What are honeypots and how they can be used to protect the network and collect data about malware and attackers who lurk on the data?

This file also contains a description of a two-week experiment using honeypots to protect the test network. Which protocol was most often attacked, which was the most popular attack vector and how many ends of the botnet tried to get to the test network during this time?

Network protection by Honeypots?

How to understand that term? For sure not literally. Honeypot itself does not protect anything. I would even say, that it even attacks itself, because how to name attracting malware to the system if not self destruction? The protection can be provided only when a honeypot is a part of a system and it is well implemented.

So, what a honeypot is?

A honeypot is in general a computing resource, whose sole task is to be probed, attacked, compromised, used or accessed in any other unauthorised way. The resource could be essentially of any type: a service, an application, a system or set of systems or simply just a piece of information/data. The key assumption is that any entity connecting to or attempting to use this resource in any way is by definition suspicious. All activity between honeypot and any entity (assumed to be an adversary) interacting with it is monitored and analysed in order to detect and confirm attempts of unauthorised usage (in particular: malicious or abusive activity). A honeypot should mimic a production resource in its behaviour as accurately as possible – from an attacker's point of view there should be no noticeable difference between a honeypot resource and a production one. Resources represented by honeypots are non production. Moreover, those resources should be isolated from any production environment. No legitimate traffic should reach the honeypot, but this rule does not necessarily apply to client honeypots.

Honeypots can be used for many different purposes, for instance for the monitoring of Internet background noise (scanning activity of worms or bots), learning about compromised nodes, identifying new exploits and vulnerabilities, capturing new malware, studying hacker behaviour, looking for internal infections or attacks from insiders, etc. Naturally, the purpose of deployment impacts both the honeypot technology selection and the way it will be deployed.

Types of honeypots

1. Previous honeypot taxonomies

One of the best known honeypot taxonomies is that created by Christian Seifert, Ian Welch and Peter Komisarczuk. The authors defined six classes. The classes are defined within a flat relationship model instead of a hierarchical one, with no subclasses. This taxonomy is presented below and contains classes (marked in bold) with possible values (marked in bold + italic):

▪ **Interaction level** – describes whether the resource is limited in the way it exposes its functionality. This criterion is very similar to the previously mentioned level of interaction. Possible values are:

- ***Low*** – exposed functionality is somehow limited,
- ***High*** – exposed functionality is not limited in any way.

▪ **Data Capture** – describes the type of data a tool is able to capture from an attack point of view. Possible values (one tool can have multiple values assigned) are:
- ***Events*** – tool collects data about changes in state,
- ***Attacks*** – tool collects malicious activity (security policy violation attempt),
- ***Intrusions*** – tool collects malicious activity that leads to a security failure (cracking) i.e. system compromise or infection,
- ***None*** – tool does not collect events, attacks, or intrusions.

▪ **Containment** – describes measures a tool takes to defend against/constrain malicious activity spreading from itself. Possible values (one tool can have multiple values assigned) are:
- ***Block*** – malicious activity is identified and blocked (attack never reaches the target),
- ***Defuse*** – malicious activity is permitted, but is defused (attack reaches the target, but is manipulated in a way so that it fails),
- ***Slow Down*** – malicious activity is slowed down,
- ***None*** – no action is taken to limit the malicious activity.

▪ **Distribution Appearance** – describes whether the honeypot system appears to be confined to one system or multiple systems (from an attack point of view). Possible values are:
- ***Distributed*** – honeypot is or appears to be composed of multiple systems,
- ***Stand-Alone*** – honeypot is or appears to be one system.

▪ **Communication Interface** – describes interfaces one can use to interact directly with the honeypot. Possible values are:
- ***Network Interface*** – the tool can be directly communicated with via a network interface,
- ***Non-Network Hardware Interface*** – the tool can be directly communicated with via a hardware interface other than a network interface (i.e. USB),
- ***Software API*** – the tool can be communicated with via software API,

▪ **Role in Multi-tier Architecture** – describes in what role the honeypot acts within a multi-tier architecture. This class is very similar to the previously mentioned type of attacked resources. Possible values are:
- ***Server*** – the tool is acting as a server,
- ***Client*** – the tool is acting as a client application.

