Building a Honeypot to Research Cyber-Attack Techniques
Interim report

Simon J. Bell

November 5, 2013

Abstract

The internet can be a dark and dangerous place; featuring viruses and cyber attacks. This project aims to uncover some of these threats and reveal just how vulnerable the internet can be. The project involves creating a honeypot - a device designed to attract cyber attackers - and to analyse cyber attacks to see what's going on in the dark underworld of the internet.

This interim report introduces some common cyber-attack techniques that are frequently used on the internet today. It then describes how a honeypot can be used to witness these attacks as they take place, and then analyse the attacks. This report also includes a breakdown of the project including: how the honeypot software will be built, professional considerations and a project time-scale.

1 Introduction

The overall aim for this project is to build an SSH honeypot to research cyber-attack techniques. This section provides an introduction to some common internet threats, what a honeypot is and why honeypots are useful at detecting cyber threats, what the SSH protocol is and what the aims for this project are.

The internet can be a dangerous place due to the many threats that exist within it. Most of the time we may be completely unaware of these threats since they're hidden and might not be immediately obvious. The internet is made up of a globally distributed network; it isn't run by one single organisation. Therefore the internet has no central governing body. This means that the internet has no global laws. What may be illegal in one country may not be illegal in another country. Another issues facing the internet is anonymity: many users of the internet believe that their actions online cannot be traced since they're "hidden" behind a computer. These reasons may lead some internet users to carry out actions which may be considered unethical or illegal (in some countries), such as cyber-attacks. As a result: the internet is littered with many cyber threats and cyber attacks which are carried out by these unethical internet users which we call *attackers*.

1.1 Common Cyber Threats

Some examples of common internet threats are explained below. For some of these examples it can be useful to represent the scenario with the fictitious characters Alice, Bob and Mallory. Where Alice and Bob are trying to communicate with each other and Mallory is trying to spy on or attack Alice and/or Bob. An example of a common form of attack is the man-in-the-middle attack [15]: this is when Alice thinks she's communicating with Bob. However, all communications are actually going through Mallory, therefore Mallory is able to eavesdrop on both Alice and Bob. This scenario is illustrated in figure 1.

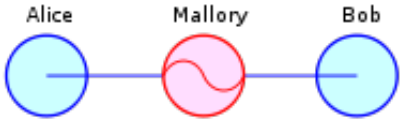

Figure 1: Man-in-the-middle attack[9].

Another common cyber-attack is the brute-force attack [23] [13]. In this scenario Alice has some data which only she and Bob should be able to access. So Alice encrypts the data. This means that the correct key (e.g. a password) is required to decrypt the data. In a brute-force attack Mallory will try repeatedly attempting various different keys until the correct key is found. Once Mallory finds the correct key she is then able to decrypt the data, thereby viewing its contents. A common strategy used in the brute-force attack is to use a dictionary (containing many different words) in order to try various different combinations of words, numbers and symbols. The brute-force attack is illustrated in figure 2 whereby it can be thought of as various keys being tried on a single lock in an attempt to find the correct key to gain authorisation.

The final common cyber-attack example is the distributed denial of service (DDoS) attack [25]. In this type of attack the attacker's aim is to temporarily or indefinitely stop a service of some host (the victim) from operating. The result is that no user is able to use that service. A good example of this is a DDoS attack on a web server: the result would be that no users can browse the website if the DDos attack is successful. A common way of carrying out a DDoS attack

Figure 2: Brute-force attack[3].

is by using a network of compromised machines known as "bots". These bots act like zombies in that they can be given commands from handlers to carry out tasks such as flooding a web server. Figure 3 shows how an attacker can send her requests to the handlers and these in turn will command the zombies (infected machines) to carry out the command. The result is that the victim (e.g. a web server) becomes inundated with requests and may become flooded, therefore ceasing to serve any further requests.

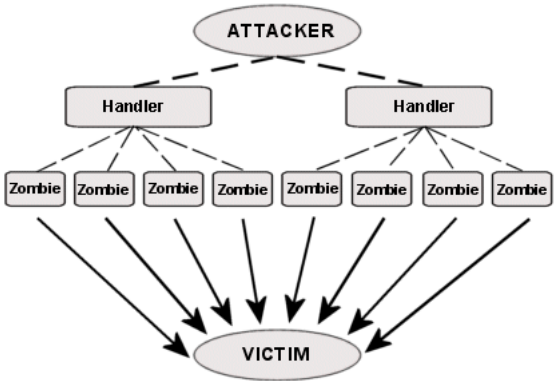

Figure 3: Distributed Denial of Service (DDoS) attack[6].

