactive Detection and Automated Exchange of Network Security Incidents

Piotr Kijewski
NASK / CERT Polska

piotr.kijewski@cert.pl

Paweł Pawliński
NASK / CERT Polska

pawel.pawlinski@cert.pl

ABSTRACT

The paper covers a recently published ENISA report – conducted by CERT Polska - on the "Proactive Detection of Network Security Incidents" and its implications for early warning at a national level. Proactive detection of network security incidents is the process of discovery of malicious activity in a CERT's constituency through internal monitoring tools or external services that publish information about detected incidents, before the affected constituents become aware of the problem. The study that lead to the report was focused primarily at national/government CERTs, but many of the issues discovered concern any team tasked with the detection of incidents. In this paper, we analyze the results of the study in order to identify key features of systems that can automate the early warning process at a national level. In such a scenario, national/government CERTs act as intermediaries that deliver security incident information. We present four tools that have been created to sustain this process of incident exchange and look how a generic analytics tool can serve a similar purpose as well. We compare all these tools based on features identified thanks to the ENISA study and show why the automation of effective early warning of network security incidents is still a long way away.

1 INTRODUCTION

Over the last decade, attempts have been made towards applying the notion of "early warning" to threats targeting computer networks [1-7]. Parallels have been drawn to systems operating in other fields, such as those aimed at warning against incoming earthquakes or weather prediction systems in general. Such analogies seem attractive, but are not easy to transfer to the computer and network security field. There is virtually no possibility of forecasting what will happen to a network security-wise tomorrow. Granted, the publishing of a remote administrator vulnerability in an operating system may allow us to predict that a worm or botnet will appear to exploit it, however, this prediction is not based on any scientific observations of a particular network activity and does not tell us when, nor which, networks will necessarily be affected.

This does not mean that the concept of early warning in the network security detection field is not applicable: expectations however have to be changed. In fact, this is how we understand the concept of proactive detection of network security incidents: the process of discovery of malicious activity in a CERT's constituency through internal monitoring tools or external services that publish information about detected incidents, before the affected constituents become aware of the problem. In this sense proactive detection can be viewed as a very valuable form of an early warning service from the constituents' perspective. Effective proactive detection of network security incidents is one of the cornerstones of an efficient CERT service portfolio capability. It can greatly enhance CERT's operations, improve its situational awareness and enable it to handle incidents more efficiently, thus strengthening the CERT's incident handling capability.

We look at the process of proactive detection of network security incidents from the perspective of a national/government CERT. In practice what this means is that the national/government CERT acts as an intermediary between data providers and data consumers at a country level. In order to better understand the perspective and problems faced by such an intermediary when handling bulk incident data and its implications for effective early warning, we turn to the findings of the "Proactive Detection of Network Security Incidents" study conducted by ENISA in 2011 [8].

2 REPORT ANALYSIS

The ENISA "Proactive Detection of Network Security Incidents" study was launched by ENISA to gain a better understanding of how CERTs proactively detect incidents and to look at ways on how the process could be improved. A noteworthy aspect of the study was that it was largely community driven – based on a survey of 45 different CERTs and on input from a security expert group specifically formed for the study, supplemented by the research and knowledge of members of the CERT Polska team and ENISA. The study listed 30 external sources that can be used for proactive detection and 12 categories of tools that can be used to collect incident information. 16 shortcomings in the proactive detection process were identified and 35 recommendations for improvements given.

One of the main discoveries of the study is that CERTs are underutilizing the various detection capabilities at their disposal. Even if they do detect incidents, they fail to share data they gather. For instance, the best rated and most popular external source of information (Shadowserver Foundation [9]) is used by only 40% of CERTs. Furthermore, just 35% share information that they collect on incident data about other constituencies – information that can be used for early warning. Automation and correlation, critical to handling large amounts of incidents, was also discovered to be a weak spot: only 33% of CERTs automatically correlate incidents.

We look into technical aspects that act as inhibitors in the proactive detection process identified in the report, that could thus form the basis of a taxonomy of security incident data sharing systems, usable in the national/government CERT community. We are aware of legal and organizational problems with the proactive detection process, but these are beyond the scope of this article – a discussion of those can be found in the ENISA report. Note that the report distinguishes two key players: data providers and data consumers. Although national/government CERTs can be both, in the context of automated security incident data exchange systems, they are intermediaries whose influence on many aspects of the exchanged data sets, while limited, can still have an impact.

A key question of the survey amongst CERTs in regard to this paper concerned areas of improvement for new sources of information. These responses are illustrated in Figure 1.

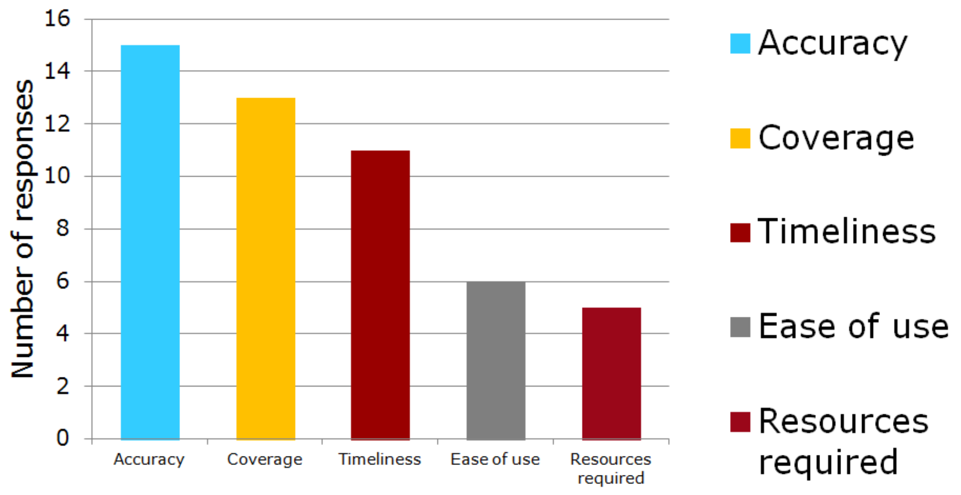

Figure 1: ENISA survey responses from CERTs: What would you like to improve when trying new sources in obtaining information about security incidents in your constituency?

The responses show that if we take accuracy (a feature that relates to false positives in a dataset), coverage (a feature that relates to false negatives in datasets – in other words, attacks or types of attacks that are missed) and timeliness (quickness of delivery of information) as elements of data quality, there is a clear expectation for improvement in this area. Ease of use and resources required, coupled with the identified lack of automation, on the other hand imply a general lack of ability or weaknesses of tools.

In the next sections of this paper, we explore in more detail issues related to data quality and take a look at existing systems for the automated exchange of incident information that can serve as early warning. We develop a set of requirements for data quality that should be addressed by such systems and evaluate each system against them.

3 DATA QUALITY

The ability to act on the received information is an important element of the early warning process. The information sent has to be reliable, otherwise the side consuming the information will lose confidence in the data being transmitted and stop acting on it. Unfortunately, there have been cases when CERTs have stopped automatically forwarding incident information from well-established sources because of the reported false positives, and have resorted to manual vetting of information instead.

