NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
FACULTY OF INFORMATICS AND TELECOMMUNICATIONS

POSTGRADUATE STUDIES

MASTER THESIS

SIEM Optimization Using Honeypots

Thomas Antoniou Ailianos

Supervisor: **Ioannis Stayrakakis,** Professor

ATHENS

January 2014

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Βελτιστοποίηση συστημάτων SIEM με χρήση τεχνολογίας Honeypots

Θωμάς Αντωνίου Αιλιανός

Επιβλέπων: **Ιωάννης Στραυρακάκης,** Καθηγητής

ΑΘΗΝΑ

Ιανουάριος 2014

MASTER THESIS

SIEM Optimization Using Honeypots

Thomas A. Ailianos
A.M.: M1155

Supervisor: **Ioannis Stayrakakis,** Professor

January 2014

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Βελτιστοποίηση Συστημάτων SIEM με χρήση τεχνολογίας Honeypots

Θωμάς Α. Αιλιανός
Α.Μ.: M1155

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Ιωάννης Σταυρακάκης,** Καθηγητής

Ιανουάριος 2014

ABSTRACT

Nowadays, Security Information and Event Management (SIEM) systems are an integral part of an organization’s security infrastructure. Although SIEM is a powerful mechanism, it can be as effective as valuable the information fed into. Given that a SIEM’s optimization is limited to the increase of the number of security devices reporting to it and fine-tune the correlation engine, the only way to truly enhance its performance is the use of external intelligence. We propose the use of high interaction Honeypots to create domestic intelligence and correlate events produced in a real organization's environment. By using extensive logging we are able to identify abnormal behavior produced by ignored threats in multiple devices in the organization's network.

SUBJECT AREA: Information Security

KEYWORDS: optimization, SIEM, correlation engine, Honeypots, Event Management, information security, organization

ΠΕΡΙΛΗΨΗ

Στην εποχή μας, τα συστήματα Security Information and Event Management (SIEM) αποτελούν αναπόσπαστο μέρος της υποδομής ασφάλειας ενός οργανισμού. Παρά το γεγονός ότι τα συστήματα SIEM είναι ένας ισχυρός μηχανισμός, μπορεί να είναι όσο αποτελεσματικός όσο πολύτιμες είναι οι πληροφορίες με τις οποίες τροφοδοτείται. Δεδομένου ότι η αύξηση της απόδοσης ενός SIEM περιορίζεται στην αύξηση του αριθμού των συσκευών ασφαλείας που στέλνουν τις καταγραφές (logs) τους σε αυτό και την βελτιστοποίηση του μηχανισμού συσχέτιστης (correlation engine), ο μόνος τρόπος για να ενισχυθεί πραγματικά η απόδοσή του είναι η τροφοδοτησή του με εξωτερικές, ως προς τον οργανισμό, πληροφορίες. Προτείνουμε τη χρήση honeypots υψηλής αλληλεπίδρασης για την δημιουργία τοπικής νοημοσύνης και τον συσχετισμό τους με γεγονότα που παράγονται στο περιβάλλον ενός πραγματικού οργανισμού. Με τη χρήση εκτενών καταγραφών είμαστε σε θέση να αναγνωρίσουμε τη μη φυσιολογική συμπεριφορά, που παράγεται από μη αναγνωρισμένες ως τώρα απειλές, σε ποικίλες συσκευές ασφαλείας (security devices) στο δίκτυο του οργανισμού.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Ασφάλεια Συστημάτων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: βελτιστοποίση, SIEM, Honeypots, οργανισμός, μηχανισμός συσχέτιστης, διαχείριση καταγραφών, ασφάλεια πληροφοριών

TABLE OF CONTENTS

CHARTS CATALOG

FIGURE CATALOG

MATRIX CATALOG

ΠΡΟΛΟΓΟΣ

Η παρούσα διπλωματική εργασία πραγματοποιήθηκε το 2014 με επιβλέποντες τους κ. Στραυρακάκη, καθηγητή του τμήματος Πληροφορικής και Τηλεπικοινωνιών του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών και κ. Ξενάκη, επίκουρο καθηγητή του τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιά. Ευχαριστιές για την συνεργασία τους και την υποστήριξη τους, που με την πολύτιμη βοήθειά τους ξεπεράστηκαν καίρια εμπόδια στην περαίωση αυτής της έρευνας.

1. INTRODUCTION

1.1 Introduction

Due to the increasing level of malicious activity seen on today's internet, organizations are spending a fortune in order to acquire the state of the art security mechanisms such as Intrusion Prevention Systems (IPS), Intrusion Detection Systems (IDS), corporate Antivirus Engines (AV), endpoint protection systems and content filtering, etc in order to protect against the latest threats. As all these security mechanisms, sooner or later, will fail to successfully protect the organization's valuable assets, a new technology has come in front in order to gather the information produced, by several different protective systems, and to detect abnormalities in the system's behavior.

1.1.1 SIEM

This new technology is called SIEM, which stands for Security information & Event Management and describes the product capabilities of gathering, analyzing and presenting information from network and security devices. These devices can be grouped in general as below::

- Identity and access management applications
- Vulnerability management and policy compliance tools
- Operating system, database and application logs
- External threat data

Even though SIEM is a powerful mechanism, it can be as effective as valuable the information fed into. Given that a SIEM’s optimization is limited to the increase of the number of security devices reporting to it and fine-tune the correlation engine, the only way to truly enhance its performance is the use of external intelligence. One method is to provide to the SIEM, lists of compromised, hostile or Third-generation Onion Routing (TOR)[7] network nodes, in order to perform a correlation, with logs gathered from the multiple event sources (AV, operating systems, firewalls etc). However, by this approach, we are able to detect the source and the destination of the attacks, but not their actual content and behavior..

1.1.2 Honeypots

Another approach to enhance the information, fed into the correlation engine, is the use of Honeypots. Honeypot as defined in [10] consists in an environment where vulnerabilities have been deliberately introduced in order to observe attacks and intrusions. Honeypots have been shown to be very useful for accurately detecting attacks, including zero-day threats, in a reasonable cost and reduced false positives, unlike other security systems such as IPS/IDS. The information acquired is valuable for security analysts to correlate local threats, with threats appearing in the region or even better worldwide

By deploying multiple Honeypots in different geographical regions in different types of organizations (public, private) with different missions (banking, telecommunications, health, universities, government etc.) we will be able to have a real-time visibility on attacks taking place worldwide. Using a large-scale honeynet will provide us information for upcoming attacks against production environments in a similar way as performed against the Honeypots.

However, maintaining a large-scale honeynet requires that no actual compromise will take place by third parties. This problem is getting worse in case multiple instances of fake services are being deployed in different clusters of the honeynet. Finally, the requirements are getting even more complex when having a constantly developing honeypot engine, with multiple sensors, targeting to continuous adaptation to latest threats.

1.1.3 Target

In response to this idea, we are designing a system that combines the power of the SIEM correlation engine and the Honeypots in order to enhance the information fed into the SIEM. With the information produced by Honeypots and performing correlation with actual events taking placing in an organization's environment, we can enhance the organization’s protection providing:

- Early alerting on attacks initiated by known external sources.
- Early alerting on attacks initiated by unknown external sources but using known techniques.
- Early alerting on a upcoming attack due similar behavior on other similar organizations in means of location(national, regional (Europe, Middle East and Africa (EMEA)), worldwide), organization type(public, private sector, universities), and organization mission(military, government, health, insurance, banking).
- Identification of internal malicious behaviors that were classified as legitimate.
- Identification of zero-day exploitation methods in order to propagate into the network.

