Annual ADFSL Conference on Digital Forensics, Security and Law

2015 Proceedings

May 20th, 2:00 PM

# Invited Paper - A Profile of Prolonged, Persistent SSH Attack on a Kippo Based Honeynet

Craig Valli
*Security Research Institute, Edith Cowan University*

Priya Rabadia
*Security Research Institute, Edith Cowan University*, sri@ecu.edu.au

Andrew Woodard
*Security Research Institute, Edith Cowan University*, sri@ecu.edu.au

Scholarly Commons Citation
Valli, Craig; Rabadia, Priya; and Woodard, Andrew, "Invited Paper - A Profile of Prolonged, Persistent SSH Attack on a Kippo Based Honeynet" (2015). *Annual ADFSL Conference on Digital Forensics, Security and Law*. 7.
https://commons.erau.edu/adfsl/2015/wednesday/7

# INVITED PAPER
# A Profile of Prolonged, Persistent SSH Attack on a Kippo Based Honeynet

Craig Valli, Priya Rabadia and Andrew Woodard
Security Research Institute
Edith Cowan University
sri@ecu.edu.au

## ABSTRACT

*This paper is an investigation focusing on activities detected by SSH honeypots that utilised kippo honeypot software. The honeypots were located across a variety of geographical locations and operational platforms. The honeynet has suffered prolonged, persistent and attack from a /24 network which appears to be of Chinese geographical origin. In addition to these attacks, other attackers have been successful in compromising real hosts in a wide range of other countries that were subsequently involved in attacking the honeypot machines in the honeynet.*

**Keywords:** Cyber Security, SSH, Secure Shell, Honeypots, Kippo

## INTRODUCTION

This paper is an investigation focusing on activities detected by Secure Shell (SSH) honeypots that utilise the *kippo* honeypot software (desaster, 2015). This paper is part of an ongoing investigation, with initial work conducted in 2012 and 2013 (Valli, 2012; Valli, Rabadia, & Woodward, 2013). All SSH honeypots were configured identically using *kippo* source code.

The focus of this particular research is primarily to identify evidence of automated attacks using password wordlists being implemented to login and gain access to three *kippo* SSH honeypots. All honeypots have the same username and password databases that contain multiple valid login password combinations. These valid combinations are part of the deception that is presented to the attacking entity by the *kippo* SSH honeypot. The passwords in these lists are drawn from well-known weak password lists. The honeypots are configured in *kippo* to present as different hostnames. The machines are further differentiated by manipulating some of the files in the fake filesystem used by *kippo*.

This paper examined a specific attack that has propagated since November 2014 and continues as of the time of writing. What is unique about the attack is that all previous attempts to attack the honeypots were detected as originating from UNIX based systems utilising SSH clients. The SSH attacks are now appearing to be coming from machines that utilise the PUTTY SSH suite of tools on Windows platform operating systems. Furthermore, the volume of SSH login attempts evinced on the honeynet in the past four months has increased at a rate of growth which is approaching that of exponential. This significant increase in attempts is likely due to Windows operating system based computers comprising a significant share of the market, and reportedly in excess of 97% in China (Popa, 2015).

## OVERVIEW OF THE SETUP OF THE *KIPPO* SSH HONEYNET

A honeynet can readily be described as a controlled and centrally administered collection of honeypots. The *kippo* SSH honeypot is a medium interaction honeypot, meaning that the honeypot imitates some functions that would exhibited by a live system (Hosting, 2013; Stevens & Pohl, 2004). The *kippo* honeypot is designed to effectively mimic an SSH server to an attacking entity. The SSH protocol is designed to securely transmit data using a point to point encryption tunnel (Ciampa, 2010), provides high grade encryption, and is a secure replacement for plaintext terminal programs such as telnet or rsh on UNIX or UNIX-like operating systems (Linux, OpenBSD, FreeBSD). Most network connected UNIX or UNIX-like operating systems have SSH installed as a client, and it is often included as a server (daemon) to help protect systems by providing a platform for encrypted communications. There are also many SSH clients available to run from Windows operating system based computers, with Putty being a commonly used Windows client (Tatham, 2015).