As a national/government CERT is often not the original source of the information, acting instead as an intermediary, there are limitations to what can be done to improve data quality. However, a number of actions (often found lacking in both the processes of the data provider and consumer) that have impact can be identified. These can be grouped into two categories: a) those actions that can result in improvement of data quality by the intermediary b) those actions that allow for a preservation of data quality i.e. do not lead to improvement (which is entirely source dependent) but can inadvertently lead to a degradation of data quality if not carried out by the intermediary. These actions should be implemented in any automated system for the exchange of security incident information.

3.1 Improvement of data quality by an intermediary

Actions leading to the improvement of data quality by an intermediary CERT include:

- **Correlation**. Correlation can be used to see if multiple sources corroborate on a particular incident, and hence used to reduce false positives.
- **Filtering**. Filtering can be performed to remove data that has proven to be unreliable in the past.
- Assignment of **confidence levels**. Different data sets, depending on the way they were collected, offer different quality. For instance, data obtained from a P2P sinkhole will have much less false positives than data gathered from an experimental shellcode detector. The ability to assign validity indicators to datasets will allow data consumers to make more informed decisions based on them. Note that this trust may be applicable not just to a particular data source but also incident type.
- **Data enrichment**. Enrichment of data collected from data providers is another step that allows not just the CERT but also consumers to make better informed decisions. This enrichment may be as simple as adding IP or WHOIS information to URLs provided.
- Providing a **feedback mechanism**. Consumers of data should have the capability of providing feedback on the quality of a data source. This information should also be forwarded to the data source. Such a feedback mechanism is necessary to be able to develop metrics of the effectiveness of the remediation processes. Ease of use is important so as to encourage consumer involvement, which otherwise may not be that forthcoming.
- Implementing **long term storage**. Storage of data for future reference creates the possibility for verification of incident information and observations on long term trends.
- Providing **situational awareness**. Situational awareness includes the ability to provide statistical supporting information when reporting a threat. For instance, this may be information whether a threat that is being reported is seen as a TOP 10 global threat. It can also include a capability to perform a more in depth investigation of a particular incidents, for example, through correlation.
- **Trend analysis**. By performing trend analysis of collected datasets, CERTs can gain a better understanding of the changing threat environment.
- Providing **anonymization**. The capability of anonymizing a data source is important, as in many cases data sources do not wish to be identifiable by the consumer [4].

3.2 Preservation of data quality by an intermediary

Actions that should be carried out to preserve the data quality of a source when processed by an intermediary include:

- Specifying the **data collection method**. This allows the consumer to better understand information being forwarded to them and its implications. An intermediary should ensure that if such a specification is present in a forwarded dataset it should not be removed.
- **Timely delivery**. Timeliness influences the perception of the quality of a dataset. Preferred methods of delivery involve real time but batch modes of delivery are the most common. A slow time of delivery of information to the consumer may render it unusable. An intermediary should therefore not introduce additional delay.
- Publishing the **data aging policy**. In some cases, to make data more actionable, publishing of blacklists is required. An example may be DNS blackholing. If a data aging policy is published by an original data source, it should be preserved along with any information that allows an end user to understand when entries were added. Similarly, in cases where an exchange system itself publishes such lists, it must be done in such a way as to contain clear information about when entries were added and a policy for their removal.

4 SURVEY OF TOOLS FOR AUTOMATED INCIDENT PROCESSING

Automation is necessary if a CERT is to effectively communicate early warning information about problems on a constituency network. This is not just because of the timeliness aspect, but also because of the scale of the problem when dealing with a large network, e.g. at a country level. For instance our team had to handle over 21 million security incidents in 2011, something not possible without automation [10].

The study found that CERTs often used simplistic in-house solutions customized to their needs. Commercial solutions, it was observed, are hardly used, as they tend to focus on an enterprise and network intrusion detection, so they do not adapt well to a national/government CERT environment.

Furthermore, maintenance of all these solutions is a problem due to the low amount of resources that can be devoted to such a task – even to set them up properly. It was also noted that there are very few tools if any that are widely adopted in the CERT community and have a good level of community support.

Differences between CERTs and their varying missions force them to customize tools to adapt to their specific needs, leading to additional drainage of resources. Incompatibility between various tools was also seen as a problem in this regard.

Data exchange is invariably linked to the age-old problem of lack of common formats. Despite numerous past efforts over the last few decades, with initiatives such as IODEF [11] or X-ARF [12], no standard has gained wide acceptance in the security data exchange community. The result is that there are many formats, especially text-based, for different automated data feeds. The ENISA study observes that although this may be seen as an issue, because each file format generally requires a different parser to process and subsequently to unify the data with other sources, that is not necessarily the case. In fact, many experts taking part in the study noted that if a given data source is stable (in other words, it does not appear and disappear all the time or change formats often) and valuable (for example, there is much data delivered and it meets high standards), the cost of creating a custom parser for this feed is acceptable when compared to the benefits. This is especially the case when the file format is simple. In fact, based on all the previous failures, it may be reasonable to assume that no one single standard can be agreed upon by everyone, which in turn means that any automated system developed will have to handle numerous formats.

This section describes all major systems for processing incoming security incident reports automatically available for CERTs and systems that we developed specifically for CERT Polska, which are not yet shared with other organizations. Due to unique requirements for batch processing incident data outlined in the section, most of the systems listed here were created from scratch, since existing tools (e.g. SIEM[1]s) are not designed for such use.

We reviewed 5 tools – Megatron, Collective Intelligence Framework (CIF) [13], AbuseHelper [14], n6 [15] and a prototype system based on existing analytics software. Except two last solutions, which were created entirely by our team, this survey is primarily based on analysis of available documentation and source code review. Additionally, we performed limited testing of CIF and took part in an AbuseHelper workshop in February 2012, which covered installation, configuration and component development.

Note that we purposefully left the term incident very generic, as there is no precise widely accepted definition of the term incident. A similar approach was taken in the IODEF RFC document [11]. CERTs should review what kind of incidents they wish to share automatically with their constituency and use this as another dimension to the evaluation process of the systems introduced below.

[1] *Security Information and Event Management systems*

4.1 Megatron

Megatron is an automation system for CERTs that aims to systematize the fight against Internet abuse. It was created by CERT-SE (the national CERT of Sweden) and entered production use in 2009.

Megatron is a centralized system oriented for batch processing of feeds from external sources and distributing the incident information to the affected organizations. Input data consists of files with incident reports, which can be downloaded from remote servers using a built-in script or fetched from other sources (e.g. email) using custom tools. Megatron has a concept of atomic **jobs**, that contain all steps required to process a single input file, once it has been downloaded. System's configuration defines all supported job types and contains directives that control how these files are parsed. Parsing involves splitting the input to individual log entries (events) and extracting a fixed set of metadata for each entry: time stamp, network addresses, ports, country codes, autonomous systems (ASNs), hostnames and URLs. Content of each log entry is preserved in the original form for reference.

