I assure on my honour that I have actively participated in these solutions, and that the solutions were developed independently from other groups.

Group members:
Nian Tang, Nian.Tang.4237@student.uu.se
Kai Hatje, Kai.Hatje.1791@student.uu.se

My name and personal number:
Ahmad Bijairimi, 780214-1171
ahab7847@student.uu.se

(2010-03-08)

Secure Computer Systems

OpenBSD Security Practices: What do they do, What does it give them

Group 8

1. INTRODUCTION

1.1 What is OpenBSD
OpenBSD is an "free, functional and secure" [1] operating system (OS). It is unix-like and currently officially runs on 12 different architectures, such as i386, amd64 and sparc. The OpenBSD Project was founded in 1994 by Theo de Raadt, using a fork of NetBSD 1.0 in 1995 [2] as the base for the OpenBSD OS. It has since become one of the leading OS' when it comes to security.

The main objectives of the OpenBSD project are to
- provide free code (via the BSD license)
- have high quality standards (of code)
- have correctness (of code)
- adhere to standards (POSIX, ANSI)
- Provide good crypto tools (SSH, SSL, IPSEC, ...)

all in order to provide better security [3].

Since security in an OS is very much dependent on its tools and software and the developers were unsatisfied with aspects like the licence or the quality of a number of tools usually used in an OS, there are quite a few spinoffs from the OpenBSD Project, the best known one probably being OpenSSH [4].

The main focus of OpenBSD is on security. This has resulted in a lot of "best practices" in OS and software development. The techniques used range from simple to quite advanced. One example of a rather simple security measure is that only required services are activated in a default

installation ("secure by default" being a leitmotiv since 1996 [3]). More advanced measures could be stack protection or advanced use of randomness.

These best practices have let to "Only two remote holes in the default install, in a heck of a long time!" [1], "a heck of a long time" being more than ten years. This slogan has resulted in some critique, as only the default services are enabled after an installation of Openbsd [12], which is not useful. However that may be, a lot of the ideas and practices used in OpenBSD have been taken to other Projects, making an impact on software outside the original project. This is intended and encouraged by the project founders, the licence being used in OpenBSD being the BSD licence [5].

The focus in this study is what the title suggests: Give an overview over the OpenBSD security practises. The goal is to outline what techniques are used to make OpenBSD more secure and to give a short introduction the techniques. The hope is that readers of this report get an idea of how and when security techniques are used and raise awareness of the lack of those in other OSes.

This report is structured as follows: Chapter 2 gives a short overview over security practices used in OpenBSD. Chapter 3 will cover the cryptographic aspects of OpenBSD more in depth. Chapter 4 will discuss the "secure by default" principle in more detail. Chapter 5 will summarize the findings and give a conclusion.

2. OPENBSD SECURITY PRACTICES

Most of this chapter is based on the presentation about "Protection measures in OpenBSD" [3] and the webpage about openbsd security [6].

This chapter focuses mainly on protection on the system level, not the network level.

2.1 "High level" security practices

Principles that are more processes than really technical measures and could therefore be put into the "high level" security category are the "secure by default" principle and the use of an audit process.

2.1.1 Secure by default

"Secure by default", as already mentioned, means that only essential services are activated in a default installation and that the default configuration of a service is providing security. This means that attack vector of a base install is very small, compared to other OSes. Another implication is that enabling a service requires manual action of the administrator. Quoting the OpenBSD website: "As the user/administrator becomes more familiar with the system, he will discover that he has to enable daemons and other parts of the system. During the process of learning how to enable a new service, the novice is more likely to learn of security considerations." [6].

2.1.2 Audit process

Auditing the code basically means reading it in search of bugs. The code of the critical software components of OpenBSD gets read several times, by different people with different auditing skills. The main objective is to find bugs, though it is always nice if it later turns out that bug would have become a security issue, if it had remained unfixed.

In OpenBSD, auditing happens at all the stages: design, coding, even simple modifications. Once an error is found, similar errors are searched for throughout the system.