*kippo* honeypots are designed to collect data from attacker interaction with an emulated SSH service. The emulated SSH service is provided by an open-source, Python based event-driven program called Twisted (TwistedMatrixLabs, 2013). Twisted provides the libraries that are utilised and deployed by *kippo* to imitate a valid encrypted SSH session to an entity. Relevant SSH artefacts are also extracted, including the SSH banner or string that the daemon or clients presents to connecting SSH entities. Each of these banners or strings are typically unique and in many cases can reliably fingerprint the connecting operating system and device. Fingerprinting is a term used in network security to describe the data which is sent by a computer when it is connect to over a network, and this data is considered to be unique to each operating system, and in some cases different versions of a given operating system. *Kippo* allows the honeypot user to change the SSH banner to any known valid fingerprints for SSH.

The honeypot also emulates a logically correct, but manufactured file system to present to the user who successfully gains access to the honeypot. The system also presents falsified system reporting and allows interaction with artefacts such as /proc/cpuinfo or .bash_history logfile. While the level of deception in the default setting is limited, this functionality is able to be expanded and modified at will. For this experiment, key elements were modified such as /proc entries and different bash entries to create a difference in each of the *kippo* hosts presented in the honeypots.

The *kippo* SSH honeypots are written in Python, and installed using the recommended process. Source code was obtained from https://github.com/ikoniaris/*kippo* which is modified *kippo* code. The setup for these particular systems used in the data collection was conducted as specified by the BruteForce Lab Guide (Labs, 2011) and further enhanced to send data to various database stores Postgresql, Mysql and also an ElasticSearch server. This setup deviates from the original *kippo* SSH documentation in that it uses the authbind daemon instead of twistd as the initial connecting daemon for the service. This configuration lets authbind handle the binding of the twistd as a non-root user to a low numbered TCP port and passes this to the twistd daemon. This configuration was found to be more consistent, reliable and secure during the conduct of the research project.

During the installation process, a local MySQL database was configured and secured to record all the interactions with the *kippo* honeypots. Figure 1 is a table from (Valli, 2012) which was sourced from the *kippo* documentation. It shows the MySQL database structure used in the *kippo* honeypots that was used to record all the interaction data.

| | |
|---|---|
| TABLE auth<br>id int(11) PK,<br>session char(32) NOT NULL,<br>success tinyint(1) NOT NULL,<br>username varchar(100) NOT NULL,<br>password varchar(100) NOT NULL,<br>timestamp datetime NOT NULL, | TABLE input<br>id int(11)NOT NULL PK<br>session char(32) NOT NULL,<br>timestamp datetime NOT NULL,<br>realm varchar(50) default NULL,<br>success tinyint(1) default NULL,<br>input text NOT NULL,<br>KEY session (session,timestamp,realm) |
| TABLE clients<br>id int(4) PK<br>version varchar(50) NOT NULL | TABLE sensors<br>id int(11) NOT NULL (PK)<br>ip varchar(15) NOT NULL |
| TABLE sessions<br>id char(32) NOT NULL PK<br>starttime datetime NOT NULL,<br>endtime datetime default NULL,<br>sensor int(4) NOT NULL,<br>ip varchar(15) NOT NULL default '',<br>termsize varchar(7) default NULL,<br>client int(4) default NULL,<br>KEY starttime (starttime,sensor) | TABLE ttylog<br>id int(11) NOT NULL PK<br>session char(32) NOT NULL<br>ttylog mediumblob NOT NULL |

| | |
|---|---|
| | |

*Figure 1 - MySQL database structure for kippo honeypot*

After recording to the local MySQL database, these data are then transmitted to a centralised Postgresql SQL server (Valli et al., 2013). Communication is achieved using a Python extension that uses a Postgresql driver to connect to the SURFIDS system IDS logging server (IDS, 2013). The centralised logging server utilises the SURFIDS system for storing the data from the honeypots into an aggregated PostgreSQL database. The database has functions and tables specifically for the *kippo* honeypots data. In addition, the honeypots running *kippo* operate Dionaea and Glastopf, which in turn report to the SURFIDS instance. It should be noted that these data are not used in this analysis or reported here.