Collected data is inserted to a MySQL database that uses a simple relational schema where each extracted metadata field has a corresponding column (Figure 2). Data not fitting the existing attributes can be put in a list of key-value pairs associated with a particular event, however Megatron has limited capabilities for dealing with this kind of information. Such approach makes the database easy to query and manipulate using any MySQL client, as long as the incident data fits the predefined data model.

Having a fixed set of metadata implies that all entries in the database are already normalized by the parser, so semantics of the attributes are not ambiguous. Once the data is inserted in the database, the system enriches events' metadata if they do not contain a complete set of attributes. ASN, country code and reverse domain name are determined from an IP address through lookups in local databases. It is also possible to resolve a domain name in order to get IP addresses associated with it.

Megatron allows operators to define list of organizations along with their constituencies, expressed through domain names, ASNs and IP ranges. Every new incident in the system is matched against these constituencies and can be associated with up to two different organizations. The system automatically distributes information about incidents to affected organizations by email. Operators can assign priority to different sources of data and set the notification threshold individually for each organization.

Megatron is controlled through configuration files and a simple command line utility, there is no graphical interface of any kind. It does not have a public application programming interface (API) for integration with external tools.

The software is available with an open source license (Apache 2.0) on e-mail request to <cert@cert.se> - access to the official web page is restricted. While there is no visible community around the project, the software is used by 3-4 CERTs and still maintained by CERT-SE with new features being added.

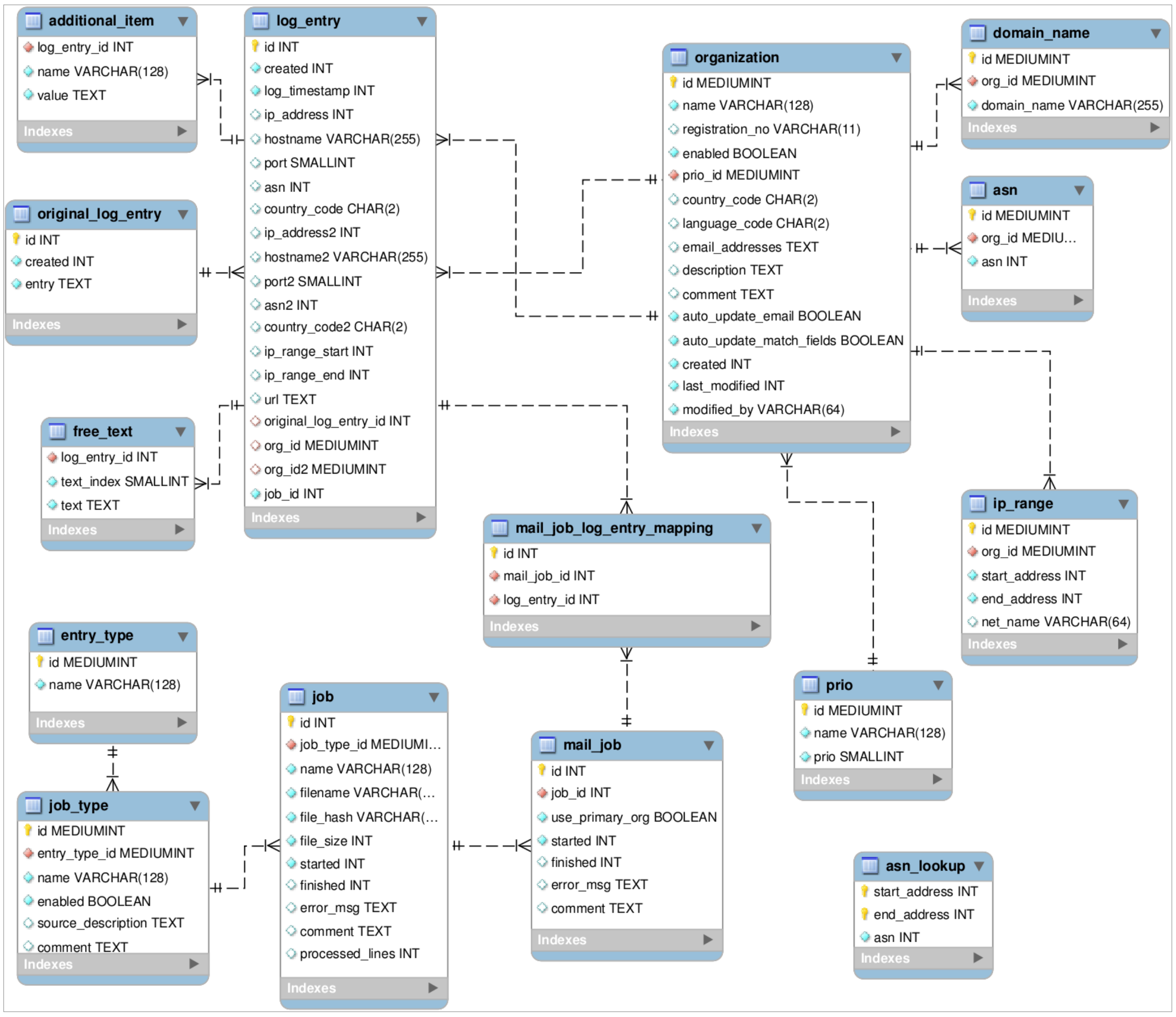

Figure 2: SQL schema of Megatron database, version 1.0.9.

4.2 Collective Intelligence Framework

CIF is a framework for warehousing security intelligence information in a single repository created by REN-ISAC (Research and Education Networking Information Sharing and Analysis Center). The main goal of the project is to collect security-related data from multiple sources and provide mechanisms to effectively query, correlate and share it. CIF evolved from the Security Event System - a project with similar goals, also developed by REN-ISAC - and is currently funded through a NSF (National Science Foundation) grant. Currently SES is considered an implementation of CIF within REN-ISAC [18]. At the time of writing 0.2 was the most recent release of the software.

The data model used by CIF is based on Incident Object Data Exchange Format [11] - a standard for sharing information among CERTs, approved by the Internet Engineering Task Force (IETF) in 2007. IODEF is an XML-based data format which can be used to describe various types of information related to security incidents in a structured form. It was designed to enable introducing greater automation in processing of incident data, decrease effort required to normalize data from different sources and provide a common platform that can be used by multiple interoperable tools. Interoperability of tools using IODEF is enforced through a schema that is part of the RFC. Adoption of IODEF means that every element of information that is a part of an incident report has well defined semantics.

Similarly to Megatron, CIF periodically fetches data from multiple sources and inserts it into a SQL database. The system has built-in mechanisms for downloading files from remote servers and parsing multiple formats, including CSV, XML and JSON. Processing details for each source are defined through simple configuration files, so in most cases there is no need for introducing custom scripts for data download.