The taxonomy presented above was created because no sufficient taxonomy had existed at that time. It presents a well-researched work. However, it is quite complex and academic, and has irrelevant classes from our more practically oriented point of view.
Niels Provos and Thorsten Holz in their book Virtual Honeypots: From Botnet Tracking to Intrusion Detection presented a simple and elementary classification schema. Honeypots are divided into low- and high-interaction and distinguished between physical and virtual honeypots. The first pair of values is similar to corresponding ones in previously described taxonomies. The second pair constitutes a new class with two values:
● Physical honeypot – describes a real machine on the network,
● Virtual honeypot – describes resources simulated by another machine.

A physical honeypot can be fully compromised, so often this tool implies high interaction.
Later authors extend the interaction-based class with a third value: hybrid systems. As in the basic taxonomy, hybrid systems combine both low-interaction and high-interaction tools in order to gain the advantages of both.
Another extension defines client honeypots (similarly as in our basic taxonomy, this tools deal with client application threats). The authors had assumed that the term honeypot is

originally synonymous with server-side honeypot.
In summary, Provos and Holz defined a very similar taxonomy to the basic one described earlier in this chapter and extended it with the concept of physical and virtual honeypots, somewhat similar to our distinction between high- and low-interaction ones.

2. Basic taxonomy

Honeypots may be classified based on two fundamental and independent criteria (classes): type of attacked resources, and level of interaction. This taxonomy is very basic and fits all other (more complex) honeypot taxonomies.
First criterion (class) – type of attacked resources – describes whether a honeypot's resources are exploited in server- or client-mode. A server-side honeypot utilises network services such as SSH or NetBIOS, listening on their standard ports and monitoring any connections initiated by remote clients. In contrast, a client-side honeypot will employ a set of client applications, such as a web browser, that connect to remote services and monitor all generated activity.
The second criterion (class) – level of interaction – determines if the honeypot is a real resource (high-interaction) or only an emulated one (low-interaction). A mixed type of honeypot which combines both functionalities is called a hybrid honeypot.

2.1 Server-side honeypots

Honeypots designed to detect and study attacks on network services are called server-side. Honeypots of this type act as a server – they expose an open port, multiple ports or whole applications and listen passively for incoming connections, established by remote (likely malicious) clients. Often these types of honeypots detect threats which use scanning as means of identifying potential victims to compromise – for instance scanning worms or bots – but they can also be used to detect manual attempts to break into machines. Server-side honeypots are considered to be the 'traditional' honeypots, and often the term 'honeypots' is by default associated with them.

2.2 Client-side honeypots

Honeypots designed to detect attacks on client applications are called client-side honeypots, often honeyclients for short. A client application is a piece of software that establishes a connection to a server and interacts with it. The most popular and the most targeted type of client-side applications are web browsers, together with associated extensions and plugins.
Client-side honeypots are very different in their operation from server-side ones. Honeyclients actively establish connections to services in order to detect malicious behaviour of either the server or the content it serves. The most popular honeyclients are those detecting attacks on web browsers and their plugins, propagated via web pages. Some also have the capability to look at various forms of attachments, and there have been attempts to create instant message honeypots as well.

2.3 Low-interaction honeypots

Low-interaction honeypots are tools that operate by emulating their resources: services (in case of server-side honeypots) or client applications (in case of honeyclients). Emulation in this context means that the resources mimicked by a honeypot resource are limited in their functionality when compared to real ones. Interaction with an attacker is limited to some degree by the accuracy of emulation. Naturally, resources of a honeypot should be as

similar to their real equivalents as possible. This degree of accuracy greatly affects the interaction process between the honeypot and the attacker. Insufficient accuracy may cause attacks to terminate early, even before the actual malicious actions take place. It also makes the honeypot much easier to detect.
The main advantage of low-interaction honeypots is that they tend to be easier to deploy and maintain. The user has full control over the attack and the infection process. It is then possible to determine the current stage of an attack, which constitutes valuable information. Emulation also reduces the risk of the system becoming compromised. On the other hand, low-interaction honeypots have some disadvantages. An inherent weakness is their low accuracy of emulation. In specific cases emulated resources tend to behave in a different way than real ones, no matter how thorough an attempt was made by the creators. This could cause the attack or infection to terminate before its final phase, or the honeypot to be detected. Another issue is the fact that it is impossible to emulate not-yet-known vulnerabilities (so called 0-day vulnerabilities). All activity, especially in early stages of the attacks, must be coded into a honeypot's logic as an attacker tool expects a specific sequence of actions.