1.2 Honeypots

A good way to uncover some of the threats that exist on the internet is by using a honeypot [34] [24]. A honeypot is a device (or in this case a piece of software) designed to look like a regular computer connected to a network and usually appears to contain valuable information. However, the honeypot is actually isolated from the rest of the machine and monitors all activity. Honeypots can be classified into two deployment types: production and research. Honeypots can then be classified into three main design types: low-interaction, high-interaction and pure honeypots [39] [26].

Production honeypots are used mainly by corporations and organisations that have an existing network infrastructure in place and are looking to improve their current security. Production honeypots require little maintenance and only produce limited information which is enough to strengthen the network or detect vulnerabilities. The advantage of production honeypots is that they are easy to deploy but they produce limited information about attackers and attacks

Research honeypots are deployed with the goal to find out complex information about current attack techniques and the attackers behind them. These types of honeypots wouldn't add direct value to an organisation because the data they produce would need to be analysed. The data produced from research honeypots can be used to study current threats and to determine how organisations can protect themselves against these threats. The main advantage of research honeypots is the volume and complexity of data they produce. However, they can be complex to deploy and maintain.

Low-interaction honeypots focus on one specific service to emulate (e.g. email service, remote login service). The main advantage of low interaction honeypots is that they consume relatively little resources and are therefore easier to deploy than more high-interaction honeypots. The disadvantages to low-interaction honeypots is that they are limited to only detecting vulnerabilities for the service which the honeypots is emulating. Examples of low-interaction honeypots include Dionaea[5]: a honeypot that captures attack payloads and malware, Glastopf[30]: a honeypots that emulates a vulnerable web server and Honeyd[29]: a honeypot for capturing attacker activity.

High-interaction honeypots emulate multiple services at the same time (e.g. a web server and an email service). The main advantage of high-interaction honeypots is that an attacker may be more convinced that the honeypot is a real machine since there are many services to attack. However, high-interaction honeypots are more expensive to maintain and deploy. An example of a high-interaction honeypot is the Honeynet Project's 3rd Generation Honeywall ('roo') framework[17].

The third type of honeypot is the pure honeypot. This type of honeypot is a complete operating system whereby the monitoring of attackers' activities is recorded via a tap on the honeypot's link to the network. Pure honeypots don't require any special software to install since they're just regular systems. However, ensuring that these systems do not cause vulnerabilities on the network does require specialist knowledge.

1.3 Secure Shell (SSH) Protocol

One of the systems that can be emulated by a low/medium-interaction honeypot is the remote login service: secure shell (SSH) [14]. The SSH protocol is a secure communication channel which allows users to remotely control systems. The SSH protocol transmits data over the TCP protocol[28]. It can be used on both Unix-like operating systems (Linux) and Windows. There are three main authentication methods used in the SSH protocol: password: whereby users are required to type the correct username

and password to gain authentication, keys: whereby users are required to provide the correct cryptographic key to gain authentication and a hybrid of both password and keys to gain authentication.

One of the main vulnerabilities of the SSH protocol is the use of weak passwords [33] and lack of cryptographic keys [37]. This is a vulnerability because weak passwords can be broken fairly easily by using the brute-force attack (as described previously). Once an attacker gains entry into the SSH protocol any number of attacks may be carried out. One specific attack that could be prepared for is the DDos attack: having gained entry into the SSH protocol an attacker might transfer and execute malicious software onto the host machine. This would allow the attacker to carry out a DDoS attack by controlling the attacked host to attack the DDoS victim (as discussed previously).

Current examples of SSH honeypots include Kippo[7] and Kojoney[18], both of which use the Twisted Conch[20, p. 172] package, which is implemented in the programming language Python[35]. These honeypots provide a medium-interaction, production honeypot. The advantage of these SSH honeypots is that they're quick to deploy and provide useful data for analysing SSH attacks on an existing network.