In order to successfully implement the new system to be successful, we should overcome the disadvantages of keeping a large scale honeypot, properly integrate Honeypots with SIEM solution, create multiple attack scenarios and meet the requirements as presented in the section below.

2. DESIGN REQUIREMENTS

This section describes the requirements of the system to be implemented in order our approach to be successful:

2.1.1 Domestic Intelligence

Using Honeypots should produce Intelligence we call Domestic Intelligence. This means that multiple nodes should be deployed in different geographical regions in different type of organizations (public, private) with different missions (banking, telecommunications, health, universities, government etc.). This way we will be able to identify different attack formations taking place in different organizations and regions. As a result early warning can be triggered in case a known attack source or a known attack formation is initiated against in similar organizations in means of mission, location or size..

2.1.2 Easily manageable

As stated before the honeynet should be deployed in multiple organizations in different regions. As multiple honeypots should be deployed, special care should be taken to reduce the administration overhead for maintaining a large-scale honeynet. For this reason the system should be easily manageable using a centralized architecture were all nodes will redirect their incoming network traffic towards a main honeypot engine. Every node has to be set up only once, without extra administrative overhead for maintenance.

2.1.3 Non-compromisable nodes

To acquire better sampling of the malicious traffic, nodes should be hosted in real organizations networks side-by-side with real production systems. To avoid possible security breaches, nodes should be non-compromisable and no actual activity should take place. For these reasons the design should include only nodes which are only responsible for redirecting the incoming traffic to the main honeypot engine. Thus, no actual risk will exist for attackers to take control of the node.

2.1.4 Low cost nodes

As the design includes nodes, which will only redirect the incoming traffic, the cost of acquiring them should be very low. The computational resources should be covered even from ultra low specs systems such as Raspberry[2].

2.1.5 High interaction engines

As stated in previous work[6] high interaction Honeypots should be used. The attacker should be able to take full control of a service, escape to the operating system, escalate to administrator and penetrate to other systems connected to the organizations networks. High interaction should be used in order to identify the attacker's behavior, in all stages of exploitation. In all exploitation phases, logs should be created by multiple security sources so as to be correlated with local events and identify existing malicious systems.

2.1.6 Integration with SIEM and Live Forensics

The system should include multiple security components such as proxy, firewall, Domain Name System (DNS) etc to successfully monitor all network activity created by an attack, taking place against a high-interaction honeypot. Visibility should remain in

high levels in all exploitation phases in order the security analyst to able to monitor the full behavior in all phases of an attack. The SIEM should be able to correlate security logs produced, with actual logs produced in real organizations. With the "recorded" behavior the SIEM will correlate actual events taking place in real organizations to identify external or internal ignored threats.

2.1.7 Modular

The honeypot infrastructure should be highly modular. This means that services should be able to be added, modified and deprecated without any administrator change on the nodes. The honeypot engine should consist of multiple operating systems and network services, which are constantly changed to be adapted to the latest threats and to simulate multiple network scenarios during high-interaction attacks. Incoming traffic should be redirected to the appropriate network service depending on the analysis phase.

2.1.8 Honeypot environment invisibility

In the attacker's point of view, an organization's asset will be attacked. The existence of the main honeypot engine should remain hidden. Additionally, if the attacker takes control of the node, the logging activity should remain hidden so as the protected environment not to be revealed.

2.1.9 Offline forensics

Our system should rely on virtualization environment, giving the ability to the security analysts to revert back to the infected system and perform filesystem forensics. This is acquired taking a snapshot of the infected system after the attack has taken place and before reverting the system to a clean state. In such a way more information can be gathered while profiling will created in order to support future investigations and identify similar attacks.

2.1.10 Multiple attack targets

The system should be able to handle multiple attacks taking placing in the same time by different sources against the same network service. For this reason the system should include an orchestration server which will redirect an attack to an instance of the service taking into consideration the source of the attack.

3. CORE ARCHITECTURE

To meet the aforementioned requirements our approach consists of a centralized system with multiple nodes redirecting the incoming traffic to the main honeypot engine. Nodes are deployed in different types of organizations (public, private, universities) in different geographical regions, in carefully selected address spaces in the Internet. This way our sample traffic remains highly representative including the attacks targeting the organizations to be protected. In this section we will describe all properties of the components including in the solution:

3.1 Nodes

Every node is installed in a different organization and has been assigned a single unused ip address of demilitarized zone (DMZ) network range of the organization. In this way the honeypot is logically located side-by-side with the actual production systems of the organization receiving almost the same malicious traffic with the original systems.

To reduce the computational requirements, every node is deployed on top of a stripped down linux based distribution. Useless packages such as Graphical User Interface (GUI), editors and drivers for unused modules have been removed, keeping only necessary packages for network operations and Virtual Private Network (VPN) services. As a result computational requirements are kept to negligible levels and node software can be hosted even on low-spec devices such as Raspberry.

The network requirements of each node are kept limited to avoid the creation of complex communication paths within the organization's network. More specifically, each node should be configured to have unlimited incoming connectivity while the only outgoing communication path required is the connection to the main Honeypot engine. The network security devices should deny connections initiated from the node towards the organization's internal networks for security reasons. Moreover, for administration actions the unlimited inbound connections are used.

To acquire connectivity to the main honeypot engine each node has an established point-to-point VPN connection using unique credentials mutually agreed and configured on the VPN concentrator deployed. The incoming – probably malicious – activity is redirected to the main honeypot engine through the established VPN tunnel. In this way we avoid complex port forwarding to redirect the incoming traffic, we limit the established connection with the main honeypot engine to only one and we avoid possible flood of the Organization's network with unnecessary packets. To redirect the traffic through the tunnel we take advantage of the powerful linux kernel module iptables.

Iptables is a Linux Kernel Firewall responsible to manage all network traffic passing through the network interface of the system. The installed ruleset is responsible to allow administrative connections while all incoming connections are forwarded using the NAT protocol to the centralized honeypot engine. The installed ruleset is provided in the APPENDIX I .

Using this setup the node remains non-compromisable, using negligible computational resources, requiring minimal network configurations and requiring minimal administration overhead as required in our design. A high level diagram of the applied architecture is displayed below.

.