2.4 High-interaction honeypots

High-interaction honeypots are tools that provide real operating systems and resources (client applications or services). Note that the fact that 'real' systems and resources are utilised means they are not emulated. However, it is possible to use a virtual environment for such purposes, and it is in fact a common practice. In this concept, scenarios of interaction with the attacker are virtually unlimited, so a compromise or infection process should be fully completed in all cases.
Real behaviour of both the operating system and resources during the attack is the main advantage of high-interaction honeypots. This type of honeypot is able to detect attacks on 0-day vulnerabilities. Still, detection scope is limited only to specific (versions of) applications installed in the honeypot environment; an attack targeting an application in a particular version does not necessarily affect the same application in other versions, whether previous ones or newer.
The amount of data collected by high-interaction honeypots can be extensive and richer than from low-interaction tools. On the other hand, due to the complexity of the honeypot environment, there are problems in determining which elements of system/application behaviour are suspicious or malicious, and which are benign. For example: it may be not clear which read/write operations performed on the memory or disk are legitimate, and which ones are symptoms of exploitation.
Another disadvantage of high-interaction honeypots is limited control of the attack steps. The risk of compromising real systems, and losing control of the honeypot as a consequence, is higher than with the low-interaction counterpart. Another issue is that high-interaction honeypots require more resources compared to low-interaction ones, due to their complexity. This affects scalability and performance. Furthermore, deployment and usage of high-interaction honeypots, including their configuration and management, requires significant effort.

2.5 Hybrid honeypots

Hybrid honeypots combine both low-interaction and high-interaction tools in order to gain the benefits of both. Three well-known hybrid tools are described in this document: server-side (SurfCERT IDS, SGNET) and client-side (HoneySpider Network).
In SGNET, a high-interaction server-side honeypot is used to learn how to handle unknown traffic, e.g. how to emulate new protocols. After this learning process, further similar traffic

is redirected to low-interaction server-side honeypots. This combination increases both threat detection level and performance. SurfCERT IDS utilises multiple low-interaction server honeypots and Argos, a high-interaction solution. Similarly in HoneySpider Network a low-interaction honeyclient filters out benign websites, while all others (suspicious or malicious) are analysed again – this time with high-interaction honeyclients.

To get back to the "protection"

Regardless of the size of companies, the problem of cyber crime and cyber threats affects everyone. In identifying the attack areas, honeypots, or traps, are designed to detect attempts to unauthorized data acquisition by pretending to be a single service, system or entire local network. Depending on the application, we distinguish low and highly interactive honeypots. Both types of traps can operate in the area of the client or server and use multiple IP addresses, including even monitor and record the actions of the attacker.

From the perspective of security professionals, honeypots are particularly useful. It often happens that an attacker must first identify a service that can be run on non-standard ports, so he must learn about the internal structure of the network and try to attack it somehow. Some honeypots, especially those highly interactive, are difficult to configure and correlate with SIEM (Security Information and Event Management) systems, but are great as a "decoy", i.e. a tool for early warning against malware, new exploits or security vulnerabilities. They are also an excellent case study for researchers and administrators who can learn about the attacker's tactics and the most frequently exploited areas of the company's network on the "living organism".

Honeypots are intended for both operator networks as well as small and medium enterprises. They are a solid supplement to protective solutions, products that correlate information from endpoints and devices that secure the entire IT environment at the interface with the Internet. Knowing attack vectors is a very valuable skill. Thanks to it, the IT team will have a better insight into the quality of implemented security measures and even before unauthorized changes in production systems.