The SSH protocol defines a strict procedure for establishing SSH sessions [38]. The vast majority of Linux operating systems use the library openSSH[10] for establishing SSH sessions, which is written in the programming language C[22]. In an attempt to create a honeypot that behaves similarly to the genuine openSSH library, this honeypot shall also be written in the programming language C. The library libssh[8] shall be used to create the SSH session, this ensures the connection conforms to the SSH RFC protocol[38]. The programming language C executes much faster than Python and it's important that this honeypot has similar timings to the genuine openSSH implementation. This is important because attackers are becoming aware of honeypots and are trying to detect them using various methods such as timings of the honeypot [21][36].

1.4 Project Aims

There are two main aims for this project: the first is to build an SSH honeypot and the second is to research cyber-attack techniques.

The first aim, building an SSH honeypot, will allow users interested in cyber-security (or information security) to easily and quickly deploy a medium-interaction SSH honeypot and be able to analyse the data produced from this to determine current threat levels. The specific objectives for building an SSH honeypot are:

- Implement medium-interaction, research honeypot in programming language C
- Log all username and password attempts
- Allow user authentication and log all attempted commands
- Allow most common set of shell commands to emulate (e.g. ls, wget, w, ...)
- Allow attackers to upload files to honeypot using wget
- Emulate running of uploaded files from attackers

The second aim, research cyber-attack techniques, will analyse current threats (with data produced from the deployment of created honeypot) and help those interested in information security and those wanting to tighten existing SSH systems. Specific objectives for researching cyber-attack techniques are:

- Deploy honeypot to public server so anyone in the world can attack it
- Produce data from honeypot deployment
- Analyse most commonly used username and passwords
- Analyse how username and password lists are created (dictionary, other lists etc)
- Analyse most commonly attempted shell commands
- Analyse uploaded files from attackers
- Analyse how host (honeypot) is used by attackers once compromised

These aims and objectives are achievable in the given time frame. The project plan (see later section) covers how there are some objectives which will only be implemented if time allows. However, for the core features, the project plan outlines realistic time frames for implementation.

1.5 Report Overview

The proceeding sections in this report are outlined below:

- Professional Considerations: this section will describe the professional and ethical considerations surrounding this project and how it will impact society
- Requirements Analysis: this section will describe what functions are to be implemented by the SSH Honeypot along with non-functional requirements
- Methodology: will outline the approach taken by the research and analysis phase of this project
- Design: will give a high-level overview of the design for the implementation
- Project Plan: details the plan for this project along with timings, schedule and objective time-frames
- Log: lists all meetings with supervisor
- Project proposal: (attached) original project proposal document

2 Professional Considerations

The subject of honeypots in computing can be a controversial one, therefore this section aims to address some of the major professional considerations of this project. The BCS[2] outlines a Code of Conduct[1] which should be adhered to when carrying out any IT project. The the BCS Code of Conduct sections: Public Interest, Professional Competence and Integrity and Duty to the Profession are applied specifically to this project in the proceeding sections. The responsible disclosure section describes how this project will deal with the social responsibility implications if a high impact vulnerability is discovered while carrying out this project. Finally, the honeypot ethics section will explore the ethical issues surrounding honeypots and aims to address these issues.

2.1 Public Interest

This project will collect IP addresses of attackers that login to the honeypot. Since IP addresses can be used to trace individuals on the internet these IP addresses will be stored securely and not revealed to the public (as per the Data Protection Act 1998 [4]). This project may at times use third party software or libraries, in such cases all relevant third parties shall be referenced and credit given. This project shall adhere to section 1(c) of the BCS Code of Conduct in that there shall be no discrimination made against any other person. This project is aimed primarily at the computer science and information security sectors. However, the outcomes of this project are not exclusive to the IT sectors and may benefit many other sectors that want to increase their computer security.

2.2 Professional Competence and Integrity

This project is being undertaken as a university undergraduate final year project. Therefore large portions of the project shall provide a strong learning experience. However, all knowledge areas of this project are within the subject areas taught under the university degree course. Where there are areas that have not been covered by the degree course relevant research shall be undertaken and references provided for background reading. This project values the opinion of others' and actively encourages honest criticisms of the work provided. Where constructive feedback for this project is received it shall be referenced within the project and actions taken as a result shall be shown. Finally, this project adheres to sections 2(f) and 2(g) of the BCS Code of Conduct in that this project shall "avoid injuring others, their property, reputation, or employment by false or malicious or negligent action or inaction" and that this project shall "reject and will not make any offer of bribery or unethical inducement".