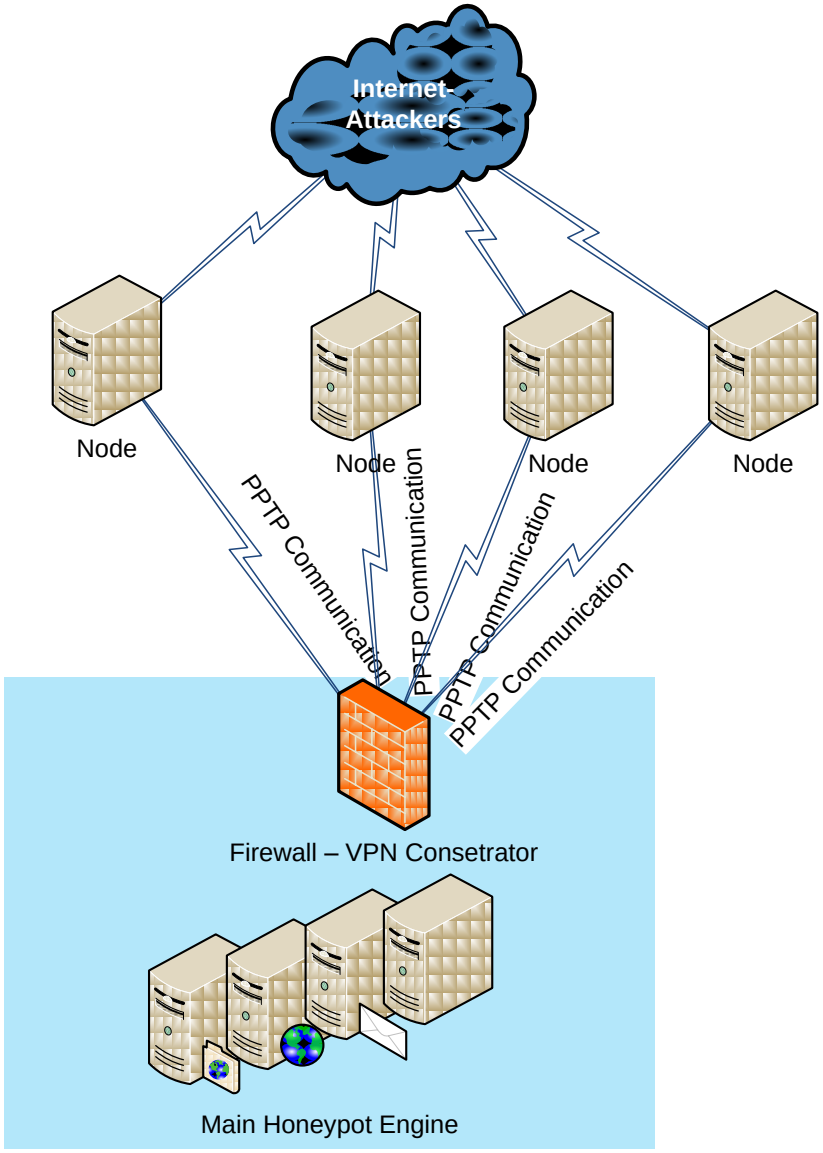

Figure 1: Honeypot Architecture Diagram

3.2 Security - Monitoring Components

The main honeypot engine consists of a variety of security components such as proxy, firewall, DNS etc. The security components are acting upon the successful compromise of a vulnerable network service to protect the infrastructure of the main honeypot engine, mitigating the attack only in the scope of the built scenario. Moreover, security components are deployed in order to closely monitor and inspect the network activity created during the attacks. Finally, special measures were taken to protect the honeypot engine from being used by attackers to propagate to other internet nodes. The implemented security components are described below:

3.2.1 Firewall - VPN Concentrator

The integrity of the centralized honeypot engine is protected with a firewall, which enforces the following protection measures:

- Restricts the interconnections between different nodes.
- Logs every connection between the node and the main engine in order to record in detail the full network behavior of each attack.

- Restricts the outgoing access from the main engine to the Internet, while simulates a typical organization network allowing HTTP traffic and DNS queries towards nodes in the Internet.
- Plays a significant role on creating multiple scenarios simulating different corporate networks

The VPN concentrator is deployed to establish the point-to-point VPN connections from the nodes towards the main honeypot engine. Through the established tunnel all incoming traffic arriving to the node is forwarded to the honeypot engine.

3.2.2 Proxy Server

The centralized honeypot infrastructure also includes a proxy server, which inspects and logs in detail, all the outgoing HTTP traffic to Internet targets. The role of web proxy is dual: Firstly, it filters any malicious outgoing activity and secondly it successfully logs any activity created including request and response headers and body. As a result all files downloaded during an attack are kept for further analysis, while http traffic is subject to extended analysis to identify custom communication protocols.

Furthermore, in order to hide the existence of the proxy no configuration takes place in the high-interaction honeypots. This is acquired by configuring the proxy server in transparent mode with the cooperation of the firewall through the use of the appropriate configuration. When a connection is initiated towards Transmission Control Protocol (TCP) ports used for HTTP communication (80,443), the firewall redirects the traffic towards the deployed proxy server. Then the server connects to the target Uniform Resource Locator (URL) gets the response and passes it to the origin.

To avoid attackers propagating to other public nodes on Internet a filter is applied to slow down or block any malicious web requests towards the Internet nodes. For this purpose, the well-known module for Apache servers, called Mod_Security[8], is used. Mod_security cooperates with the Apache's engine inspecting HTTP request headers, request body, response headers and response body to indentify malicious signatures. The mod_security engine keeps abnomality scores for each transaction and in case a transaction score passes a predefined threshold then it blocks the transaction to protect against attacks. The ruleset used is the one provided by Open Web Application Security Project (OWASP) as recommended configuration for mod_security. The applied configuration can be found on the APPENDIX III.

3.2.3 DNS Server

The only communication path left available for being exploited, by an attacker in order to communicate with the Internet are the DNS queries. This can take place by using a methodology called "DNS tunneling"[4] where the attacker creates multiple DNS queries for dynamic records with encapsulated data encoded in a legit DNS packet. In order to have access to this communication path, we have deployed a DNS server based on Bind DNS server, configured to extensively log, in detail, every query and response taking place during an attack. Sample of the configuration can be found on the APPENDIX IV.

3.3 Virtualization

All the components of our approach are part of a virtual environment, giving us the ability to dynamically change the networking infrastructure in order to simulate multiple scenarios of corporate network being exploited. Moreover, due to the fact that attacks take place in real operating systems the virtual environment give us the ability to revert

the operating system to a clean state in order to be ready to receive the next upcoming attack. For this reason, we use Vmware ESX product, which offers an Application Programming Interface (API) that can be used in order to programmatically revert the operating system after the attack has finished.

3.4 Honeypot engines

The honeypot engine consists of multiple operating systems of various types hosting various services that will be analyzed later in this document. As the attacks take place in actual operating systems and services it is crucial that the logging level is optimized in order to log every possible action of the attacker. For example in Unix systems both syslog and auditd daemons have been configured properly to log any file access, binary execution and audit change, while custom solutions ensure that all commands executed on the operating are send to the log concentrator (APPENDIX II). In this way we are able to perform live forensics in order to identify the methodology used by the attacker in order to exploit the operating system.

3.5 SIEM

Finally, the last component of the centralized honeypot infrastructure is the SIEM solution called ArcSight ESM. Security logs produced from every security component of the honeypot engine and from the network services are gathered from a SIEM solution in order to be correlated with actual logs produced in real organizations. ESM is a Security Event Manager, which receives logs from different security devices, normalizes the events and based on correlation rules, produces security information notifications if needed. In our project the ArcSight ESM is used in order to collect the events from multiple event sources and produce statistics regarding the honeypot engine operation. Finally, in case an attack is identified special resources are created in order to save all evidence of an attack towards a vulnerable network service. In this way we are able to identify similar behavior produced by unidentified threats in multiple devices in the organization's network.