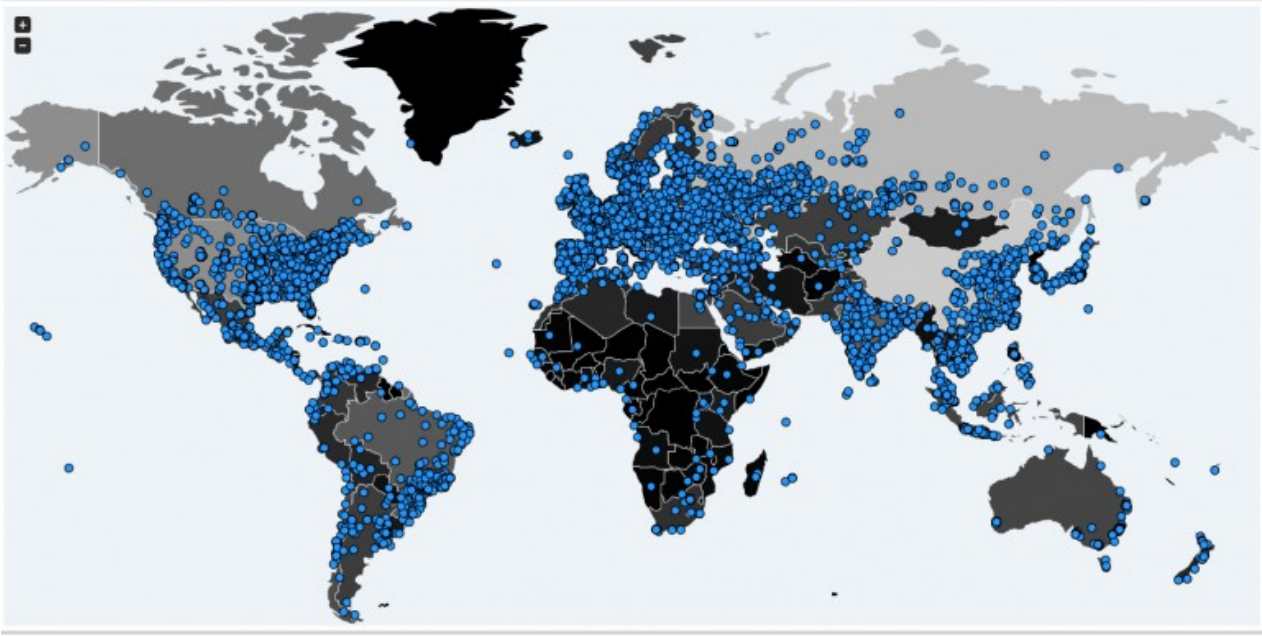
An experiment using honeypots and WatchGuard

I promised the example, so it is time to get back to it. WatchGuard is a producer of NGFW (Next-Generation Firewall) / UTM (Unified threat Management) devices, regularly classified as a leader and visionary in Gartner's reports, already having over one million implemented solutions in the SMB (small and medium-size business) and Enterprise sectors worldwide. What's interesting is that WatchGuard uses two types of honeypots in its solutions. One of them is located on every workstation, protected against encrypting files as part of the Host Ransomware Prevention. HRP is supported by the APT (Advanced Persistent Threat) Blocker module, which works in proxy mode for HTTP and SMTP protocols, blocking communication of the workstation with malicious servers. This works in such a way that the light sensor available for Windows systems creates hidden folders and files that after being deleted or modified by the malware are restored to the state before the infection. Thus, HRP, by monitoring a number of characteristic indicators defining malware ransomware, can stop the attack before encrypting files.

The second type of honeypots used by WatchGuard are "traditional" traps, which are part of the honeypots' outer farm. These traps play a very important role in collecting information about 0-day threats and blocking communication with C&C (Control and