2.3 Duty to Relevant Authority

This project shall be carried out with due care and diligence in accordance with the University of Sussex requirements. Advice has been sought and permission granted to carry out this specific project. This project shall try to avoid any situation which may provide a conflict of interest between this project and the University of Sussex. This project is being carried out by Simon Bell whom accepts professional responsibility for the work carried out within the project. This project shall not disclose any confidential information except with the permission of the University of Sussex or where required to do so by law such as under the Regulation of Investigatory Powers Act 2000 [12]. Finally, this project shall not "misrepresent or withhold information on the performance of products, systems or services (unless lawfully bound by a duty of confidentiality not to disclose such information), or take advantage of the lack of relevant knowledge or inexperience of others" as declared in section 3(e) of the BCS Code of Conduct.

2.4 Duty to the Profession

This project shall be carried out to a high standard in order to uphold the reputation of the profession and the BSC. The author of this project, Simon Bell, shall act with integrity and respect towards other members of the BCS and also other professionals and shall encourage and support fellow members in their professional development.

2.5 Responsible Disclosure

Due to the nature of this project, researching cyber-attack techniques, it may be possible to discover flaws or vulnerabilities in existing software, systems, protocols etc. Responsible disclosure[32] is a vulnerability disclosure model which states that any discovered vulnerabilities must be reported to the relevant authority (e.g. the software producer). Once reported, the relevant authority has a period of time to patch the vulnerability before the vulnerability can be publicly disclosed. If a vulnerability is discovered that has a high impact on society then the vulnerability may be disclosed sooner, as agreed with the relevant authority, in order to prevent a false sense of security in society.

2.6 Ethical Considerations

The subject of honeypots used in computing can be a controversial one, therefore this sections aims to analyse these controversies and explore the ethics surrounding honeypots.

One of the major issues of using a honeypot is that it could be seen as encouraging criminal activity, since the purpose of this project is to build a software system which allows attackers to gain unauthorised entry into it. However, one of the major aims of this project is to allow

attackers to believe that they're gaining unauthorised access (when in fact they're actually not). Therefore the attacker isn't actually gaining unauthorised entry at all. The details of the attackers' IP addresses will remain anonymous throughout this project, other then determining approximately which countries/states certain attacks originate from (although some attacks may route through a proxy which could be in a different location/country from the originating attack).

Another major issue surrounding the use of honeypots is that of deception (Dittrich, 2012 [19]). This is because participants in this project do not know that they're participating in a research project. This is a difficult subject to address due to the nature of the aims of honeypots. If attackers were informed about participating in a research project; they wouldn't participate. If participants were told to "attack" the honeypot it would likely produce unrealistic results since the attackers know they're being monitored.

Some experts consider honeypots to be unethical because they're strengthening the attackers' ability to detect honeypots. This could then allow attackers to stop targeting honeypots and, instead, only attack genuinely insecure systems. The result could be argued that honeypots are contributing to attackers becoming more sophisticated and creating a bigger problem.

However, the use of honeypots has resulted in many insecure systems being toughened, viruses and malicious code discovered and the information security sector as a whole has developed due to the use of honeypots.

3 Requirements Analysis

This section sets out the requirements of the software to be built: an SSH honeypot. The functional requirements set out what the software shall do and the non-functional requirements set out how the software shall do it.

There are currently two major public SSH honeypot applications: Kippo[7] and Kojoney[18]. As previously mentioned: both of these applications use the Twisted Konch package[20, p. 172] which is written in Python. Although the Twisted Konch library provides a good implementation of an SSH session, its major flaw is that it might be detected as being a honeypot. One of the main reasons for this is that the timings of Python based SSH honeypots are much slower than the service which they're emulating: the SSH daemon. This timing difference is because the Python programming language is slower to execute compared to C programming language implementations. This is mostly due to Python being a more abstract (or higher level) language than C.

One of the main advantages of the honeypot being produced in this project is that is can run on systems whereby access to the Python programming language is not possible. Also, this honeypot should provide a more suitable implementation for security researchers that requite a honeypot which behaves more similarly to the openSSH library than other existing SSH honeypots.