The entire honeynet had its *kippo* code modified to support transmission of attack data to an Elasticsearch instance. In addition to storing the data in local MySQL databases, this code allows the researchers to concurrently transmit it to an ElasticSearch engine ("Elasticsearch," 2015) that has a Kibana("Kibana," 2015) frontend engine. The data in this system can subsequently be queried using customised Kibana frontend queries. The utilisation of the Kibana frontend allows the user to create many custom views of the data, allowing for detection of anomaly and threat. Demonstrative figures extracted from Kibana are included later in this paper.

## GAINING ACCESS

To gain access to these honeypot systems, the correct username and password must be entered at the emulated login screen, as would be the case for a real system. While general user accounts on well administered systems may have lockout of the account for unsuccessful attempts, it is not a feature that is enabled on administrative and root accounts at any time. The reason being that repeated deliberate unsuccessful login attempts can result in a denial of service, thereby locking out access of administrative or root accounts. The lack of an account lockout for unsuccessful password attempts is the Achilles heel of availability for administrative accounts or system accounts, and can be routinely exploited by the use of automated attack tools. The generic tool used for this type of activity is called colloquially a password cracker.

Passwords crackers can be deployed to identify the correct password by trying different passwords against the particular service or system. It should be noted that the rate of password attempts is reaching billions of passwords a second when using multi CPU- or GPU-enabled password crackers, with the limiting factor being that of the target machine speed in terms of network or processing power.

There is a finite number of passwords for any given system password implementation, often referred to as a key space, and while finite, these key spaces can be computationally large. For example, the standard Windows LM password key space for all possible passwords is $2^{43}$. While it is relatively infeasible for a single conventional computer to derive these passwords in a timely fashion, this does not hold true for advanced techniques using compute clustering or GPU technology that can factor these passwords at the rate of billions per second. Furthermore, techniques such as pre-computed rainbow tables (Oechslin, 2003) can greatly increase speed, as the key space is computed once and each possible password stored as a hash within a database table or binary file structure for easy reuse. The limiting factor then becomes the speed at which the password hash can be compared against every entry in the rainbow table database.

Passwords are typically stored in file structures as a cryptographic hash or set length ciphered text, and not as plaintext. Without the use of hashing and cryptography, compromise of the password is trivial. Compromise is achieved by simply opening the file that contains the password and reading it. To increase the security of passwords, they are usually protected by applying a cryptographic process to the password, with the resulting output referred to as a hash.. In this format, the probability of an attacker obtaining or guessing the password on a first guess is very low. The MD5 hash algorithm is a common method employed to achieve password obfuscation in this manner (Marechal, 2008).

There are different techniques that can be used to break or crack passwords. A brute force attack uses a systematic method of guessing the password by enumerating through a combination of characters, symbols and numbers of the allowable characters. A dictionary attack creates hashes of words that appear in a dictionary, and compares them to the stored password or feeds the hash as input to the login mechanism of a live system. The former method is commonly referred to as an offline attack, and the later as a live attack. Rainbow tables are databases comprised of various character combinations that have been pre-computed and stored typically in an efficient binary structure, allowing fast retrieval. Password techniques that utilise plaintext wordlists can also be deployed. These types of attack tend to utilise social engineering techniques and deductive reasoning to pick viable candidate passwords. In some cases, these are provided as defaults with the security software distribution or attack utilities used in, for example, Kali. Evidence from the (Rabadia & Valli, 2014) paper proves use of these password lists by attackers. *Kippo* facilitates the use of these default passwords to produce a list of acceptable passwords.