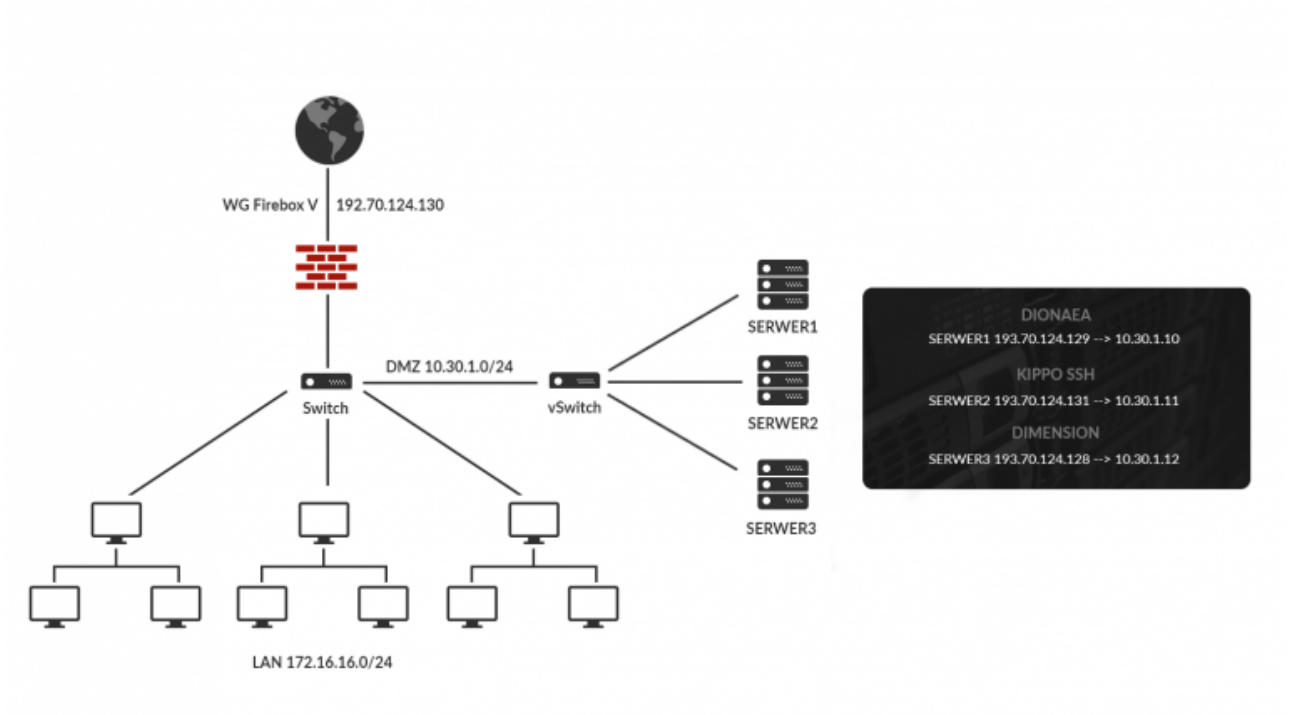
Command) servers. In addition, the TDR (Threat Detection and Response) component plays a key role in the protection of workstations - which analyses data using the proprietary behavioural algorithm and assigns a comprehensive assessment and rank (importance) to the threat. This powerful mechanism of correlation of information between the WatchGuard Firebox series devices and the manufacturer's cloud allows better protection of enterprises against unknown, malicious software and malicious IP hosts that make up the malware delivery chain.

The domino effect of the WannaCry ransomware.
(Source: https://intel.malwaretech.com/botnet/wcrypt)



One of the honeypots used for the 193.70.124.129 server is Dionaea. The lure puts a lot of emphasis on the implementation of the SMB protocol. The vulnerability of MS17-010 using DoublePulsar exploits is used on a large scale to spread Internet worms and cryptic malware: WannaCry, NotPetya, BadRabbit and others. Dionaea is a good source for obtaining malware for analysis: it can save binary files used by the attacker, emulate SMB / 445 protocol while supporting many other services, e.g. SIP (VOIP), HTTP, FTP, TFPT, MySQL and even SmartTV, IoT whether the XBOX console via the UPnP and MQTT protocols. It allows you to send files for analysis to external sites, such as VirusTotal (private API key is required), and also allows honeypot integration with the publicly accessible Sandbox Cuckoo sandbox - in the form of an online service or in the form of an implemented sandbox on the local network.

The following diagram shows the logical schema of simulation of servers with public services in the corporate network through two honeypots: Dionaea and Kippo SSH.