3.1 Functional Requirements

- SSH honeypot shall run as a server (daemon) in the Linux operating system environment
- Allow clients to connect to the server by initiating an SSH session
- Allow clients to try various different passwords
- Allow user to define a username and password to allow authorisation of attacker into the emulated shell environment
- Ability to log all login attempts and commands executed within the emulated shell environment
- If enough time: Implement virtual environment to execute uploaded malicious code

3.2 Non-Functional Requirements

- The honeypot needs to be reliable and secure (it cannot become or create a vulnerability to the host system)
- The system shall be stable
- Response times shall be less than 10 milliseconds
- The system shall be portable across multiple Linux environments that support the dependant libraries (libssh)
- The system shall not be exploitable
- The emulated shell environment shall produce its output with timings that are similar to the openSSH library shell environment in order to convince attackers that it's a real shell environment

4 Methodology

In order to research cyber-attack techniques, data needs to be collected. It's important to collect data reliably in order to provide the best data analysis. Since any server running a honeypot could itself become compromised, the data produced from the honeypot shall be communicated to a separate server for data analysis.

Any honeypot used in the research phase of this project shall, upon receiving a new connection from an attacker, transmit the data containing the monitoring of the attacker to securehoney.net. That way, if a honeypot server becomes compromised, an attacker will be unable to view logs from previous monitoring logs. This preserves the anonymity of attackers and protects any personal information.

At each phase, once certain features have been implemented, a current working version of the honeypot will be deployed to numerous servers to collect data. Each honeypot will log all attacks made to it and transmit these logs to a central monitoring server (securehoney.net).

4.1 Location

There are two main locations of this project; the first is the physical location where the author shall work, this is in Brighton (United Kingdom). The second is the location of the honeypots and the central monitoring server. One honeypot will be hosted by Rackspace on a VPS server (location: London) the other honeypot will be hosted by Amazon on an Amazon Elastic Compute Server (EC2). Finally, the monitoring server will be hosted by Xilo Communications Ltd (location: Sheffield).

4.2 Research Design

Data shall be collected by deploying honeypots to numerous servers on the internet. These honeypots shall wait for incoming connections from clients. One of the honeypots shall not have its IP address publicised and will collect data from people scanning IP address ranges. The other honeypot server shall be running a web server which contains content which is popular in society and might therefore attract attention from attackers.

4.3 Sampling procedure

The sampling procedure for this research will depend on the specific data to be extracted. For example: determining the most commonly attempted password will be a case of selecting the most commonly attempted password in the entire data set. In most cases, the entire data set will be used but it will be filtered depending on the data to be extracted.

4.4 Data gathering

Data will be gathered from the servers running the honeypot. These servers will transmit their data to a central monitoring server (securehoney.net). This central monitoring server will store data in a MySQL database[27]. Incoming data shall be processed via a script written in the programming language PHP[11].

4.5 Data analysis

Data will be analysed by querying the MySQL database with SQL queries and PHP code. The outputs of these queries will be presented on a statistics page, written in PHP. This statistics page will display tables and charts to represent the data.

5 Design

Based on the requirements analysis this section details a high-level design (or basic algorithm) for the honeypot:

- Client initiates request for SSH connection to honeypot server - relevant SSH protocol (RFC) data is transmitted to establish SSH session
- Server sends client request for username and password authentication
  - All authentication attempts received from client are logged (and sent to remote logging server)
- If correct username and password entered: send client authorisation success and welcome message
- Log all attempted commands (everything typed into the emulated shell)
- If client enters a command which is on predefined list; emulate that command
- Terminate SSH session if client sends close, connection is lost, or time-out (5 mins of inactivity) occurs

6 Project Plan

Phase 1: Research / background reading
Summer 2013

- Background reading: types of honeypots, network security and operating system security
- Complete Coursera course: "Malicious Software and its Underground Economy: Two Sides to Every Story"[16]
- Continued background reading: lookup various research papers on honeypots, SSL, SSH, malicious software, reverse engineering code
- Decide on best language to implement code in
  - Brief research and tutorials into the programming language Scala as a possible implementation language for project
  - Learn C programming language: tutorials[31], code examples, etc (decided to implement project in C programming language)

Phase 2: Establish SSH session
September & October 2013

- Implement code in C that will run honeypot as a server waiting for clients to initiate SSH sessions
- Allow clients to attempt authorisation (enter a username and password)
- Deploy initial version of honeypot to public server
- Log all attempted usernames and passwords and send to remote logging server
  - store received data from honeypots in MySQL database
- Setup securehoney.net