All kinds of fetched data is split into individual events, normalized on the fly to a modified IODEF and then events are saved directly in a PostgreSQL database. CIF introduces to important changes to the original IODEF - the XML syntax is replaced with a JSON one and incidents are divided into a different set of categories (**impact** attribute). Rationale behind the syntax change may come from the design of the IODEF - according to the RFC, this format should serve only for transport. Long-term storage and in-memory processing are explicitly listed as use cases that might not be addressed by IODEF. According to CIF authors, the JSON syntax will be replaced in favor of Protocol Buffers (binary serialization format developed by Google [20]) in the next upcoming version of the software [19].

The system does not use a classic relational schema for storing data and stores whole IODEF+JSON messages in a text column of a single table. In order to enable efficient queries, selected metadata is extracted and put in dedicated tables that serve as indexes referencing back to complete incident data. For example, there is a separate table for IP addresses (metadata type) related to botnets (category) with columns corresponding to addresses, protocols and ports. This approach guarantees flexibility of the CIF data model and, at the same time, allows achieving good performance thanks to indexes that can be created for multiple attributes.

CIF periodically runs a set of data enrichment routines (**analytics**) on newly collected events. Majority of analytics implemented in the current version query external databases through DNS to retrieve extra data related particular attributes of incidents. Among other features, the system integrates with the Team Cymru service for checking malware hashes [16], looks up entries in the Spamhaus database [17] and uses the normal DNS infrastructure to extract A and NS records for domains.

The system has a client-server architecture with a single server that entirely responsible for storage and processing. Client applications use a simple REST API to query the database. Both server and client side are implemented in Perl and have a high degree of modularity, so adding analytics or new types of feeds should be relatively simple.

CIF emphasizes some key IODEF concepts and reflects them in its data model and in the query semantics. Every incident is characterized by its impact, which is composed of **severity** and **type**. Severity is self-explanatory, type corresponds to one of predefined categories that represent the overall meaning of the event, e.g. phishing source, exploitation attempt, botnet activity, etc. Additionally, an estimate of the validity of the report is associated with every event in the system through the **confidence** attribute. On top of these concepts, CIF features the granular disclosure policy model from IODEF, where each incident and element of an incident can have a **restriction** attribute defined. Information may be unrestricted (public), distributed on a need-to-know basis, or entirely private.

The system periodically generates feeds of recent reports for every type of threat based on the means that can be used to identify a particular threat, e.g. an IP address, URL, or cryptographic hash. These feeds are further divided into specific incident types, e.g. domain-botnet. Apart from feeds, all historical data in CIF can be searched for particular identifiers (addresses, ASNs, etc.). Queries to the system can specify desired severity, type of incidents (impact), confidence and restriction levels.

Normally all queries are performed through the standard CIF client, which allows transforming results into one of many supported formats, including human-readable tables, CSV, JSON, or Snort IDS rules. The client application must also be used in order to fetch feeds generated automatically by CIF, since the system does not provide any other means of distributing data.

CIF is actively developed by REN-ISAC - at the time of this research (August 2012) the software was undergoing major changes in preparation for the release of 1.0 version and does not offer backwards compatibility yet. The project is available under 3-clause BSD license and already has a user community providing feedback, improving existing features and integrating external tools, including log-analysis platforms like ELSA and Splunk (described in more detail in section 4.5).

4.3 AbuseHelper

Another solution for automated processing of incident reports was developed by CERT-FI, CERT-EE (national CERTs of Finland and Estonia) and Clarified Networks. AbuseHelper is a complete redesign of Autoreporter - a system that CERT-FI used internally for the same purpose. It was released for public in 2010 and has gathered a community of national CERTs and other organizations since then.

Instead of using a database centric approach that characterizes Megatron and CIF, AbuseHelper is based on a framework of distributed agents, named **bots**, processing event streams in real time. Bots communicate over XMPP (Extensible Messaging and Presence Protocol [21]) - an open XML-based messaging protocol, originally designed for instant messaging and related services, standardized by IETF. Using a widely adopted and open protocol greatly improves interoperability, since existing tools can be used to communicate with the system.

AbuseHelper splits processing of security data into basic steps performed by independent bots. While bots can be implemented in any technology, in practice they are all created in Python, using a common communication library which exposes a high-level API for scripts. AbuseHelper can be viewed as a flexible framework for real time event processing - there exist many processing components (bots) that perform various tasks but the data flow between them is entirely configurable and there is a library that simplifies adding new ones. Additionally, bots are controlled entirely through XMPP, meaning that the whole system can be reconfigured at run time.

A typical sequence of steps that a national/government CERT acting as a proxy takes for each incident report it receives could be straightforwardly translated into the following AbuseHelper data flow. First, the **collector** bots fetch data from variety of sources, using adequate transport mechanisms, including HTTP, IMAP, and even real time feeds like IRC. Raw input data is immediately transformed into a stream of discrete **events**, represented as an arbitrary list of properties (key-value list). In order to keep consistency of selected set of properties throughout the system, events are passed to a **sanitizer** bot appropriate for the source, which normalizes the input data. The next processing stage is enrichment, where bots responsible for enrichment (**experts**) augment events with additional information such as ASN, country code for an IP address, records from passive DNS data and GeoIP information for IP addresses. Most of expert bots simply query external services like Team Cymru or Internet Storm Center (ISC). All information added by experts is combined back to a single event by a dedicated bot. After the enrichment has finished, incidents are distributed to channels that represent organizations that should be notified about particular event. This filtering process is performed by a dedicated bot that uses a rule-based configuration in order to route events to appropriate destinations. Finally, a **mailer** bot picks up events from organization-specific channels and sends email notifications about incident to affected parties.

As can be seen from above example, AbuseHelper deployed in a organization that just forwards incident reports does not have to use any database at all. Moreover, currently the official source code repository does not contain any bot that would store events in a persistent database. The standard method of achieving persistence in AbuseHelper is using the **archive** bot, which logs in text files all events that it sees. Naturally, nothing prevents third-parties from creating new bots that integrate AbuseHelper with databases of their choice and to our knowledge, such software have already been developed internally.

The project's source code is hosted in a public repository [14] and is provided under the permissive MIT license, however all documentation is available only on a wiki with restricted access. AbuseHelper is actively developed and used by an increasing number of CERTs. At the time of writing the code repository contained over 40 bots, with various collectors being the largest group.

4.4 Network Security Incident Exchange (n6)

Until early 2011, CERT Polska did not have a single system for automated processing of incident reports received from multiple sources. Since our team processes security incidents for all of Poland, we receive quite large volume of data related to events occurring in our constituency. Additionally, our own monitoring systems - ARAKIS [22] and Honey Spider Network [23] - constantly generate feeds about threats which are often located outside of our constituency. Various ad hoc solutions were developed for processing and distributing some of these feeds, however they did not scale and required too much maintenance. As a result, we were unable to function effectively even in the role of a proxy and some reports that we received got lost.

In such a situation it became obvious that we had to deploy a single coherent system to manage all the security-related data that we have. After a cursory research did at that time, it was deemed that existing tools for automated incident processing were too complex and not mature enough (AbuseHelper, CIF) or would require a substantial effort to adapt them for our needs (Megatron). For these reasons we developed a new system to store security-related data in a systematic way and share it with other entities - n6. One of the main design goals was to keep the system as simple as possible, making it easy to manage, maintain, and extend.