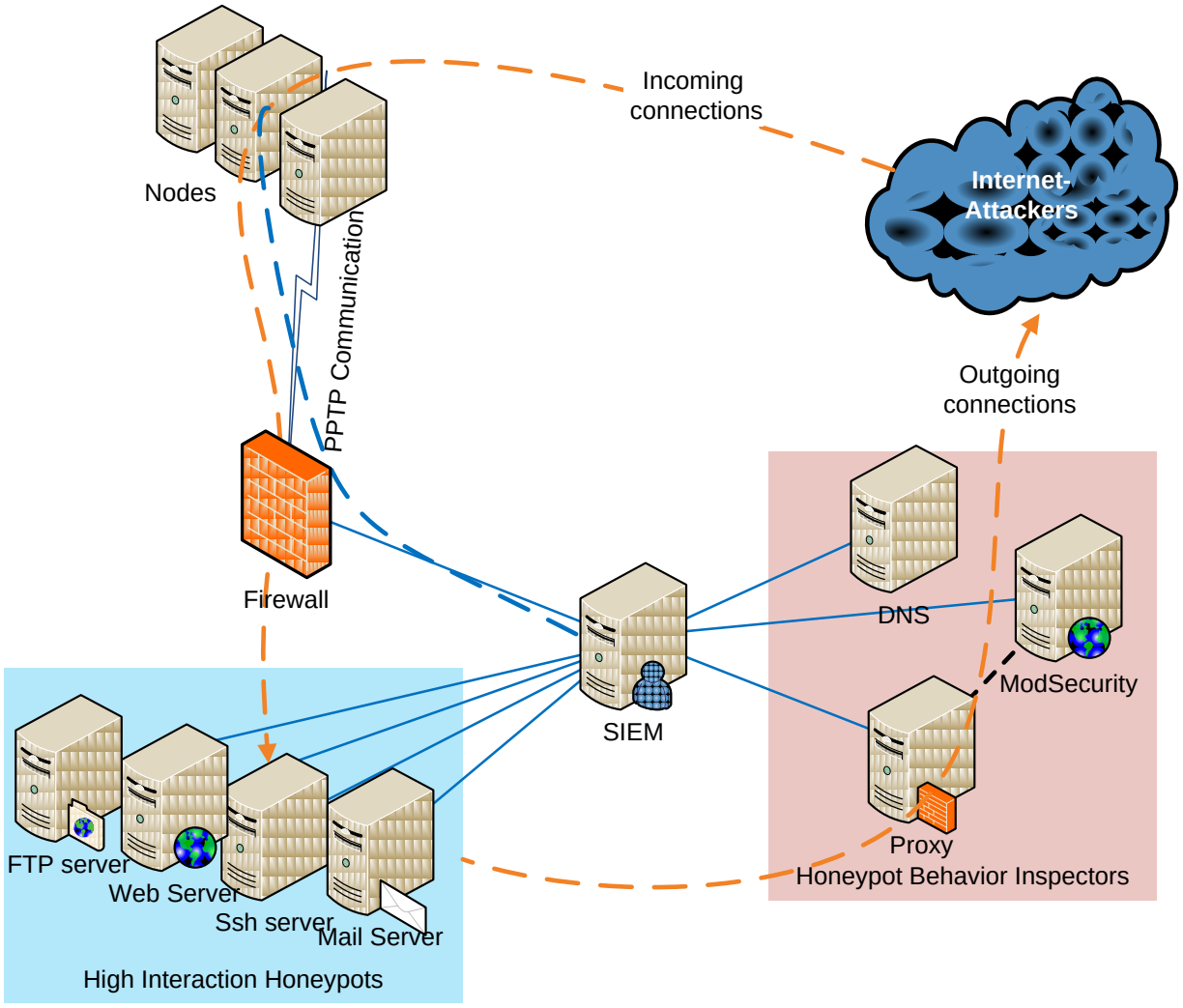

Figure 2: System Architecture

4. METHODOLOGY USED

The honeypot engine consists of multiple selected operating systems, which receive traffic according to the phase of the honeynet deployment. In the beginning of the simulations only low-interaction honeypots were used like honeyd. Honeyd is a very popular and lightweight daemon with many interesting properties such as network emulation. We have used honeyd in order to create multiple fake services and operating systems and identify which of them are subject to attacks more frequently. Moreover, except from the services a first evaluation of the specific Vendors and product has been created based on the destination port.

Based on the popularity of application in terms of users and attack instances, traffic stops being forwarded to the Honeyd engine and it is forwarded to medium interactive honeypots instead. Their goal is to identify popular attack vectors against specific devices (Vendor, Product, service). For example we use the Nepenthes honeypot engine in order to identify the different type of attack vectors used against a Samba server. Using this method we are able to separate brute force attempts and direct exploit attempts. As an additional step the open source services are customized in order to log all login attempts, including usernames and passwords used by the attackers.

The last step of the honeynet project is to create a high-interaction environment according to the popularity of the services and the attack vector of each service. In this phase attackers are able to take control firstly of the service deployed, secondly of the operating system as limited users and finally as system users. As a result, we can explore both exploitation techniques from the service to the operating system and privilege escalation techniques used in different operating systems. In this way we are able to identify the popularity of known exploitation methods but also to identify and explore unknown exploitation methods. This is achieved by using extensive logging on all participating components necessary for the service deployment such as operating system, firewall, proxy server, DNS server as described in previous sections.

Finally, we simulate multiple Organizations environments in order to investigate the attacker's behavior upon the successful exploitation of the operating system and the methodology used to propagate and penetrate inside the simulated organization network. The security components along with the detailed logging on the vulnerable operating systems will inform us regarding every action taken by the attacker.

5. IDENTIFYING AND ANALYZING AN ATTACK

As honeypot nodes have been configured with an unused ip address on the networks to be examined, we can consider every established connection, as suspicious, in matters of severity. Our severity increases when the connection has impact on the operating system or the service, ensuring the malicious profile of the source ip address.

We can consider, as the starting timestamp of an attack, the moment when the first connection establishes on the node. We then assume that the attack ends after 1 hour (upper bound) of inactivity from the same IP address. Upon the termination of an attack the virtual machine hosting the attacked service is suspended and a snapshot is acquired for future reference. Then the virtual disk is externally mounted, in order to export the structure of the filesystem. Additionally, the files are being hashed to identify all changes that took place. Finally, the system is reverted to a clean state being ready to host the next attack.

Investigating an incident requires diving into the logs produced by the firewall, the proxy server and the DNS server. All these components will provide information about the profile of the attacker. Additionally, the filesystem logs, along with the commands executed by the attacker and the filesystem changes will finalize the attacker's profile. Binary files downloaded from the internet or uploaded though the command line are subject to static analysis. This analysis can be performed using open source tools like Cuckoo[3].

Finally, all collected logs are stored in a resource of ArcSight ESM along with a description for future reference. Later in this document we will analyze a case study taken place in real environment.

6. RESULTS

Our system has been deployed on the Internet and has been running for 106 days. The scope included honeypot nodes deployed in 3 universities, 1 private sector organization, 1 public sector organization and 1 DSL. The honeypot engine has received connection from 15000 distinct ip addresses. In the next sections we will describe the main aspects of the deployment.