  - setup MySQL database to store all data
  - setup PHP script to handle incoming data from honeypots
  - create statistics page (implemented in PHP) to display data
- Interim report write-up

Phase 3: Authorise client, log all commands
November 2013

- Ability to set a username and password to authorise client
- Based on data from phase 2 set username and password to
  - most commonly tried on one alpha honeypot
  - a more complex password, that is still used reasonably frequently
- Authorise client when correct username and password provided
- Log all attempted commands by client and send to remote logging server
  - store received data into MySQL database
  - amend attempted commands into statistics page on securehoney.net
- Setup one of the honeypots to run a web server which contains information that will attract attackers

Phase 4: Emulating shell commands
December 2013 & January 2014

- Based on data from phase 3, determine commonly attempted commands
- Implement some of these commands as emulation (e.g. ls, w, wget)
- *If time allows:* try emulating a virtual environment to run uploaded code to
- Log all emulated commands and how client uses them and send to remote logging server
  - store received data into MySQL database
  - amend emulated commands into statistics page on securehoney.net

Phase 5: Analysis
February 2014

- Analyse data with aim of understanding the techniques used by attackers
- Analyse most commonly used username and passwords
- Analyse how username and password lists are created (dictionary, other lists etc)
- Analyse most commonly attempted shell commands
- Analyse uploaded files from attackers
- Analyse how host (honeypot) is used by attackers once compromised
- Production of charts etc, conclude any hypotheses
- This may involve theorising about certain username and password usage, determining if an attacker is running an automated script to execute the attack, determining if an attacker is trying to determine if theyre in a honeypot or not etc

Phase 6: Draft report
March 2014

- Start draft report
- Completed by 17 March 2014

Phase 7: Final report
April 2014

- Complete final report for hand-in on 17 April

7 Log

- **June/July/August 2013:** Met to discuss project ideas, areas to research, language implementation. Supervisor recommended Coursera course (Malicious Software and its Underground Economy: Two Sides to Every Story[16]), completed Coursera course and received certificate of achievement, looked at various research papers on different types of honeypots
- **Thursday 26th September 2013:** Met with supervisor to discuss progress on project. Agreed to produce a working C server which allows basic connection/login.
- **Thursday 10th October 2013:** Discussed automation tests to ensure project code is reliable, created GitHub page for project, discovered memory leak in code causing server to not accept new connections, discussed how to implement and create an SSH session using SSL, looked at various SSH libraries
- **Thursday 17th October 2013:** Stuck on implementing the SSH connection, looking at various libraries, website and twitter account setup, explored how Kippo uses the Twisted Konch library (looked at source code), read through SSH RFC, discussed interim report, discussed sharing honeypot on forums/reddit to publicise (when finished)
- **Thursday 24th October 2013:** First version of SSH honeypot server implemented, honeypot sends data to securehoney.net via the CURL method in C, this sends data to a PHP script on securehoney.net which inserts data into MySQL database, looked at statistics page

to see commonly used usernames and passwords, hypothesised about the use of complex passwords (password databases etc), discussed various statistic display options (graphs, charts etc), looked at darrenpopham.com (another SSH honeypot stats page, based on data produced from Kippo), discussed next phase (allow authorisation on honeypot) and interim report.

8 Proposal Document

See attached

References

[1] British Computing Society: Code of Conduct. http://www.bcs.org/category/6030.

[2] British Computing Society, The Chartered Institute for IT. http://www.bcs.org/.

[3] Brute force attack image. http://lightningbase.com/security/wordpress-brute-force-attack/.

[4] Data Protection Act 1998. http://www.legislation.gov.uk/ukpga/1998/29/contents.

[5] Dionaea, catches bugs. http://dionaea.carnivore.it/.

[6] Distributed denial-of-service attack image. http://www.betterhostreview.com/tag/ddos-attack-protected-hosting.

[7] kippo - SSH Honeypot - Google Project Hosting - Google Code. http://code.google.com/p/kippo/.

[8] libssh, The SSH Library. http://www.libssh.org.

[9] Man in the middle attack image. http://en.wikipedia.org/wiki/File:Man_in_the_middle_attack.svg/.

[10] OpenSSH, OpenBSD Secure Shell. http://www.openssh.com.

[11] PHP: Hypertext Preprocessor. http://www.php.net.