The system's architecture is built around a central repository which functions as an archive for all types of collected information. The storage and processing is file oriented - different subsystems put and read whole files from the repository using a well-defined protocol and process them according to their function. The protocol defines access methods and the strict structure of the file system, so given a type of information and a time period, one can derive an unambiguous location to the corresponding file. In the current implementation of n6, standard system utilities - SCP (Secure Copy) and file manipulation commands invoked through SSH (Secure Shell) - are used for accessing the repository, however it would not take much effort to provide an HTTP interface using an identical file identification scheme. Apart from the basic methods for inserting and retrieving data from the repository, n6 has a built-in notification mechanism that is used to inform subscribed subsystems whenever a new file has been added to the repository. Thanks to this feature, the system can work asynchronously, processing new data as it appears almost in real time.

Most of incident reports in the system come from external sources. Each source is handled by a dedicated Python or shell script, that is responsible for fetching the data using appropriate transport mechanism (e.g. HTTP, mailbox, web scraping) and putting it in the repository. These scripts use shared libraries for integration with n6 and performing common tasks like decoding encrypted files or downloading files, so the source-specific code is very brief and new feeds can be easily added.

Data from selected sources is enriched on the fly by adding information that can be used later to determine which organization is affected by the particular incident. Currently implemented enrichment is limited to resolving domain names through a caching DNS proxy and adding ASNs corresponding to IP addresses. This additional information is merged with downloaded files but original contents of individual entries (incidents) remain unchanged and clearly separated from the appended data. One of our requirements for n6 was to preserve the original raw data for reference, therefore the system does not normalize incident reports before saving them in the repository and all modifications are easily reversible. This approach guarantees that we never lose any information due to limitations of our representation or misinterpretation of the incoming data. Storing the original contents of all reports has also the following advantages - we can introduce changes in n6 without having issues related to backwards compatibility and it is possible to read this data using alternative processing systems (e.g. Megatron).

Every file put in the repository containing incident reports is passed to the **filtering engine** which uses a flexible tagging system to determine which organizations should be notified about incidents (figure 3). The first stage of selecting recipients of the data is done on the source level - each partner organization is defined in the system’s configuration and has a set of tags assigned. These tags are matched against tags associated with the source that generated the file, and only if there is at least one match, the organization will be included in the distribution process. The second stage is performed on the level of individual reports. The engine passes the input file through a **preprocessor** - a source-specific script that splits its input into individual entries and extracts certain tags (metadata) corresponding to IP addresses, ASNs, domains, URLs, etc. These tags are again matched against partners' configuration and distributed to them if their tags match. It is also possible to negate tags creating exclusion lists, i.e. instruct the engine to omit certain entries that otherwise would be distributed to a particular organization. This feature is used to implement a simple disclosure policy through restricting distribution of information on certain networks, domains, etc., but sharing the rest of entries.

Filtered data directed to a particular partner organization is stored in its private directory, accessible over HTTPS with certificate based authentication. That directory contains original files, usually in text format, but only records relevant to the partner are kept. Data is split in subdirectories by date and kept for 6 months for reference. The access method and organization of files was chosen specifically to allow easy integration of automated tools with the system.

Since n6 is often used to handle non-volatile, important operational data, reliability of the solution was also one of the initial requirements. All major components have built-in basic fault-tolerance mechanisms that allow the system to handle crashes of any of its components without losing data.

CERT Polska created n6 to address our problems with effective data management and distribution. So far no effort has been taken to make the software more generic and easier to use outside our environment, therefore it has not been shared with general public or other CERTs yet. It was developed in-house with very small resources - approximately 4 man-months - and is in production use since December 2011.

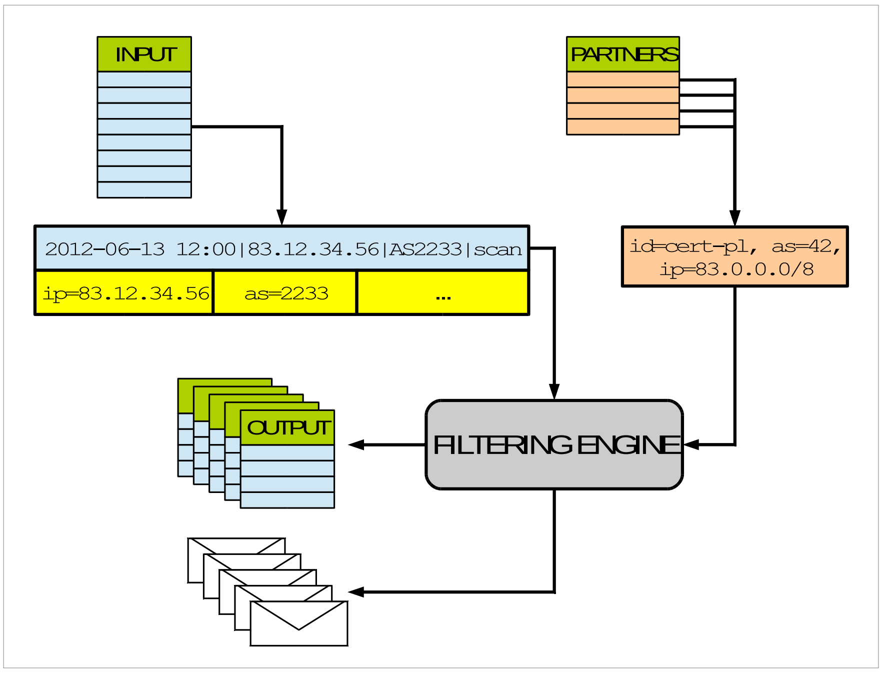
INPUT
PARTNERS
2012-06-13 12:00|83.12.34.56|AS2233|scan
ip=83.12.34.56
as=2233
...
id=cert-pl, as=42, ip=83.0.0.0/8
OUTPUT
FILTERING ENGINE

Figure 3: Tag-based filtering in n6.

4.5 Application of a generic analytics software for handling security data

In August 2012 our team made an experiment, attempting to recreate most of the n6 functionality related to incident sharing using an existing event-oriented analytical platform that provides ready to use component and functions. The goal of the experiment was to estimate how much development time can be saved if existing general-purpose software is used as a base for a simple incident-sharing system similar to n6 and does it have any advantages over systems built specifically for handling security events.

We created our solution on top of Splunk - commercial monitoring and analytics software capable of scaling to large volumes of data [24]. Splunk collects and indexes variety of machine-generated data and provides mechanisms to search it, generate reports, dashboards, graphs and other visualizations. A proof-of-concept system that realizes filtering of data for partner organizations and displays dashboards with the summary of the current operational situation and system's diagnostic information was completed in less than 2 man-weeks, with no prior experience of working with the analytics platform.