6.1 Popular targets

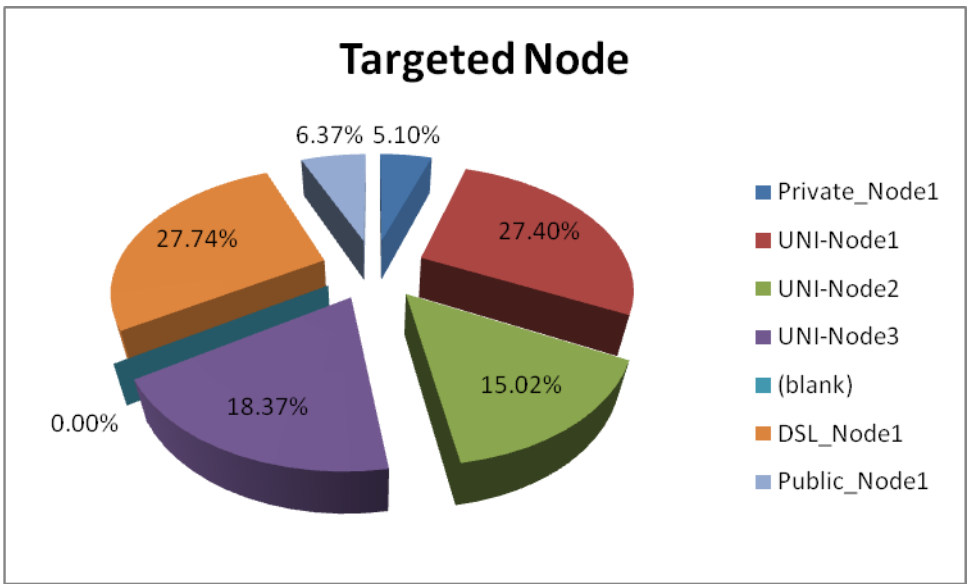

Chart 1: Chart of Targets by popularity

In the above graph we can identify that the main target of attacks are the nodes deployed in the Greek Universities. This can be explained by the fact that Universities network are known to have deployed multiple test systems, which are not properly hardened. Moreover, there are parts of universities networks that are being administrated by not experienced employees or even worse internships, not paying the attention to follow security policies and well-known practices. As a result universities networks are being target to multiple attacks in order to be part of botnets and jump points towards other targets.

On the other hand, DSL nodes tend to be less famous targets. This can be explained by the fact that they are often unstable and that most DSL connections use dynamic addressing, while computers behind a DSL network are partially protected by the embedded firewall of the DSL router. Additionally, DSL connections host mostly desktops, which are randomly operated. As a result we can classify the DSL nodes are as low priority targets for international attackers.

Finally, private and public sector companies are less famous targets as their network address spaces tend to be kept unknown and most of the times networks are more secure.

6.2 Popular targeted services

In the graph below we can identify the top 25 most popular subjects to attack ,network services. This graph is the main tool directing our research and specifying the services that should be deployed to the main honeypot engine for further investigation. As presented below SSH service attacks – mostly brute force attacks – are subject to attack with probability more than 50%.

The blank field refers to Internet Control Messaging Protocol (ICMP) protocol part of which is the echo request packet – known as ping - used as a reconnaissance method to identify the status of a host. A response to an echo request packet verifies that the host is alive, thus the attacker will spend resources to initiate a port scan to identify running network services. For this reason the default network configuration of the nodes permit such ICMP protocol communication in order to encourage port scans and ”invite” attackers.

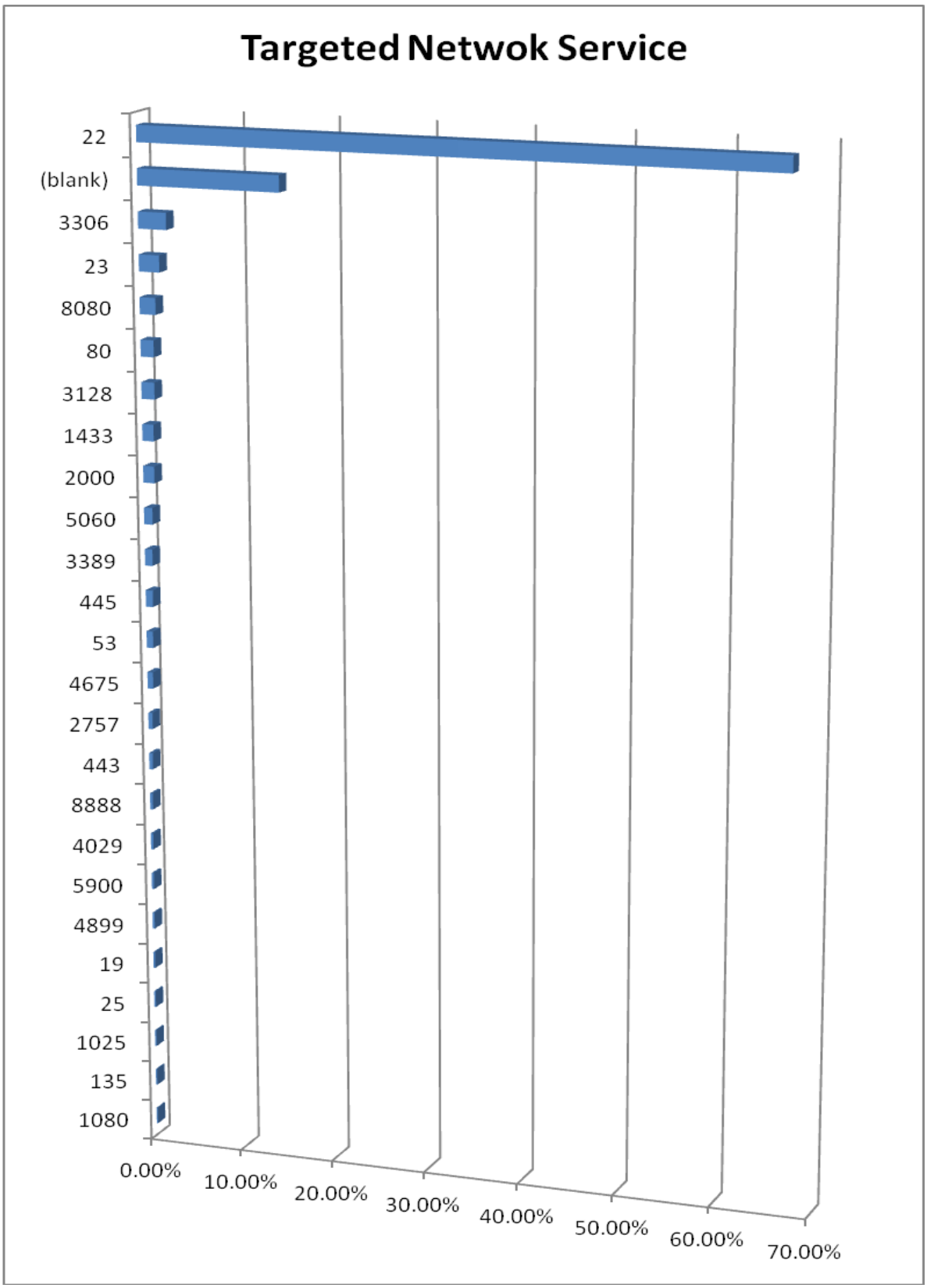

Chart 2: Recorded Attacks by Target Network Service

6.3 Popular attack sources

The graph below presents the top 30 most popular sources of the attacks towards the network services including the type of the target.

As we can identify by the graph below China is the main source of the attacks being the source of the attacks of more than one third of the total amount. Unlikely other attackers, Chinese attackers seem to be equally interested for all type of nodes.