## ATTACKER BEHAVIOUR POST-COMPROMISE

After achieving login on an account, an attacker will typically want to have administrative control of the device, also referred to as "owning" the system. The attacker then typically downloads malicious code and executes it, compromising the machine with infected binaries or privilege escalations that allow for remote administrative access of the machine. Achieving remote access allows provides persistent access and allows the cyber-criminal to use the compute device for their own activities at will.

By design, the *kippo* honeypot allows all of this malicious activity to occur i.e. if the attacker logs in they are able to interact with a fake shell and download files to honeypot. The files are downloaded using *wget* functionality and stored in a sandbox for later retrieval and examination by the honeypot operator.

Apart from logging and recording the shell interactions, as attack activity occurs *kippo* also extracts other relevant artefacts from the sessions with the attacker. As mentioned previously, one such artefact is the presented SSH signature from the session that can be used to identify the attacking entity by its digital fingerprint. This fingerprint information was instrumental in detecting a significant change in SSH malicious activity since this research commenced in late 2011.

In addition to the *kippo* honeypot software, all of the honeypot systems use *p0f*, a passive operating system fingerprinting tool (Zalewski, 2015). This program works by looking at the TCP transmission and TCP/IP stack responses, and tries to determine the attacker's operating system through fingerprinting and signature matching. A commonly used offensive tool *nmap* works on similar principles of operation. The major difference is that the p0f program does so passively, while nmap is proactive and sends packets to the target.

**The story so far**

The *kippo* honeynet in this research had been in existence since early 2011 and has expanded with the addition of new sensors. There are now 22 sensors in total which are spread physically around the globe. There are VPS servers located in USA, Germany, Netherlands, Singapore, Australia and the Philippines, and as previously mentioned these are all installed on a maintained Ubuntu LTS (Long Term Support) platform which is currently Ubuntu 14.04 LTS. In addition to VPS assets, there are ADSL based honeypots deployed in Australia. These utilise Raspberry Pi implementations as well as i686 based Ubuntu servers that have identical configuration to the VPS servers.

The project detected a wide range of SSH fingerprint signatures as shown in Table 1 prior to 12th November 2014, totalling approximately 1.2 million interactions, increasing to 18.6 million interactions by 5th Mar 2015 (Table 2). The attackers that had connected to the honeypots prior to the 12th November 2014 had predominantly been Unix/Unix-Like signatures as shown in Table 1, with a predominance of the Kali and BackTrack Linux distributions representing 99% of all malicious login attempts on the honeypots using the libssh2 libraries.

**Table 1 – Top 10 SSH Signatures detected by honeypots**

| | | |
|---|---|---|
| 1 | SSH-2.0-libssh2_1.4.2 | 825729 |
| 2 | SSH-2.0-libssh2_1.4.3 | 342920 |
| 3 | SSH-2.0-libssh2_1.4.1 | 7101 |
| 4 | SSH-2.0-JSCH-0.1.51 | 4390 |
| 5 | SSH-2.0-libssh2_1.4.0 | 2230 |

| | | |
|---|---|---|
| 6 | SSH-2.0-OpenSSH_5.2 | 1530 |
| 7 | SSH-2.0-paramiko_1.8.1 | 1157 |
| 8 | SSH-2.0-libssh2_1.0 | 843 |
| 9 | SSH-2.0-OpenSSH_6.0p1 Debian- | 322 |
| 10 | SSH-2.0-libssh2_1.4.3 PHP | 134 |
| | Total | 1186356 |

**Table 2 - Top 10 SSH Signatures until 05/03/2015**

| | | |
|---|---|---|
| 1 | SSH-2.0-PUTTY | 12477973 |
| 2 | SSH-2.0-libssh2_1.4.2 | 3536116 |
| 3 | SSH-2.0-libssh2_1.4.3 | 1853226 |
| 4 | SSH-2.0-libssh-0.1 | 310530 |
| 5 | SSH-2.0-libssh2_1.4.1 | 225762 |
| 6 | SSH-2.0-JSCH-0.1.51 | 65791 |
| 7 | SSH-2.0-PuTTY_Release_0.63 | 51646 |
| 8 | SSH-2.0-libssh-0.4.8 | 37160 |
| 9 | SSH-2.0-libssh2_1.4.0 | 9131 |
| 10 | SSH-2.0-JSCH-0.1.44 | 6472 |
| | Total | 18573807 |

As of March 5th (Table 2) these Linux signatures only represented 30.2% of all malicious login attempts. At that point in time, the dominate signature was that of SSH-2.0-PUTTY, which represented 67.1% of all attempts.