This process is probably one of the main security features of OpenBSD. It takes a lot of time, but there are many payoffs. One example is that not only simple bugs have been discovered, but often whole new classes of security problems. Another example is that bugs fixed in OpenBSD components often let to similar fixes in components of other OSes (or their kernel), having the security of OpenBSD pass on to other systems.

With this process came a lot of different tools, like llvm and new functionalities in gcc (-Wbounded parameter), which of course can be used by other projects as well.

2.2 "Low level" security practices

"Low level" security practices are mechanisms that are used to prevent certain things from happening, eg. a buffer overflow.

2.2.1 strlcpy()/strlcat()

strlcpy() and strlcat() are functions replacing strncpy() and strncat(). Problems found with strncpy() and strncat() where for example that both functions are not very intiuitive, leading to programmer errors, and that there is a rather big performance penalty in using them over strcpy() and strcat(). Both strlcpy() and strlcat() guarantee NUL-termination and provide an easy way to detect truncation, making it easier for programmers to program secure code.

2.2.2 Memory protection

There are different means of memory protection in OpenBSD. One example is the use of canaries, named "Propolice". A canary is a random check value, placed on the stack, in front of local variables. Before a function returns, it is checked if the value is overwritten. An overwritten canary is a sign for an overflow.
Another example for memory protection is W^X. W and X stand for write and execute, the ^ stands for xor, meaning that on a page in memory either write or executed is permitted, but not both.

Another memory protection scheme is a randomized stack. While on other OSes the stack might grow strictly from on side to another when memory is allocated, OpenBSD allocates memory in different positions on the stack, making "stack smashing" attacks or buffer overrun attacks even harder.

2.2.3 Privilege separation

This means that it is tried to run system deamons with a uid that is not 0 (which would correspond to root). One way of doing this for deamons that

typically require uid 0 is to split them up into two processes, one that contains most of the programms logic and another one that has only the logic that needs the high privilege. One example for this is OpenSSH [13], where the unprivileged child processes all the network traffic, but needs to pose requests to the privileged parent, for example authentication request. If the child process gets corrupted by network traffic and thinks a user is authenticated, the parent process is still clean and prevents that user from logging in.

2.2.4 Privilege reduction

This means revoking privileges from formerly privileged processes after all the operations that needed those privileges are done. This prevents privilege escalation in the later runtime of the programs.

2.2.5 Chroot jailing

This is a technique that is very common to other OSes as well. Chroot stands for "changeroot", meaning that the root folder is changed for a program. Consider the following example: An webserver is running on a system, having /var/www/webdata/ as the folder where the files it serves are located. Then, a user could possibly get access to the configuration files in /etc/ on that system by requesting a file from http://hostname.tld/../../../etc/, where ../ is one directory above the current one. If the webserver is run in a changeroot environment, its root directory would change to /var/www/webdata/, and now ../ would stop in the folder that is set as the root.

Next two sections will discussed two main aspects of OpenBSD security, Cryptography and Secure by Default.

3. CRYPTOGRAPHY IN OPENBSD

Cryptography is very important for an operation system as it can strongly enhance the security of it. Hence the OpenBSD project has embedded cryptography into numerous places in the operating system. [7] The utilization of Cryptography can be found in the following areas: SSL, IP Security, Pseudo Random Number Generator, Open SSH, and Cryptographic Transforms and Hash Function.

3.1 SSL

OpenBSD provides Secure Socket Layer (SSL) and Transport Layer Security by a toolkit called libssl. Due to patent restrictions, libssl in the OpenBSD distribution supports only digital signatures with DSA, but an additional package is provided for users outside the USA to add back RSA-signature support [8].

3.2 IP Security

IPsec is a remedy for IP in order to deal with security issue and it provides strong end-to-end security. IPsec is consisted of two parts: ESP (Encapsulating Security Payload) and AH (Authentication Header). Two modes, transport mode and tunnel mode, are offered for operations. The information about which algorithm and which key to use to decrypt or verify an ESP packet is stored in Security Association (SA).