Moreover, although Greece is a small country compared to others, it is relatively high in the list of the attack sources with main targets the Greek universities. This can be explained by the regional knowledge of Greek attackers, being aware of the universities network address space and network configuration. Nevertheless, Greek seem to avoid attacks towards organizations of public or private sector which are probably monitored and logged.

Unlike other attackers, the ones from Germany and France seem to be really interested for DSL nodes instead of large organizations or universities.

Moreover, it is strange that small developing countries such as Chile, Bietnam, Colombia and Brazil are high in the list instead of other developed counties such as Sweden, Japan

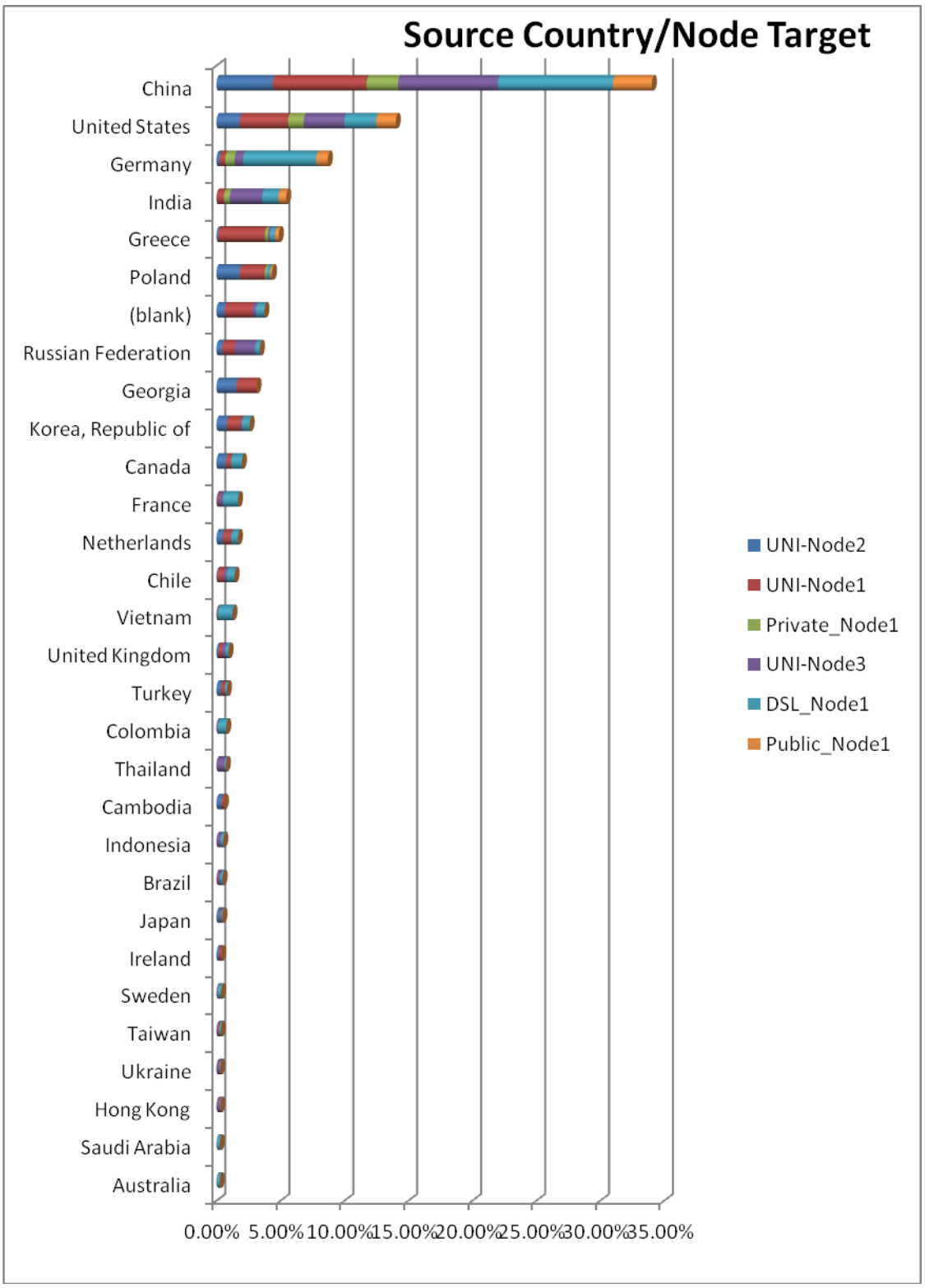

Chart 3: Recorded attacks by Source Country and Target Node

7. USE CASES

During the 3-month operation of the honeypot engine we have identified multiple interesting cases where the attacker successfully compromised the honeypot engine, or his identity was revealed or future actions were identified or just the attacker got our attention for another reason. The most interesting case are presented below:

7.1 Proxy Server – Real information

Due to increased levels of port scanning on port 3128 on nodes deployed in universities networks we decided to deploy a proxy server on the well-known port 3128. No protection was installed during this attack scenario, thus users had unlimited access to the internet. Upon the deployment of the proxy server we have observed that multiple users started using this proxy.

The reason for this increased traffic on the proxy service is that using public facing proxies is a common behavior for attackers to perform web attacks hiding their original source. In order to encourage users to use this proxy, we configured the proxy server to hide the original source (HTTP Header tag: Originated) and spoofed the identity of the proxy to a commercial one. Finally, the proxy had almost unlimited bandwidth as the original system was located on the backbone of the Greek university network. All these characteristics made the proxy a perfect candidate to be used by attackers.

Furthermore, we have identified that one of the attackers used our proxy using google search to identify WordPress and Joomla sites that had installed vulnerable modules. More specifically, the attacker was searching via Google search for particular URIs in order to identify existing files proving the existence of an installed vulnerable module. After – probably – creating a list of targets, the user started attacking the sites found using Google search engine in order to get application administrator rights.

Nevertheless, after a few “working” hours the user made a HUGE mistake. He used the same proxy to book a hotel using his name, identification number and other personal information. Of course this action created a match between the IP address and the real id of the attacker as the proxy was logging the full Hyper-Text Transfer Protocol (HTTP) and Hyper-Text Transfer Protocol Secure (HTTPS) communication passing through the proxy.

Despite the interesting results, we had to close the public facing proxy after a few days of testing in order to avoid a possible abuse report.

7.2 Open DNS - DNS Amplification attacks

Another interesting scenario deployed is the use of DNS servers. Due to an increased activity on port 53, well-known port for DNS service, we decided to deploy a public facing DNS service available for all internet users.

By deploying the DNS service on the main honeypot engine had the same result as deploying the DNS service in all nodes used. Once again more popular targets were the Greek universities networks.

A specific day we identified an increased port scan targeting UDP port 53 followed by some test DNS queries. The following day, we identified that the same attacker used DNS amplification methodology to attack towards a popular telecommunication provider in Greece.

In this scenario security analysts could correlate the events of different days to predict upcoming attacks and identify the original source of the attack towards the telecommunication organization.

7.3 Targeting Operating system – Typical scenario

The most popular attack was the one targeting directly the operating system via the SSH service on its well-known port TCP/22. Multiple attackers were identified performing port scan and to identify a public facing ssh service. After the reconnaissance phase the attackers most times started a brute force attack using dictionaries of popular passwords and dictionaries of already compromised accounts.

