A Study on Cyber Defence Honeynet Technology and Configuration Examples

Huimin GUO [1], Jiaomin LUO [1] and Qian GENG [1]

[1]*School of Jincheng College,* Nanjing University of Aeronautics and Astronautics, Nanjing, China

***Abstract* — Honeynet is a new type of network security technology based on active defense, its main feature is to collect the attacker's attack interest rates to achieve more effective security defense. With the extensive application of honeypots, the weaknesses of traditional honeypots are highlighted. For example, they can only capture activities that interact with them, and the true identity is under high risks to be identified. Therefore, they are replaced by the honeynet techniques, comprising a group of honeypots that are highly interactive to obtain extensive threatening information. There are three aspects of challenges faced by the honeypots, including: i) hiding the capture tools, ii) capturing the encrypted session data, and iii) converting the channel of data transmission. In this paper we provide detailed implementation and deployment solutions of Honeynet techniques and present the emerging research directions in this area.**

Keywords - Honeypot; Honeynet; Capture tools

I. INTRODUCTION

The honeypot is a kind of security resource, with a value to be scanned, attacked, and captured. It sacrifices the real operating system (generally the Linux-based operating system) with no patches to deceive the intruder, aiming to collect the hackers' attacking methods and protect the real host targets. In the work of [1-3], The structure of honeypot is described as we can see from *Fig.1*.

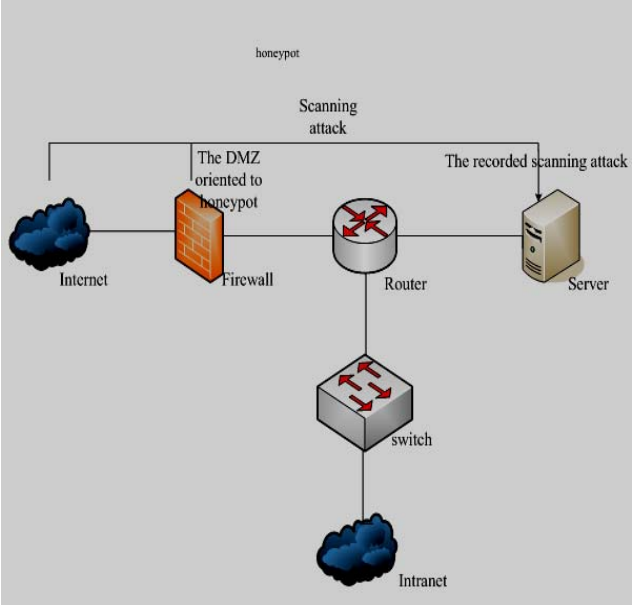

Figure 1. The structure of honeypot

Traditional honeypots have a lot of advantages. For example, they keep a high fidelity on collected data, and they do not rely on any complicated test techniques. In this way, the false negative rate and the false positive rate are decreased. The honeypot technique can be used to collect new attacking tools and methods, while most currently used Intrusion Detection Systems detect known attacks only by feature matching. However, with the extensive application of traditional honeypots, their weaknesses are exposed: (1) The honeypot technique only monitors and analyzes behaviors attacking against the honeypots. They cannot monitor the entire network the same as the Intrusion Detection System through techniques like bypass interception,(2) The honeypot technique cannot protect the information system with loopholes, and it can be utilized by attackers, bringing certain security risks, and (3) There are more and more behaviors of attackers on the encrypted channels, such as IPSec, SSH, SSL .etc. It takes time to decode the captured data, increasing difficulties in analyzing the attack behaviors.To resolve the aforementioned problems, the honeynet technique emerges.

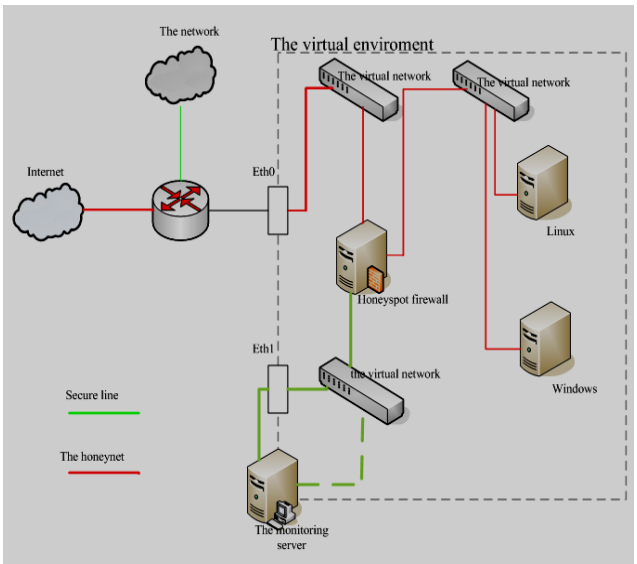

Figure 2. The design of the structure of the third generation honeynet system

From *Fig.2*, We can see the design of the structure of the third generation honeynet system. In essence, the honeynet technique is a kind of highly interactive honeypot technique. It is distinguished from the traditional honeypot technique for its hacker trap network system architecture, which comprises one or more honeypots, to ensure the high controllability of the network and provide various tools to facilitate the collection of and analysis on the attack information. For more results on this topic, we refer readers to [4-6] and the references therein show us the differences.