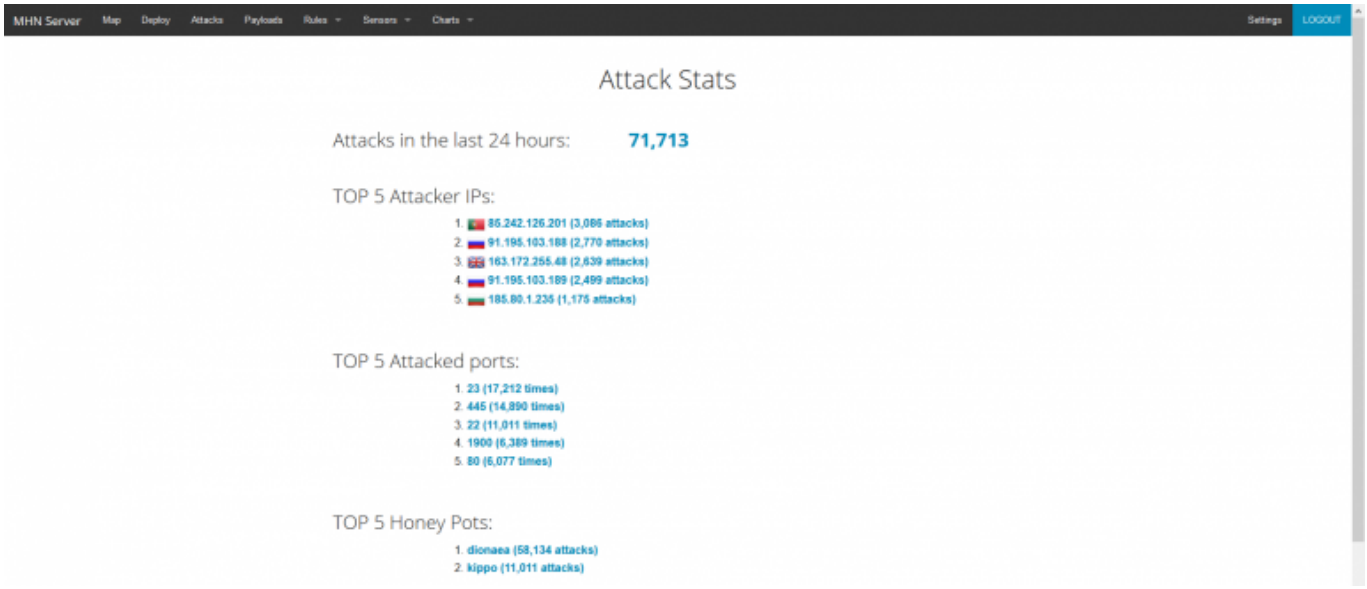


As mentioned, Dionaea emulates, inter alia SMBv1 protocol for Windows with MS17-010 vulnerability. Administrators and researchers can observe not only the tasks performed by the server, but also the process of running the code in the operating system - although not as detailed as during manual analysis.

The second honeypot for the 193.70.124.129 server is Kippo, which emulates the SSH service and stores information about dictionary attacks. Administrators, because of the practical use of Kippo SSH can strengthen system security. Kippo is equipped with a number of functions, including:

- emulating the start and end of SSH sessions,
- emulating *wget* and *curl* commands,
- storage of all downloaded files during an SSH session and launched by the attacker,
- creation of a binary file that stores session history that can be recreated for further analysis,
- allowance to substitute the attacker with any files and directories,
- emulates the file / etc / passwd.

The Modern Honeypot Network project was used to graphically consolidate information from honeypots. MHN Is completely free software, has basic API and can be integrated with SIEM systems.
MHN visualize inter alia the number of attacks in the last 24 hours.



MHN allows you to integrate dozens of the most popular honeypots, display the number of attacks per servers within 24 hours, indicate the most frequently attacked services and ports, collect information about malicious software used by attackers and much more. In its default configuration, MHN sends statistical data to Anomali Inc., which is responsible for the development of the project. It is worth remembering and disable this function if necessary.

The results of an experiment

Tests lasted about 2 weeks. The experience summary was divided into the results from before the implementation of WatchGuard FireboxV in the company network and those already after the implementation and configuration of UTM.

Observing incidents related to network security, the following conclusions were noted. Here are the most important of them:

- Five days after the implementation of the Dionaea trap on the Server1 sensor, 10472 attacks using different vulnerabilities to emulated services were observed. At the same time, 1762 attempts to login for the SSH service for the Server2 sensor were registered.
- As a result of attacks on Server1, but before the implementation of the WatchGuard FireboxV device in the corporate network, 27 malware samples were collected. Half of them were pests from the krypto-ransomware family, which were provided by a vulnerability in the SMB protocol with the use of the DoublePulsar exploit.