OpenBSD was the first operating system to ship with an IPsec stack [7]. OpenBSD has APIs to setup and maintain the SA. There are some algorithms available for encryption such as DES, 3DES, Blowfish. For key management, two daemons are available, *isakmpd* implementing IKE and *photurisd* implementing Photuris [8]. In OpenBSD, there are hardware acceleration for IPsec.

3.3 Pseudo Random Number Generators

Random numbers are very important because many protocols like Kerberos, TCP SYN, DNS query-IDs are using random numbers. A Random Number Generator should hold such properties:

1. the output should be unpredictable.
2. No pattern can be traced in the generated numbers.
3. The algorithm will have same output while giving the same initial values.

In order to give an identical input for the Pseudo Random Number Generator (PRNG), OpenBSD keeps a Kernel randomness pool which called entropy pool. The OpenBSD kernel uses the mouse interrupt timing, network data interrupt latency, inter-keypress timing and disk IO information to fill an entropy pool [7].

And then MD5 hash is used to produce the input of the algorithm. In this way, the system achieves randomness as this is gained by the nature of multi-user operation system. The arc4random(3) is the function which use the entropy pool as the seed to the algorithm.

In FreeBSD, PIDs are generated by PRNG to satisfy security demands. By using random numbers of Inode, it would be harder for an attacker to get filehandles. One time password database also has benefit when using PRNG. RPC transaction IDs(XID) , NFS RPC transaction IDs(XID), IP datagram IDs, TCP ISS value and salts for password algorithm can be generated by using PRNG.

3.4 Kerberos

The Kerberos system was developed at MIT within the project Athena in the 1980s.[Computer Security] Kerberos is a widely accepted authentication protocol. In FreeBSD, the login, xdm, and su programs are used for Kerberos. Other programs using Kerberos for authentication includes cus, sudo, and xlock [8].

3.4 OpenSSH

OpenSSH offers protections for data communication as all information are encrypted in OpenSSH. Strong encryption algorithms (3DES, Blowfish, AES and arcfour) are supported by OpenSSH. Strong authentication is another attribute and protects against several security problems, e.g., IP spoofing, fakes routes, and DNS spoofing [9]. Ticket for Keoberos and AFS can be passed through OpenSSH. OpenSSH also supports port forwarding and agent forwarding. OpenSSH compresses data before encrypte it. This may give performance increase to low bandwidth network connection.

3.5 Cryptographic Transforms and Hash Function.

OpenBSD provides DES, 3DES, Blowfish as cryptographic transform functions. Cryptographic Hash functions includesMD5, SHA1, and RIPEMD-160. The system has special encrypt function Bcrypt which uses a 128-bit salt generated by acr4random(3) and encrypts a 192-bit magic value. S/Key, a one-time password system used for authentication, uses hash function like MD5 or SHA1 [8].

3.6 Cryptographic Hardware Support
OpenBSD has some support for cryptographic hardware. One example is OpenBSD Cryptographic Framework (OCF), a service virtualization layer implemented inside the kernel that provides uniform access to cryptographic hardware accelerator cards by hiding card-specific details behind a carefully designed API [10].

4. SECURE BY DEFAULT

Quoted from [6] *"To ensure that novice users of OpenBSD do not need to become security experts overnight (a viewpoint which other vendors seem to have), we ship the operating system in a Secure by Default mode. All non-essential services are disabled. As the user/administrator becomes more familiar with the system, he will discover that he has to enable daemons and other parts of the system. During the process of learning how to enable a new service, the novice is more likely to learn of security considerations".*

Secure by default is the main principal or motto of OpenBSD. This principal is really hard to achieve as lots of hassle and trade-off that must be taken into account. In spite of this, OpenBSD developer team has done a really great job in achieving their goal to be champion in security. This is mention in [6] that their goal is to be NUMBER ONE in the industry for security. This goal is achieved by striving in three features:

- Extensive code auditing (this feature is mentioned briefly in section two)
- Integrated cryptography (this feature is discussed in section three)
- Thorough documentation, that can be viewed extensively in [11].

In code auditing, there are two levels of code that is analysed:

- Kernel level, which is the core of OpenBSD
- User code, this includes user code in ports and packages.

and the audit process that was first initialised in 1996, is still ongoing until now. All codes including all changes that is made are viewable via CVS.