[12] Regulation of Investigatory Powers Act 2000. http://www.legislation.gov.uk/ukpga/2000/23/contents.

[13] Klaas Apostol. Brute-force Attack. 2012.

[14] Daniel J Barrett and Richard E Silverman. *SSH, the Secure Shell: the definitive guide*. O'Reilly Media, Inc., 2001.

[15] Franco Callegati, Walter Cerroni, and Marco Ramilli. Man-in-the-Middle Attack to the HTTPS Protocol. *Security & Privacy, IEEE*, 7(1):78–81, 2009.

[16] Lorenzo Cavallaro. Malicious Software and its Underground Economy: Two Sides to Every Story. https://www.coursera.org/course/malsoftware, 2013.

[17] George Chamales. The honeywall cd-rom. *Security & Privacy, IEEE*, 2(2):77–79, 2004.

[18] Jose Antonio Coret. Kojoney-A honeypot for the SSH Service. http://kojoney.sourceforge.net/, 2006.

[19] David Dittrich. The ethics of social honeypots. *Available at SSRN 2184997*, 2012.

[20] Abe Fettig and Glyph Lefkowitz. *Twisted network programming essentials*. O'Reilly Media, Inc., 2005.

[21] Xinwen Fu, Wei Yu, Dan Cheng, Xuejun Tan, Kevin Streff, and Steve Graham. On recognizing virtual honeypots and countermeasures. In *Dependable, Autonomic and Secure Computing, 2nd IEEE International Symposium on*, pages 211–218. IEEE, 2006.

[22] Brian W Kernighan, Dennis M Ritchie, and Per Ejeklint. *The C programming language*, volume 2. prentice-Hall Englewood Cliffs, 1988.

[23] Lars R Knudsen and Matthew JB Robshaw. Brute Force Attacks. In *The Block Cipher Companion*, pages 95–108. Springer, 2011.

[24] Iyad Kuwatly, Malek Sraj, Zaid Al Masri, and Hassan Artail. A dynamic honeypot design for intrusion detection. In *Pervasive Services, 2004. ICPS 2004. IEEE/ACS International Conference on*, pages 95–104. IEEE, 2004.

[25] Jelena Mirkovic and Peter Reiher. A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53, 2004.

[26] Ivatiti Mokube and Michele Adams. Honeypots: concepts, approaches, and challenges. In *Proceedings of the 45th annual southeast regional conference*, pages 321–326. ACM, 2007.

[27] AB MySQL. MySQL: the world's most popular open source database. http://www.mysql.com, 1995.

[28] Jon Postel. *Transmission Control Protocol*. RFC 793, September 1981.

[29] Niels Provos. Honeyd-a virtual honeypot daemon. In *10th DFN-CERT Workshop, Hamburg, Germany*, volume 2, 2003.

[30] Lukas Rist, Sven Vetsch, Marcel Kossin, and Michael Mauer. Know your tools: Glastopf-a dynamic, low-interaction web application honeypot. *The Honeynet Project*, 2010.

[31] Zed A Shaw. Learn c the hard way. http://c.learncodethehardway.org/book/, 2010.

[32] S Shepherd. Vulnerability disclosure: How do we define responsible disclosure? *GIAC SEC Practical Repository, SANS Inst*, 2003.

[33] Eugene H Spafford. Preventing weak password choices. 1991.

[34] Lance Spitzner. *Honeypots: tracking hackers*, volume 1. Addison-Wesley Reading, 2003.

[35] Guido Van Rossum and Fred L Drake. *An introduction to Python*. Network Theory Ltd. Bristol, 2003.

[36] Ping Wang, Lei Wu, Ryan Cunningham, and Cliff C Zou. Honeypot detection in advanced botnet attacks. *International Journal of Information and Computer Security*, 4(1):30–51, 2010.

[37] Stallings William and William Stallings. *Cryptography and Network Security, 4/E*. Pearson Education India, 2006.

[38] Tatu Ylonen and Chris Lonvick. The secure shell (SSH) protocol architecture. http://tools.ietf.org/html/rfc4253, 2006.

[39] Feng Zhang, Shijie Zhou, Zhiguang Qin, and Jinde Liu. Honeypot: a supplemented active defense system for network security. In *Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. Proceedings of the Fourth International Conference on*, pages 231–235. IEEE, 2003.