It should be noted that the SSH-2.0-PUTTY signature had not been previously seen on the honeynets prior to 27th October, 2014, when there was an observation of 10 connections in a relatively short period of time. The next significant event was on the 13th November where 69 attempts were recorded. A significant increase in the use of the tool commenced on the 22nd November, where 13,788 attempts were made from a /24 network. This /24 was not initially able to be identified on IP based geolocation databases, but it is now identified as apparently originating from China. Initial traceroute reconnaissance by the researcher also indicated that the traffic was propagating from Chinese mainland assets. The other interesting part to note about the traffic was that prior to 12th November there was less than 20 contacts in total from that /24 IP address space over the entire period of operation of the honeynet. A histogram of all attacks with the signature SSH-2.0-PUTTY is show in Figure 1

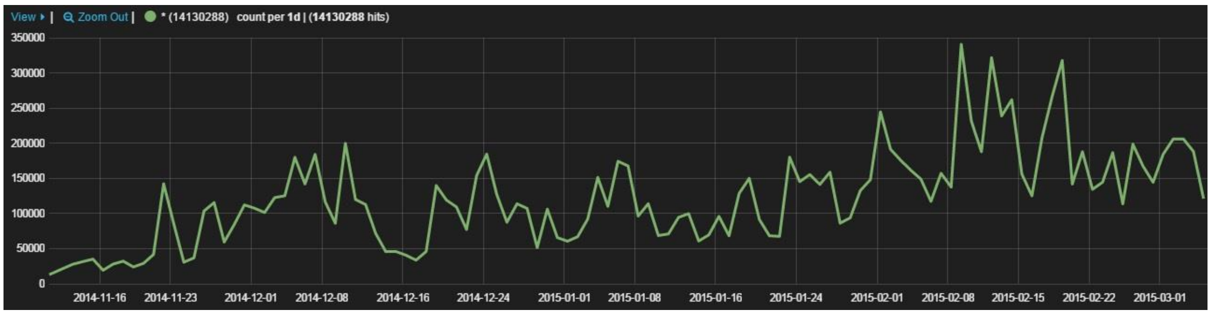

**Figure 1 – Histogram of login attempts against all honeypot sensors where the SSH-2.0-PUTTY client was used**

All IPs within the particular /24 still have daily contact with the honeynet, with the total number of attempts ranging from 0 ~ 50,000 on any given day from single IPs in that network address range. Figure 1 is the histogram for that time period, and represents 14130288 logins with only 42236 successful or 0.30% success.

The attack "network" has grown significantly as the attackers have compromised machines globally. Initial contact with the honeypots with the **SSH-2.0-PUTTY client** signature was restricted to the same /24, but as they successfully compromised machines they in turn started to contact the honeynet nodes. Contact from nodes other than the /24 numbered 210,159 and the following figure (Figure 2) shows the geographic spread of these contacts.