OpenBSD developer team has done a very great job in analysing each and every files of the operating system one by one to find software bugs. As mentioned in [6] the effort has been started since 1996, and it is still continuing. This effort is not a mere effort as code auditing maybe one of the hardest work to do, which is far harder than writing new codes. One can imagine how hassle and time consuming it is to analyse thousands of lines

of codes, but it is proved by OpenBSD team, that this effort is achievable even with only a small team (six to twelve developers). As a result, OpenBSD has become a champion when it comes to what kind of operating system to choose for implementing security devices such as firewall.

4.1 OpenBSD as a firewall

Firewall is one of the most essential network security devices which operates as the first defence. Packet filter (PF) is the solution developed by OpenBSD, specifically for the purpsoe of firewall implementation. PF is a stateful packet filter firewall that runs on the lower level of TCP/IP stack which will filter all packets in and out of the network. The filtering is done based on firewall set of rules with the default rule is to deny all in and out packets. Then, by learning and understanding what services are needed for the system, the ports will be allowed to pass through. Because of secure by default policy, OpenBSD is already secure by default, which means it is ready to be set up as a firewall box at any time. The only requirement needed is to have a second Ethernet card enabled on the system as OpenBSD required at least two network cards to build up a firewall, one card for a local network connection, and the other card is for a connection to router.

4.2 OpenBSD as a server

OpenBSD is the best option to set up a secure web server as the system is readily built-in with SSL-ready httpd and RSA libraries. As for the OpenSSH, as it was developed by OpenBSD team themselves, it should work the best with OpenBSD. In addition, as the default system is secured by default, the web server administrator does not have to worry much about blocking port this and that. What needs to be done is just to get the web server up, and open the ports needed to run web services and it is done. In addition, there are many more built in features for web server such as load balancing with pf, data compression and optimisation with a built-in Apache or lighthttpd and many more.

4.3 OpenBSD as a desktop

However when it comes to a desktop computer, OpenBSD might not be preferred by many users as it is hard to use. Due to OpenBSD strict rules and policy, it support less softwares and hardwares.

5. SUMMARY

To summarise, OpenBSD is a really good operating system with a special emphasise on providing a free, functional and secure system. From security point of view, OpenBSD is the best solution for firewall and secure servers, but not for desktop. This is certainly alright, and it is not a big issue where OpenBSD has to struggle to win the heart of desktop users. In this world, there is no single operating system that can satisfy all types of users needs. As for OpenBSD, the target group might be system administarators and not normal desktop users. Meanwhile for desktop operating system, there are many players out there that are making millions grand of dollar by selling their operating system. So, just let them to take care of desktop users need, and OpenBSD just need to stay still as they are. However, the security practices of OpenBSD actually should be

followed by other operating systems as well, so that they can provide a secure desktop to users. To have only secure server is not enough, because each and every entities in the Internet are interconnecting with each other. Hence, secure by default principal must be used as a motto for any kind of software development by following the practice of OpenBSD.

References:

[1] http://openbsd.org/, various, 2009.
[openbsd_platforms] http://openbsd.org/plat.html, various, 2010.
[2] http://en.wikipedia.org/wiki/OpenBSD#History_and_popularity, various, 2010.
[3] http://homepages.laas.fr/matthieu/talks/openbsd-h2k9.pdf, Matthieu Herrb , 2009.
[4] http://www.openssh.com/history.html, various, 2004.
[5] http://www.openbsd.org/policy.html, various, 2007.
[6] http://openbsd.org/security.html, various, 2009.
[7] http://www.openbsd.org/crypto.html
[8] Theo de Raadt *et al.* The OpenBSD Project
[9] http://www.openssh.com/features.html
[10] Angelos D. Keromytis *et al*
[11] http://www.openbsd.org/faq, various, 2010
[12] http://en.wikipedia.org/wiki/OpenBSD#Security_and_code_auditing, various, 2010.
[13] http://www.openssh.com, various, 2004.