Splunk can read input data from static files in batches or through real time feeds. Nevertheless, it does not provide facilities that would be helpful in integration with external sources of incident data, so in our prototype we used scripts from n6 to fetch data to the local file system. Splunk reads them asynchronously as they appear and is responsible for further processing.

Once the input files are indexed by the software, it is possible to perform complex queries on the archived data using a dedicated **search language**. By default, Splunk offers effective full text searches and does not extract metadata from input files except timestamps. However, an operator can define field extraction rules that will be used to associate metadata with individual events on the fly and we leveraged this mechanism to extract IP addresses, ASNs and other properties. Definition of field extraction can be done through a wizard in the graphical user interface so it does not even require constructing regular expressions by hand. Splunk is also able to enrich events by looking up additional metadata fields from external sources. In our prototype we used this feature to add GeoIP data for IP addresses and to map ASN and IP information to constituencies of partner organizations.

When the association between events and affected organization is established, we take advantage of the built-in alerting feature of Splunk to send notifications to appropriate entities. Since the platform is capable of stream processing, these alerts are generated in real time when incidents appear. Alternatively, a periodic query could be scheduled for that purpose. We set up Splunk to run our custom script whenever a new alert is raised. This script takes the all data related to an alert on input and appends entries to the relevant file in the directory of the affected organization, simulating behavior of the n6 filtering engine described in the previous section.

Although the prototype developed in the course of the experiment has never been deployed in production, we gathered enough experience to provide a rough estimate of how much effort would be saved by adopting this approach instead of building n6 from scratch. Further development of the software from the proof-of-concept to a production service that would have all major features of n6 with regard to reliability and filtering would not take much less resources than the initial development of n6 did. This conclusion is partially drawn from the fact that by using a general-purpose software we would have to modify and extend its default behavior when it does not meet our requirements - e.g. custom scripts for sending email notifications would have to be used instead of built-in Splunk functionality due to its lack of flexibility. However by basing our solution on an existing analytical platforms we have access to a lot of features missing from n6, especially with regard to monitoring, situational awareness, and trend analysis. These aspects of systems for automated processing of incident reports are studied in more detail in section 5.2, however usefulness of analytical features for CERTs cannot be underestimated and this is the biggest advantage of the approach described in this section.

The reason to choose Splunk as the base of the experimental incident processing solution was the number of ready to use features provided by the platform and the fact that the basic version is available free of charge. Alternative platforms for log processing could be used instead but they were not evaluated for this purpose. For example the Enterprise Log Search and Archive (ELSA) is an interesting open source log collection and analysis system inspired by Splunk [25]. It seems to be actively developed (e.g. recent CIF integration), and may be worth testing before developing a solution based on commercial software.

5 USE CASES FOR AUTOMATED INCIDENT HANDLING

Systems presented in the previous section have different design goals and methods of dealing with security events. In order to meaningfully compare them and evaluate their usefulness form the CERT perspective, we will describe how they implement two primary functions expected from this class of tools: data sharing and analytics.

5.1 Data sharing

The main purpose of sharing data is to effectively deliver early warning information about threats in networks belonging to the CERT's constituency. The end goal is, of course, mitigation of such threats. Since national/government CERTs often work as proxies and are not able to tackle incidents within their constituencies on their own, they must forward incident reports to the affected organizations, which is the perfect use case for introducing automation in the process of data exchange.

5.1.1 Methods of distributing incident reports

Incident handling systems provide different mechanisms for distributing data about incidents. The form of an incident report depends on the expected recipient - if it will be read by a human, then email with textual information is probably the most universal method for communicating the information. Such approach is taken by Megatron - it uses feed-specific templates with free-form structure, usually containing fragments of original reports for reference.

When the intended recipient is some kind of automated system, the data should be sent in a form that is easy to parse and its interpretation should not be difficult, e.g. use well-defined field names. AbuseHelper has the mailer bot that transforms any set of events into a file in the common CSV format and attaches it to the outgoing message. Additionally, it can provide a real time event feed over XMPP - its native format. A potential problem with this approach is that semantics of event properties may not be clear for the receiving party and has to be communicated out of band.

N6 shifts this problem to individual sources - it provides files with original contents but the actual records are filtered for each organization. The original files usually come in variants of the CSV format and either have a brief header that explains what kind of data they contain or such header is added by n6 itself. Our solution does not impose any interpretation of the data on the recipients - they gain access to raw contents of feeds. Unfortunately, as a consequence, they have to work with many different sources, which requires more effort on their part. Apart from the raw files available through HTTPS, n6 sends notifications to partner organizations when new incidents appear. These messages do not contain contents of incidents and are meant for human operators.

The Splunk-based prototype offers functionality similar to n6 except sending notification emails, so it will not be described here. It could be improved without much effort thanks to its extensibility.

Perhaps the best approach with regard to the format of shared data was taken by CIF - the client application supports multiple output formats, including both human readable and designed for machine processing, like IODEF+JSON or CSV. It is also the only system with an explicit disclosure policy and confidence estimation associated with incidents, which can be used when generating data feeds that will be shared with other organizations. The main issue with CIF when compared to other systems, is that it lacks the concept of a partner organization. As a consequence, it does not provide even a basic functionality related to distribution of incident reports. In particular, the system does not generate partner-specific feeds, does not have facilities to host files or send emails with reports. Thanks to the REST API implementation of these features should not take much effort, however it has to be performed independently by the CERT deploying CIF for distributing incident reports.

5.1.2 Selected properties of data sharing solutions

Apart from data distribution methods described above, automation systems can be evaluated using data quality criteria, based on properties listed in section 3. Since our Splunk-based prototype does not offer any functionality related to data sharing that is not present in n6, this solution is not present here.

Enrichment

All of the evaluated systems are capable of enriching raw incident information to some degree. Megatron and n6 add missing IP addresses and ASNs primarily to correlate incidents with the organizations that they affect, and do not necessarily provide this information to recipients. AbuseHelper uses multiple independent expert bots to gather additional information on input events. Currently implemented bots include passive DNS, GeoIP, Team Cymru and ISC queries. Additional data is recombined with the original event and can be processed further. A flexible set of enrichment methods is also implemented in CIF, with focus on fetching A and NS DNS records and querying external databases (Team Cymru, Spamhaus). It is possible to export all of this data as a CIF feed with appropriate output format.

Correlation

None of the described systems have built-in event correlation capabilities. Architecture of some solutions, e.g. CIF, make correlation of new event with archived incidents easier to implement, but no such functionality has been released yet.

Confidence levels

Of all systems presented in this paper, only CIF has an explicit concept of confidence, which is associated with every event in the system. Simpler systems - n6 and Megatron - are limited to including information about expected validity of the data in textual descriptions of individual sources. AbuseHelper does not assign confidence ratings to events but this feature could be implemented simply as an additional attribute.