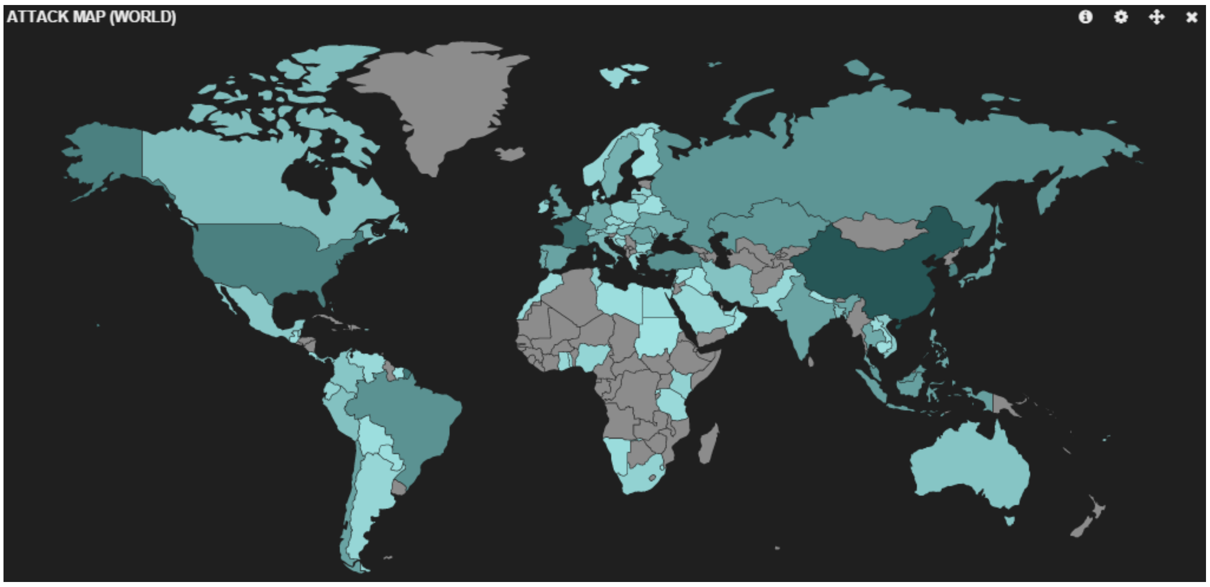

**Figure 2 – Geographic spread of new attackers attacking the honeypot sensor network**

The top 5 countries identified as attacking the honeynet are 73341, France 25643, USA 15416, Turkey 7509 and Brazil 6472. This is reflected pictorially in decreasing shades of green i.e largest = darker shade. It should be noted that while these are attackers that are seen by the 22 nodes in the honeynet, this is not an exhaustive mechanism. However, given that modus operandi of the repeated multiple attempts from the new members of the attack network is consistent with the "original" /24, it seems likely that it is the same. The bruteforce nature of the attempts indicate automated bruteforce retries of logins. When login was achieved the packet captures also evince high repetitious reuse of the same script or code signatures to attack the systems once compromised.

## DISCUSSION AND CONCLUSION

These attacks are ongoing and persistent now for over four months, and appear to be increasing in magnitude over time. The attack would appear to be relatively non-sophisticated, repetitive, verbose and inefficient.

From analysis of the collected data it would appear that the attacking entities are not sharing attack data, and the attacks are noisy and not as efficient and optimised. One possible explanation for this is that the honeypots are not responding back to or potentially providing “alive” tokens to the attacking entity, as we are not running the malcode they download. This lack of response by the honeypots could be the causation of the retries by the attackers. This aside, the logic employed would appear to be:”if the compromise of the box was successful i.e we were able to login and successfully download the malfeasant code, then leave alone.” However, the observed behaviour was:”if code has not deployed successfully because we do not have control, then, re-attempt installation”. This finding has implications for honeypot design sophistication and deployment, and is a valuable outcome in of itself. To prevent this behaviour, a method for sending “false positives” back to the attacking entity mimicking command and control would need to be developed.

This pattern of reattempted compromise in this occurrence is consistent with the intention of a honeypot, which is to exhaust or distract resources away from legitimate targets through deception. Every retried compromise and install represents resource usage by the attackers. This usage includes, but is not limited to, actual machine run time, consumption of network bandwidth and scanning activities, all of which consume finite resources on the part of the attacker. In addition to resource wastage the activity provides, with every attempt, more evidence of the actual attack and in most cases would represent repeat criminal offences.