II. CHALLENGES FOR HONEYPOTS ON THE HONEYNET.

The honeynet is a systematic architecture. To successfully deploy a honeynet environment, two requirements raised based on the weaknesses of the traditional honeypots must be strictly met, which are data control and data capture.

Data control is a mechanism limiting attackers' behaviors, to reduce the security risks. Data capture is used to monitor and record all attackers' behaviors on the honeynet. For example, it can record keystrokes during encrypted sessions, restore SCP-copied files, capture passwords recorded by the remote system, and restore the passwords of protected binary system programs. In the work of [5, 6], the problem of honeypot is referred. These captured data will be analyzed, so that the hackers' tools, strategies, and motivations will be learned.

The difficulties include: (1),It requires collecting as much data as possible while ensuring the process of the data capture cannot be detected by hackers, (2) More and more hackers use encryption tools to protect their transmission channels. Even if the target machine is not provided with encryption services, they can install services like SSH, encrypted GUI client, or SSL by themselves. If the password is not obtained, the network-based data capture tools cannot view the transmitted data, and (3) It requires gathering all collected data to the data collection server inside the honeywall through a secret channel.

III. SOLUTIONS

A. Capture Tool Hiding

1）Module Hiding

The capture data tool works in the modularization mechanism by dynamically being loaded to the kernel to be a part of it after the Linux system is initialized. After the module of the capture program is loaded to the kernel, a linked list records the information of the loaded module. Users can view the information in the module file under the proc file system dynamically generated in the memory. When the privileged user root invokes the /sbin/insmod command to load a module, a system invokes sys_create_module( ). This function can be found in the kernel/module.c inside the source code of Linux 2.4. It will insert the struct module which contains the information of the newly loaded module into the module linked table named moudle_list.

```
sys_create_module(const char *name_user , size_ t size)
{
        ......
mod->next = module_list;
mod->name = (char *)(mod + 1);
mod->size = size;
memcpy((char*)(mod+1),name,namelen+1);
put_mod_name(name);
module_list = mod;
        ......
}
```

Figure 3. Key codes of create_module

As is showed in *Fig 3*, When a module is loaded, its information will be inserted in the header of a singly-linked list. When hackers intrude into the honeypot system, they can find and unload the suspicious honeypot capture module according to the existing loophole, to deactivate the honeypots.

To hide the module, delete the pointer of it in the linked list after the module is loaded. By doing this, the module cannot be found by iterating the list, As the key codes we can see from *Fig. 4.*

2）Process Hiding

A process is an instance of a computer program that is being executed which is constantly changing with the execution of a program. The honeypot capture program may execute multiple processes inside the system. The information of all processes can be obtained using process information query commands like ps, making it easy to expose the existence of the honeypot. Since there is no system call that directly queries the process information in Linux, the process information query commands like ps are realized by querying the proc file system, which is a virtual file system realized through the interfaces of the file system to output the operating status of the system. In the form of a file system, the proc file system provides an interface between the operating system itself and the application processes, making the application programs obtain the operating status of the system and internal data of the kernel in a safe and convenient manner. It also can modify the configurations of certain systems. Realized in the form of a file system, the proc file system can be accessed as a common file. However, it only exists in the memory, which makes it possible to hide all the files inside the proc file system in a way of hiding common files, so as to hide the process.

```
……
struct module *mod_current;
mod_current = &__this_module;
While (mod_current.next)
{
    if (strcmp(mod_current.next,str));
    { mod_current.next = mod_current.next->next;
      break;
    }
  else mod_current=mod_current.next;
  }
……
```

Figure 4. Key codes of module-hiding

The system call used to query file information in the Linux system is sys_getdents( ). When the information related to a file or directory is queried, the Linux system invokes the sys_getedents( ) fun the user end. If the system call is changed, delete the information related to certain files in the proc file system in the result. By doing this, all programs using the system call cannot access the file, so as to hide the process.

To determine whether a file is under the proc file system, observe whether it only exists in the memory rather than other actual devices. Therefore, the kernel of the Linux assigns to the file a specific primary device number 0 and a specific secondary device number 1. In addition to these numbers, there is no other i nodes corresponding to the file on the external storages. Therefore, the system also assigns a special node number PROC_ROOT_INO (the value is 1) to the file. However, the index node 1 on the device is not used. In this way, the methods to determine whether a file is under the proc file system are as follows: (1) Get the file corresponding to the inode structure d_inode, (2) if (d_inode->i_ino == PROC_ROOT_INO && !MAJOR(d_inode->i_dev) &&MINOR (d_inode - >i _dev) == 1)

The prototype of the file information query system calls are as follows:Int sys_getdents(unsigned int fd, struct dirent *dirp,unsigned int count).