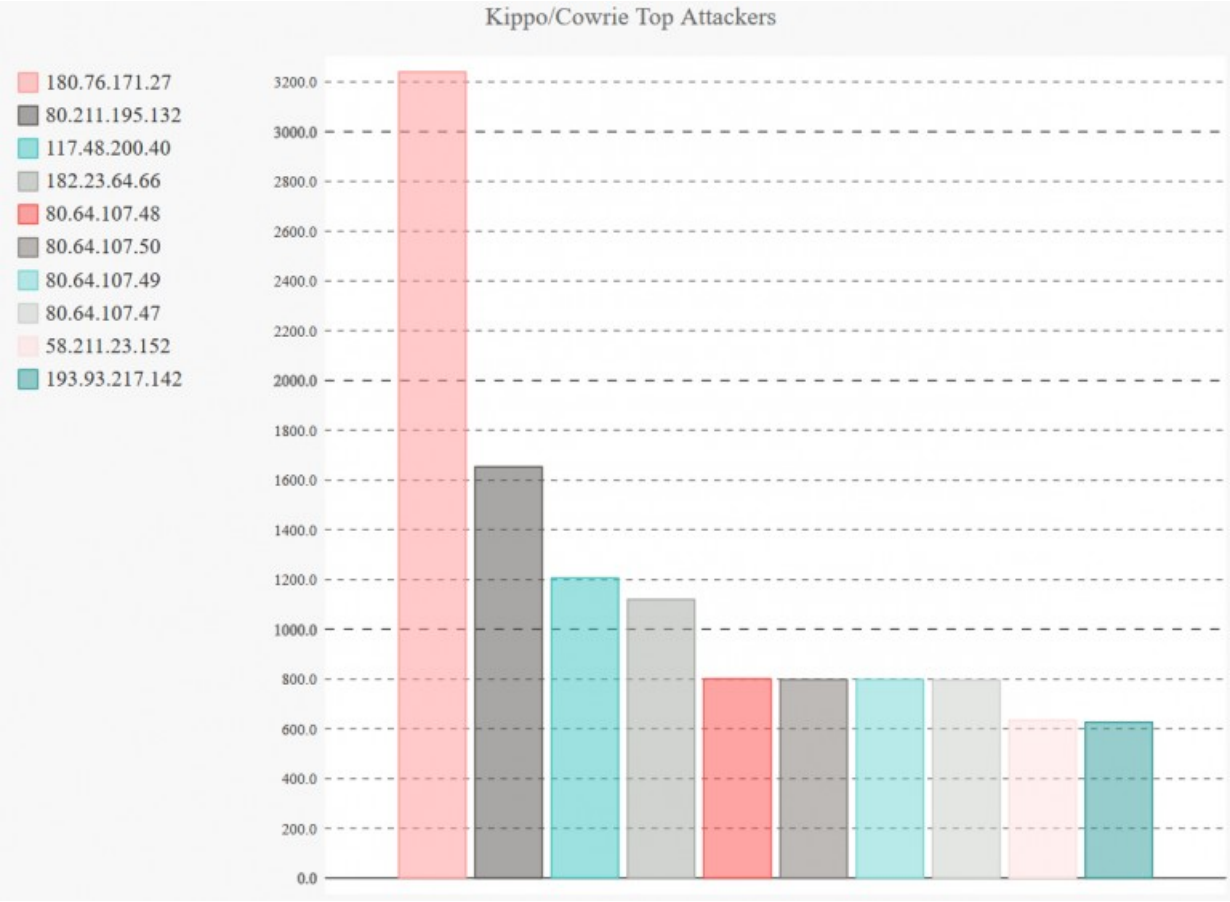
After 20 days from the implementation of the WatchGuard FireboxV device in the company network, no infections were detected on the Server1 sensor. Since the implementation of the WatchGuard FireboxV device in the corporate network, almost 2,400 attacks attempts with identified vulnerabilities have been blocked:

- 2354 attempts to use susceptibility in the SMB protocol to remotely execute the code.
- 14 attempts to overflow the buffer through a vulnerability in the HTTP service.
- 9 attempts to use vulnerabilities (CVE-2004-0113) that give the attacker an insight into the configuration of the HTTP server.
- 4 attempts to remotely execute code through vulnerabilities in other services.
- 4 attempts to use susceptibility to provide remote access to the shell.
- 3 attempts to use several vulnerabilities in OpenSSL that give a preview of the

information sent between the client and the server.

- 3 attempts to remotely execute code on the HTTP server.
- 2 attempts to use vulnerabilities that give an attacker the ability to run arbitrary code on a server with user privileges.
- 2 attempts to use the vulnerability in PHP configuration.
- 1 attempt to use the buffer overflow in Photodex ProShow Producer v5.0.3256.

Ip addresses from which login attempts were most often made.



System is a key to success

The case shows how powerful can a system based on a honeypot be. What really matters is implementation. From the theory and documentation to the implementation there is still a long way to through and it is not easy to be done well, but the example shows clearly, that it is worth an effort.