After a successful login of the - probably automated – procedure and after a manual login was identified. The typical behavior of the attacker was to stop the history system of the shell system of the operating system (bash or sh) and then install a backdoor – most of the times using cron – in order to secure a reverse shell towards a command and control server. Afterwards, reconnaissance was performed on the operating system in order to identify the exact distribution and version and search for a possible local exploit to escalate to root privileges. In case of a successful exploitation and escalation using administrator rights, a second backdoor was installed to ensure a reverse shell running by a super user. Finally, the typical attacker is trying to penetrate further in the organization's network pointing especially for servers such as domain controllers and fileserver to retrieve valuable information.

7.4 Telephony – VOIP

Upon the depreciation of the dial-up connections to the internet and for a few years the malicious software called dialers responsible for charging users with huge telephony bills were about to be vanished. Nowadays, and with the spreading of the Voice Over IP (VoIP) systems a new type of dialers has been identified which connects to VoIP systems and charges users with international calls. Inspired by the increased port scans of UDP port 5060 we have deployed a VoIP system to examine the attackers behavior. For this reason for every initiated call a fake answer is provided using a random recording. In this way we have created a list of malicious targets of phone calls.

8. FUTURE WORK

8.1 Attack Termination Time and Automatic revert

An automatic clever way should be implemented to identify the end of an attack. Our assumption that an attack stops after a constant time variable is not accurate enough. Moreover, the API of the VmWare ESX should be used to implement an automatic revert mechanism upon the end of the attack.

8.2 Load Balance

The system should be extended to include an orchestration server with the role of advanced load balancer, which will balance multiple attacks to different instances of a vulnerable service, taking place in the same time. This can be acquired using a load balancer, which will redirect an attack to an instance of the service taking into consideration the source of the attack. For this reason attack states should be kept in order to ensure that an attack would always target the same network service.

8.3 Services

New services should be tested and implemented in order to identify the attackers behavior on these. Windws services such as remote desktop (TCP:3389), samba(445) and MSSQL (1433) were not tested due to limitations on customizing and mitigating an attack.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Security Devices	Συσκευές Ασφαλείας
Logs	Κουτσουρα
Correlation Engine	Μηχανισμός Συσχέτισης

SHORTCUTS – ABBREVIATIONS – ACRONYMS

SIEM	Security Incident and Event Manager
HTTP /S	Hyper-Text Transfer Protocol / Secure
TOR	Third-generation Onion Routing
DNS	Domain Name System
IPS	Intrusion Prevention Systems
IDS	Intrusion Detection Systems
AV	Antivirus Engines
EMEA	Europe, Middle East and Africa
DMZ	demilitarized zone
GUI	Graphical User Interface
VPN	Virtual Private Network
OWASP	Open Web Application Security Project
TCP	Transmission Control Protocol
URL	Uniform Resource Locator
ICMP	Internet Control Messaging Protocol
VOIP	Voice Over IP
API	Application Programming Interface

APPENDIX I - Iptables Configuration

iptables -i eth0 -A INPUT -p tcp -m state --state NEW,ESTABLISHED --dport 65123 -j ACCEPT

iptables -A OUTPUT -m state -p tcp --state ESTABLISHED --source-port 65123 -j ACCEPT

iptables -i eth0 -A INPUT -p tcp -m state --state ESTABLISHED --source-port 1723 -j ACCEPT

iptables -A OUTPUT -m state -p tcp --state NEW,ESTABLISHED --dport 1723 -j ACCEPT

iptables -i eth0 -A INPUT -p tcp -m state --state ESTABLISHED -s 83.212.110.123 --source-port 65125 -j ACCEPT

iptables -A OUTPUT -m state -p tcp --state NEW,ESTABLISHED -d 83.212.110.123 --dport 65125 -j ACCEPT

#dns service

#ntp service

#update service

iptables -t nat -A PREROUTING -d 195.134.66.227 -j LOG --log-prefix "IN "

iptables -t nat -A PREROUTING -p tcp -i eth0 --dport 1:65100 -j DNAT --to-destination 192.168.9.100

iptables -t nat -A PREROUTING -p udp -i eth0 --dport 1:65100 -j DNAT --to-destination 192.168.9.100

iptables -t nat -A POSTROUTING -o ppp0 -j SNAT --to 192.168.9.3

iptables -A FORWARD -i eth0 -o ppp0 -j ACCEPT

APPENDIX II - Auditd Configuration Scripts

```
#!/bin/sh
echo ## Binary execution ##
for name in `find / -executable -type f`;
do
echo "-w $name -p xwa -k binary_action"
done

echo ## ETC MONITORING ########
for name in `find /etc -executable -type f`;
do
echo "-w $name -p wa -k etc_modification"
done
```