Filtering

All systems have different capabilities with regard to selecting which types of information should be received by particular organizations. N6 has a simple tagging system where data sources can be grouped together and the configuration explicitly defines which partners should have access to these groups. Megatron uses a more limited approach based on priorities of events - it is possible to specify the minimal priority of events that will be shared with a particular organization. AbuseHelper can be configured to apply an arbitrary routing of events - the bot responsible for filtering data for specific clients accepts a set rules that control where new events should be directed. Nevertheless, the system does not provide any facilities that would simplify management of a large number of partners, hence maintaining the configuration may be problematic in the long term. Definitely CIF offers the broadest spectrum of selection options for generating event feeds - a query may specify desired impact, severity, confidence level, and disclosure restrictions of incidents. Lack of explicit source filtering should not be a problem, since input data has configurable confidence level and disclosure policy.

Feedback mechanism

Architectures of the evaluated systems are not designed to receive feedback from partner organizations related to individual events or whole data sources, so no such feature is provided.

Long term storage

Megatron and CIF use SQL databases for storing persistent data and should have no problems with storing data for the long term, as long as there are no major schema changes. However, if a CERT receives a large number of feeds, traditional SQL databases may have scalability issues. Both systems lack mechanisms for event deduplication and will store the same report as many times as it was fetched. This problem mostly affects CIF, since it is designed to deal with a large number of diverse feeds.

Since n6 stores all fetched feeds in separate files, it does not have problems with storage scalability - even if a normal filesystem becomes a limitation, it can be moved to a distributed one. Additionally, built-in fault tolerance mechanisms mean that the chance to lose information due to n6 downtime is kept low. All records are kept in original form or can be easily reversed to such, so it is always possible to get back complete historic data.

Long term storage is not part of the AbuseHelper design - it is focused on real time, transient data and by default does not gives any reliability guarantees. Interface to a database may be implemented by a separate bot but it is not a part of the core system and the only standard mechanism for data persistence (archive bot) has very limited capabilities.

Anonymization

All presented systems can be configured to hide the original source of an incident report. They do not, however, have more fine-grained anonymization, e.g. selectively removing IP addresses from individual reports.

Data collection method

Megatron and n6 provide a very basic method for explaining contents of an automatic incident report that is sent to a partner organization - they attach a predefined source-specific header before the list of events. Others do not have such option and any additional information must be communicated outside of the system itself.

Timely delivery

Among described automation systems, AbuseHelper excels at minimizing the latency between receiving and forwarding an incident report. The whole framework is oriented for stream processing in real time, and the overall timeliness of reports depends only on data sources and the notification policy (whether partners require periodic feeds). The only concern with the system is related to its throughput - the XMPP adds a lot of overhead for each event and a single broker must relay all messages exchanged between bots. If the communication channels get saturated, latency introduced by the system may increase significantly.

Megatron uses a more traditional, batch-oriented approach - its scripts for downloading feeds are run periodically. After a new file is downloaded, the main application is executed in order to process all records and notification emails are sent immediately.

n6 works in similar manner when dealing with sources published periodically, but for other input feeds, e.g. email, it supports asynchronous processing. In the latter case, once the complete file has been put in the repository, it is passed to the filtering engine with only minimal delay.

The current version of CIF does not support stream or asynchronous processing - input data is fetched periodically and in order to generate output feeds the server must be polled with the desired frequency. Such approach introduces extra latency to the data distribution process.

Data aging policy

All reviewed systems accept blacklists as one of the possible inputs, but they are simply treated as a list of incident reports and specifics related to the age of entries and the removal policy are disregarded. Tools that pass original log entries to recipients (Megatron, n6) will distribute this information if it is present in the source, but they do not interpret it. Output feeds generated by CIF can be used directly for blocking malicious traffic, however the policy for entering and leaving these lists is very simple – only recent incidents with high confidence are included – and do not take into account other factors, e.g. history of given addresses.

5.2 Analytics

Apart from sharing data, the other essential role of a system for automated incident processing is providing **threat intelligence**, which we understand in this context as insight into the current security situation within the CERT's constituency and overall trends important for effective incident mitigation. This service can also be seen as a form of early warning concerning security incidents.

5.2.1 Situational awareness

The situational awareness of a CERT means that it is able to properly identify most important threats and their characteristics, so appropriate preventive measures can be introduced. Probably the most common method for presenting such information is displaying a list of known types of malware, sources and targets of attacks, etc., sorted by the number of related incidents in the selected period. Even such basic statistics should reveal what services are currently most threatened or which sources should be considered for blacklisting.

Of course more advanced analyses can be performed to extract useful operational information. An example of such technique is ranking autonomous systems by their relative threat, derived from the number of attacks and the estimated number of IP addresses assigned to it [28]. Another possibility is tracking botnets in order to determine where the most infected users are located and how does the CERT's constituency compare to other areas with regard to the infection rate.

Apart from dealing with known threats, an important benefit of operational analytics is the ability to spot new ones quicker than it is possible otherwise. Thanks to the large amount of data that can be put into the automated system, analysts may observe emerging patterns that would be difficult to detect when dealing with incidents individually.

TC Console [26] is a good example of a situational awareness system, however it is available only as a service, without possibility of using other data sources. It provides a graphical web interface that displays a variety of information related to the user’s network. A user can see what types of malware was recently detected on his or her network, how does the malicious activity from the network change over time, and other statistics based on the Team Cymru dataset. Information is displayed in the form of tables, graphs, and lists of individual incidents. Additionally the service can export selected feeds for machine processing. It is also possible to report incidents and mark false positives directly from the web interface.

Existing automation systems described in this paper do not come with such features, they do not even offer a graphical user interface of any kind. The Splunk-based prototype is an exception - being built upon an analytics platform, it has a lot of features to visualize, report and browse the collected data. We could easily create dashboards displaying situational information, including graphs and maps updated in real time.

Additional software can be integrated with other incident handling systems to gain some situational awareness capabilities. Codenomicon VSRoom (Virtual Situational Room) [27] is an example of an open source solution created for that purpose - it integrates with AbuseHelper feeds and displays events in real time using various visualization methods, including maps, histograms and contingency tables. The software is not limited to monitoring real time feeds and allows fetching events from a persistent database.

Creating a basic situational awareness service around systems based on SQL databases (Megatron, CIF) should be relatively simple, however no such solutions have been made available for public yet. Current capabilities of CIF in this regard are limited to community projects for integrating it with external tools - ELSA and Splunk.

Although the core functionality of n6 does not include operational analytics, we created an SQL database that hold incident properties and relationships between events, addresses, domains, etc. The database works as an optional module of the system - it reads data from the central repository and uses a parser library shared with the filtering engine to extract metadata from original reports. One of the main applications of this component is computation of various statistics, which support the reporting process, e.g. an ASN ranking. These statistics are displayed in tabular form through a simple web interface.

5.2.2 Long-term trend analysis

Achieving situational awareness allows reacting to the current threats accordingly. However, in order to evaluate performance of a CERT and to plan future operations, analyzing data from past months or even years may be necessary. Despite the fact that such analysis is long term, changes in trends can also be viewed as an early warning on what a network is likely to face in the future. Similarly to situational awareness which deals with the current information, there is a variety of methods that can be used to analyze incident data, and we will mention only few selected examples.