Of interest is the observation more recently of the initial use and subsequent significant increase of attacks using the Putty SSH tool. Further, it was observed that a significant quantity of these attacks apparently originated in China. There may be a number of likely reasons as to why this was observed, but one hypothesis is that the attackers leveraged compromised Windows operating system computers in China as the initial attack platform. Data suggests that the majority of computers in China are running Windows, with most of these copies are pirated and largely unpatched and thus are insecure and susceptible to compromise themselves (Popa, 2015). Use of compromised computers as a third party attack platform is not uncommon, as it makes it harder to identify the true origin of a cyber-attack(Livadas, Walsh, Lapsley, & Strayer, 2006). This does call into question whether these attacks are truly originating in China, as has been suggested in previous honeypot research (Pouget & Dacier, 2004).

Further research is being conducted now on the downloaded payloads from the attacking entities. One of the features of the honeynet is that it will download, check the md5 sum of the file, and if it already exists will discard the download. This is advantageous in these cases as otherwise there would be significant storage implications for this research alone.

There is also data with respect to detected OS fingerprints for attacking entities which will be presented in further research papers.

Finally, the honeynet is functioning as it should, and this particular persistent attack has and continues to yield significant data for analysis and interpretation.

## REFERENCES

Ciampa, M. D. (2010). *Security Awareness: applying partical security in your world* (3rd ed.). Boston: Course Technology.

desaster. (2015). *kippo*. Retrieved from https://github.com/desaster/*kippo*

. Elasticsearch. (2015). https://www.elastic.co/products/elasticsearch: ElasticSearch BV.

Hosting, G. P. (2013). *Kippo*. *Kippo SSH Honeypot* Retrieved 09.10.2013, from http://code.google.com/p/*kippo*/

IDS, S. (2013). SURFcert IDS Retrieved 20/10/2013, from http://ids.surfnet.nl/wiki/doku.php

. Kibana (Version 3.1.2). (2015). https://www.elastic.co/products/kibana: Elasticsearch BV.

Labs, B. (2011). Installing *Kippo* SSH Honeypot on Ubuntu Retrieved 27.09.2013, from http://bruteforce.gr/installing-*kippo*-ssh-honeypot-on-ubuntu.html

Livadas, C., Walsh, R., Lapsley, D., & Strayer, W. T. (2006). *Usilng machine learning technliques to identify botnet traffic*. Paper presented at the Local Computer Networks, Proceedings 2006 31st IEEE Conference on.

Marechal, S. (2008). Advances in password cracking. *Journal in computer virology*, *4*(1), 73-81.

Oechslin, P. (2003). *Making a Faster Cryptanalytic Time-Memory Trade-Of*. Paper presented at the The 23rd Annual International Cryptology Conference, CRYPTO '03, Santa Barbara, California, USA.

Popa, B. (2015). More than 97 Percent of Computers in China Now Running Windows, Mostly Pirated Retrieved March 2015, 2015, from http://news.softpedia.com/news/97-Percent-of-Computers-in-China-Now-Running-Windows-Mostly-Pirated-472110.shtml

Pouget, F., & Dacier, M. (2004). *Honeypot-based forensics*. Paper presented at the AusCERT Asia Pacific Information Technology Security Conference.

Stevens, R., & Pohl, H. (2004). Honeypots und Honeynets. *Informatik-Spektrum*, *27*(3), 260-264. doi: 10.1007/s00287-004-0404-y

Tatham, S. (2015). PuTTY: A Free Telnet/SSH Client, from http://www.chiark.greenend.org.uk/~sgtatham/putty/

TwistedMatrixLabs. (2013). What is Twisted? Retrieved 23.09.2013, from http://twistedmatrix.com/trac/

Valli, C. (2012). *SSH: Somewhat Secure Host*. Paper presented at the Cycberspace Safety and Security, Melbourne Australia.

Valli, C., Rabadia, P., & Woodward, A. (2013). *Patterns and Patter - An Investigation into SSH Activity Using Kippo Honeypots*. Paper presented at the Australian Digital Forensics Conference, Edith Cowan University.

Zalewski, M. (2015). p0f v3 Retrieved March, 2015, from http://lcamtuf.coredump.cx/p0f3/