APPENDIX III - Mod_Security Configuration

```
# -- Rule engine initialization ----------------------------------------------

# Enable ModSecurity, attaching it to every transaction. Use detection
# only to start with, because that minimises the chances of post-installation
# disruption.
#
SecRuleEngine DetectionOnly


# -- Request body handling ---------------------------------------------------

# Allow ModSecurity to access request bodies. If you don't, ModSecurity
# won't be able to see any POST parameters, which opens a large security
# hole for attackers to exploit.
#
SecRequestBodyAccess On

# Enable XML request body parser.
# Initiate XML Processor in case of xml content-type
#
SecRule REQUEST_HEADERS:Content-Type "text/xml" \
    "id:'200000',phase:1,t:none,t:lowercase,pass,nolog,ctl:requestBodyProcessor=XML"


# Maximum request body size we will accept for buffering. If you support
# file uploads then the value given on the first line has to be as large
# as the largest file you are willing to accept. The second value refers
# to the size of data, with files excluded. You want to keep that value as
# low as practical.
#
SecRequestBodyLimit 13107200
SecRequestBodyNoFilesLimit 131072

# Store up to 128 KB of request body data in memory. When the multipart
# parser reachers this limit, it will start using your hard disk for
```

```
# storage. That is slow, but unavoidable.
#
SecRequestBodyInMemoryLimit 131072

# What do do if the request body size is above our configured limit.
# Keep in mind that this setting will automatically be set to ProcessPartial
# when SecRuleEngine is set to DetectionOnly mode in order to minimize
# disruptions when initially deploying ModSecurity.
#
SecRequestBodyLimitAction Reject

# Verify that we've correctly processed the request body.
# As a rule of thumb, when failing to process a request body
# you should reject the request (when deployed in blocking mode)
# or log a high-severity alert (when deployed in detection-only mode).
#
SecRule REQBODY_ERROR "!@eq 0" \
"id:'200001', phase:2,t:none,log,deny,status:400,msg:'Failed to parse request body.',logdata:'%{reqbody_error_msg}',severity:2"

# By default be strict with what we accept in the multipart/form-data
# request body. If the rule below proves to be too strict for your
# environment consider changing it to detection-only. You are encouraged
# _not_ to remove it altogether.
#
SecRule MULTIPART_STRICT_ERROR "!@eq 0" \
"id:'200002',phase:2,t:none,log,deny,status:44, \
msg:'Multipart request body failed strict validation: \
PE %{REQBODY_PROCESSOR_ERROR}, \
BQ %{MULTIPART_BOUNDARY_QUOTED}, \
BW %{MULTIPART_BOUNDARY_WHITESPACE}, \
DB %{MULTIPART_DATA_BEFORE}, \
DA %{MULTIPART_DATA_AFTER}, \
HF %{MULTIPART_HEADER_FOLDING}, \
LF %{MULTIPART_LF_LINE}, \
SM %{MULTIPART_MISSING_SEMICOLON}, \
IQ %{MULTIPART_INVALID_QUOTING}, \
```

```
IP %{MULTIPART_INVALID_PART}, \
IH %{MULTIPART_INVALID_HEADER_FOLDING}, \
FL %{MULTIPART_FILE_LIMIT_EXCEEDED}"'

# Did we see anything that might be a boundary?
#
SecRule MULTIPART_UNMATCHED_BOUNDARY "!@eq 0" \
"id:'200003',phase:2,t:none,log,deny,msg:'Multipart parser detected a possible unmatched boundary.'"

# PCRE Tuning
# We want to avoid a potential RegEx DoS condition
#
SecPcreMatchLimit 1000
SecPcreMatchLimitRecursion 1000

# Some internal errors will set flags in TX and we will need to look for these.
# All of these are prefixed with "MSC_".  The following flags currently exist:
#
# MSC_PCRE_LIMITS_EXCEEDED: PCRE match limits were exceeded.
#
SecRule TX:/^MSC_/ "!@streq 0" \
        "id:'200004',phase:2,t:none,deny,msg:'ModSecurity internal error flagged: %{MATCHED_VAR_NAME}"'


# -- Response body handling -----------------------------------------------

# Allow ModSecurity to access response bodies.
# You should have this directive enabled in order to identify errors
# and data leakage issues.
#
# Do keep in mind that enabling this directive does increases both
# memory consumption and response latency.
#
SecResponseBodyAccess Off
```

```
# Which response MIME types do you want to inspect? You should adjust the
# configuration below to catch documents but avoid static files
# (e.g., images and archives).
#
SecResponseBodyMimeType text/plain text/html text/xml

# Buffer response bodies of up to 512 KB in length.
SecResponseBodyLimit 524288

# What happens when we encounter a response body larger than the configured
# limit? By default, we process what we have and let the rest through.
# That's somewhat less secure, but does not break any legitimate pages.
#
SecResponseBodyLimitAction ProcessPartial


# -- Filesystem configuration -------------------------------------------------

# The location where ModSecurity stores temporary files (for example, when
# it needs to handle a file upload that is larger than the configured limit).
#
# This default setting is chosen due to all systems have /tmp available however,
# this is less than ideal. It is recommended that you specify a location that's private.
#
SecTmpDir /tmp/

# The location where ModSecurity will keep its persistent data.  This default setting
# is chosen due to all systems have /tmp available however, it
# too should be updated to a place that other users can't access.
#
SecDataDir /tmp/


# -- File uploads handling configuration --------------------------------------

# The location where ModSecurity stores intercepted uploaded files. This
# location must be private to ModSecurity. You don't want other users on
```

```
# the server to access the files, do you?
#
#SecUploadDir /opt/modsecurity/var/upload/

# By default, only keep the files that were determined to be unusual
# in some way (by an external inspection script). For this to work you
# will also need at least one file inspection rule.
#
#SecUploadKeepFiles RelevantOnly

# Uploaded files are by default created with permissions that do not allow
# any other user to access them. You may need to relax that if you want to
# interface ModSecurity to an external program (e.g., an anti-virus).
#
#SecUploadFileMode 0600


# -- Debug log configuration ---------------------------------------------

# The default debug log configuration is to duplicate the error, warning
# and notice messages from the error log.
#
#SecDebugLog /opt/modsecurity/var/log/debug.log
#SecDebugLogLevel 3

# -- Audit log configuration ---------------------------------------------

# Log the transactions that are marked by a rule, as well as those that
# trigger a server error (determined by a 5xx or 4xx, excluding 404,
# level response status codes).
#
SecAuditEngine RelevantOnly
SecAuditLogRelevantStatus "^(?:5|4(?!04))"

# Log everything we know about a transaction.
#SecAuditLogParts ABCFHZ
SecAuditLogParts ABIDEFGHZ
```

```
# Use a single file for logging. This is much easier to look at, but
# assumes that you will use the audit log only ocassionally.
#
#SecAuditLogType Serial
SecAuditLogType Concurrent

#SecAuditLog /var/log/modsec_audit.log
SecAuditLog "|/usr/local/modsecurity/bin/mlogc /usr/local/waf/conf/mlogc/mlogc.conf"

# Specify the path for concurrent audit logging.
#SecAuditLogStorageDir /opt/modsecurity/var/audit/
SecAuditLogStorageDir /usr/local/waf/logs/data
# -- Miscellaneous ------------------------------------------------------------

# Use the most commonly used application/x-www-form-urlencoded parameter
# separator. There's probably only one application somewhere that uses
# something else so don't expect to change this value.
#
SecArgumentSeparator &

# Settle on version 0 (zero) cookies, as that is what most applications
# use. Using an incorrect cookie version may open your installation to
# evasion attacks (against the rules that examine named cookies).
#
SecCookieFormat 0

# Specify your Unicode Code Point.
# This mapping is used by the t:urlDecodeUni transformation function
# to properly map encoded data to your language. Properly setting
# these directives helps to reduce false positives and negatives.
#
#SecUnicodeCodePage 20127
#SecUnicodeMapFile unicode.mapping
SecServerSignature Microsoft-IIS/9.0a
```

APPENDIX IV - DNS BIND Server Configuration

```
yum install bind bind-chroot caching-nameserver
cd /var/named/chroot/etc
vi named.conf

options {
        listen-on port 53 { any; };
        directory "/var/named";
        dump-file "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";
        memstatistics-file "/var/named/data/named_mem_stats.txt";
        allow-query { any; };
        allow-query-cache { any; };
};

logging {
        channel bind_syslog {
                syslog local1; #select the facility
                severity debug; #select the severity level
                print-time no; #required by ArcSight
                print-severity no; #required by ArcSight
                print-category no; #required by ArcSight
        };
        category default { bind_syslog;};
        category client { bind_syslog;};
        category config { bind_syslog;};
        category database { bind_syslog;};
        category dnssec { bind_syslog;};
        category queries { bind_syslog;};
        category resolver { bind_syslog;};
        category security { bind_syslog;};
        category update { bind_syslog;};
        category update-security { bind_syslog;};
        category xfer-in { bind_syslog;};
        category xfer-out { bind_syslog;};
};
```

```
view localhost_resolver {
        match-clients { any; };
        match-destinations { any; };
        recursion yes;
        include "/etc/named.rfc1912.zones";
};
```

REFERENCES

[1] Dshield.org, distributed intrusion detection system. http://www.dshield.org
[2] Raspberry Pi, a credit-card sized computer. http://www.raspberrypi.org/faqs
[3] Cuckoo sandbox, a malware analysis system http://www.cuckoosandbox.org/
[4] T. van Leijenhorst, K. Chin & D. Lowe, "On the viability and performance of DNS tunneling," in International Conference on Information Technology and Applications, 2008
[5] F.Pouget and M. Dacier "Honeypot-based Forensics" in AuCERT Asia Pacific Information technology Security Conferenece 2004
[6] I. Mokube and M. Adams, "Honeypots: Concepts Approaches and Challenges" ACMSE 2007 Winston -Salem North Carolina USA March 2007
[7] TOR, Third-generation Onion Routing, https://www.torproject.org/
[8] Modsec, Open Source Web Application Firewall http://www.modsecurity.org/