Interesting conclusions can be drawn from analysis of **remediation rate** [29] - the proportion of IP addresses that are repeatedly reported in incident reports to the total number of attack sources. This metric can be used to evaluate effectiveness of mitigation of reported incidents.

Historic data can also provide insight into the evolution of threats, allowing to see changes in sources and types of incidents. Moreover, this information can be used to predict characteristics of future threats, e.g. knowing that a particular type of botnet communication mechanism becomes more common over time, the organization may devote more resources to monitor this kind of network traffic.

Unfortunately, none of the investigated systems supported long-term trend analysis of any kind. Such capability can be implemented independently for automation systems that collect incident reports in SQL databases (Megatron, CIF or the optional n6 component). Since they keep events in a normalized form, data can be fetched directly from the database using standard utilities and processed further by custom software. Development of a solution capable of performing a basic set of analyzes using that approach should not require significant resources.

6 CONCLUSION

In this paper, we introduced a practical definition of early warning about network security incidents. Based on an ENISA study, we identified key actions influencing data quality that should be carried out by automated security incident exchange systems to effectively provide such a capability for national/government CERTs. We studied four different systems for automated incident exchange: Megatron, CIF, AbuseHelper and n6, adding an experimental platform based on Splunk as well. However, we discovered that none of these systems is currently mature enough to fulfill all the requirements outlined in this paper. No system performs correlation on data or has the capability of enforcing a data aging policy on incidents it reports. There is no support for feedback mechanisms from end users either, leaving this functionality to out-of-band methods. Another useful feature – the communication of how data was initially collected - is missing in some, limiting trust in the quality of transmitted information. The systems also lack long term trend analysis. Indeed, even situational awareness analytics is generally missing.

Each tested system demands intensive reconfiguration to the specific requirements of a particular CERT – none of the systems is “plug and play”. Megatron is overall easier to deploy than others. CIF is the most elaborate, with many features not present in others. AbuseHelper is the outlier in this set: it is the only one that is provided in the form of a framework. It is the most innovative in terms of architecture, but at the same time the most demanding in terms of effort in implementation and customization to meet a CERT's needs. On the other hand, we discovered that it is possible to build an in-house solution customized to specific needs at very little cost, as was the case with our n6 system and experiment with Splunk. Indeed in our case, this cost was lower than that associated with deployment of Megatron, AbuseHelper or CIF.

In summary, we discovered different design and goals make these systems difficult to compare. To make a systematic approach at evaluation possible, we introduced a number of criteria, primarily based on data quality. We hope that this study and the approach presented can help CERTs and other organizations in selecting a solution. No CERT is exactly the same. Thus which of the above systems is ultimately the optimal choice or whether an in-house solution is best instead, depends on the needs of a specific CERT.

ACKNOWLEDGEMENTS

We would like to thank Wes Young (REN-ISAC), Tor Johnson (CERT-SE) and Adam Kozakiewicz (NASK) for reviewing the document and providing comments, as well as Łukasz Juszczyk (CERT Polska/NASK) for help with testing Splunk.

REFERENCES

[1] SANS Internet Storm Center, https://isc.sans.edu/about.html

[2] P. G. Spirakis, V. Vlachos, V. Karakoidas, D. Liappis, D. Kalaitzis, E. Valeontis, S. Kollias, G. Argyros, *Blueprints for a Large-Scale Early Warning System*, 14th Panhellenic Conference on Informatics (PCI), 2010.

[3] Leurrecom.org Honeypot Project, http://www.leurrecom.org

[4] J. Sander, H.-P. Jedlicka, *Carmentis – Frühe Warnung im deutschen Internet*, 14. DFN-CERT Workshop, Hamburg, Germany, 2007, http://www.carmentis.org/CarmentiS-DFN-CERT-Workshop-2007.pdf

[5] R. Spoor, *A Distributed Intrusion Detection System based on passive sensors*, 18th Annual FIRST Conference, Baltimore, Maryland, USA, June 2006.

[6] A. Kozakiewicz et al., WOMBAT Project, *Early Warning System: Experimental Report*, http://wombat-project.eu/WP5/FP7-ICT-216026-Wombat_WP5_D23_V01_Early-warning-system-experimental-report.pdf

[7] CAIDA – The Cooperative Association for Internet Data Analysis, http://www.caida.org/home/

[8] K. Gorzelak, T. Grudziecki, P. Jacewicz, P. Jaroszewski, Ł. Juszczyk, P. Kijewski, A. Belasovs (editor), *Proactive Detection of Network Security Incidents*, ENISA report, December 2011 [available from http://www.enisa.europa.eu/activities/cert/support/proactive-detection]

[9] Shadowserver Foundation, http://www.shadowserver.org/wiki/

[10] CERT Polska Annual Report for 2011, http://www.cert.pl/PDF/Report_CP_2011.pdf

[11] R. Danyliw, J. Meijer, Y. Demchenko, The Incident Object Description Exchange Format, RFC 5070, December 2007

[12] X-ARF, Network Abuse Reporting 2.0, http://www.x-arf.org

[13] Collective Intelligence Framework, http://code.google.com/p/collective-intelligence-framework/

[14] BitBucket: AbuseHelper, http://bitbucket.org/clarifiednetworks/abusehelper/

[15] n6 - Network Security Incident Exchange, http://n6.cert.pl/

[16] Team Cymru SHA1/MD5 MHR Lookup v1.0, http://hash.cymru.com/

[17] The Spamhaus Project, http://www.spamhaus.org/

[18] Doug Pearson, SES / CIF, Internet2 Combined Industry and Research Constituency Meeting, April, 2012

[19] Wes Young, "*in the cloud. a talk about buzz words…*", August 25, 2012, https://collective-intelligence-framework.googlecode.com/files/2012_GFirst.pdf

[20] Protocol Buffers, http://developers.google.com/protocol-buffers/

[21] The XMPP Standards Foundation: Protocols, http://xmpp.org/xmpp-protocols/

[22] P. Kijewski, ARAKIS – *An Early Warning and Attack Identification System*, 16th Annual FIRST Conference, Budapest, June 2004

[23] P. Kijewski, C. Overes, R. Spoor, The HoneySpider Network – fighting client-side threats, 20th Annual FIRST Conference, Vancouver, June 2008, available from http://honeyspider.net/

[24] Splunk Installation Manual: Hardware capacity planning for your Splunk deployment, version 4.3.3 (August 2012)

[25] Enterprise Log Search and Archive, http://code.google.com/p/enterprise-log-search-and-archive/

[26] TC Console, https://www.team-cymru.org/Services/TCConsole/

[27] Codenomicon Virtual Situation Room, http://www.codenomicon.com/vsroom/

[28] A. Dulaunoy, *BGP Ranking - Security Ranking of Internet Service Providers*, 32nd TF-CSIRT Meeting, Barcelona, February, 2011

[29] R. E. Perlotto, *Recidivism and the Art of Remediation*, SECURE 2011 conference, Warsaw, Poland, October, 2011.