The character fd is a file descriptor of the directory file. The function reads the related dirent structure according to fd, puts it in the dirp, and provides pseudo codes that hide certain processes according to the aforementioned analysis. Through the above analysis, the pseudo code representation of the specific process is given as we can see in *Fig.5*.

B. Session Encryption Data Capture

Encrypted information must be decrypted somewhere if it is used. The unencrypted data can be captured by bypassing the encryption process, and then obtain the unprotected data, which is the basic mechanism of decryption. The binary Trojan horse program is one method to cope with encryption. When an intruder breaks through the honeypot, he/she may log in to the captured host using the encryption tools like SSH, and input the password. At this moment, the Trojan shell program can record the behaviors. However, the binary Trojan program can be easily exposed, and the intruder may install their own binary programs.

```
Hiding_Proc(unsigned int fd, struct dirent *dirp, unsigned int count)
 { /* Call the original system call */
    sys_getdents(fd,&dirp,count);
   Get the corresponding node FD;
   if(This file is proc file system && The file name needs to be hidden.)
   { Remove the file from  dirp ;}
   }
```

Figure 5. Key codes of process-hiding .

Replacing the original function means changing the function pointer of the system call table. When the standard read ( ) and write ( ) functions are invoked in the processes on the user end, a system call is generated, with an offset index when the call is mapped to the system call table. The capture module changes the index of the read and write pointers to point to its own function implementations, when the command to read the kernel data is implemented, the read and write calls changed by the capture module are executed. By doing this, the capture module views all the accessing data through the system calls. This technique can also be used to change any system call to be monitored.

C. Establishment of Hidden Data Transmission Channel

When the honeypot captures data, it has to transfer the data to the honeynet server before the intruder notices the capture. Honeypots are usually deployed on a LAN. If the capture module simply uses UDP streams to transfer the data to the server, the intruder can determine whether there is a honeypot system only by intercepting the data transmission on the network. However, the capture module can use the UDP streams to transfer data to the server after it changes the kernel so that the user cannot access the data packages, including the data packages of the same kind with the same configurations sent by other hosts. When the capture module transfers the data on the network, even the operating system cannot stop the transmission of the data packages.

If the data capture module which is improved in the aforesaid manner is installed on each honeypot on every LAN, the intruder cannot discover any data of the capture module. However, the server end can access all the data captured by the honeypot clients. Each read ( ) or write ( ) invoking request may generate one or more log data packages, each of which contains one aspect about the invoking contents and the accessed data. Each package also contains one record comprising the description of the

invoking process, generating time of it, and the size of the recorded data. All these packages are generated and transferred by the capture module rather than the TCP/IP protocol stack. Therefore, the system cannot access or block the data packages. As soon as the data packages are created, they are directly sent to the device driver, bypassing the original socket code and packet filtering code.

There is also another problem to be resolved, how to stop honeypot A from detecting the data packages captured by honeypot B? Ethernet switching cannot resolve this problem, because the data packages do not obtain the destination MAC address corresponding to the destination IP address through ARP. Therefore, the packages are naturally immune to ARP spoofing. However, in some occasions, data packages on honeypot B can be viewed from honeypot A, so that intruders can execute the sniffer on honeypot A to capture the data packages from honeypot B on the LAN. To resolve the problem, the capture module should realize the unique original socket implementation in the package generating mechanism, so that data packages from other honeypots on the LAN can be ignored. The preset destination UDP port and magic number (used to determine whether the data package needs to be hidden or ignored) are defined in the transferred data packages. If the two values are both matched, the data package is ignored. In this way, honeypot A will discard the data package from honeypot B and transfer them to the next data package, so that the intruder cannot capture the data package even by using the sniffer.

IV. CONCLUSIONS.

The honeynet technique is essentially the application of a group of highly interactive honeypot techniques. The honeynet is a network comprising multiple honeypots used to interact with attackers. This paper provides detailed research and resolutions to the problems faced by the honeypots in the honeynet: how to hide themselves, how to capture session encryption data, and how to establish a hidden data transmission channel with the honeynet gateway. However, there are some insufficiencies. For example, while protecting the data itself, the mentioned method and process to hide the honeypot data capture module also bring some problems: the capture module cannot be unloaded once it is loaded, and the root user cannot find it; once there is bug in the capture module, the kernel may become unstable and even the system may break down. These problems may impact the normal operation of honeypots and even the performance of the entire honeynet. Further researches are required to resolve these problems.

ACKNOWLEDGMENT

This work is supported by the Natural Science Foundation of Jiangsu Province of China (No. 15KJB510015), the Foundation of Graduate Innovation Center in Nanjing University of Aeronautics and Astronautics, China (No. KFJJ20150403) and the Fundamental Research Funds for the Central Universities. Besides, the work is also supported by The fine course, jincheng college of nanjing university of aeronautics and astronautics (No. 2012-J05). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] Li, Zhitang and Xu, Xiaodan, Analysis of dynamic honeypot and its design, Journal of Huazhong University of Science and Technology (Natural Science Edition), vol. 33, no.2, p86-88,February 2005.

[2] Abbas, Cláudia J. Barenco, etal, Implementation and attacks analysis of a honeypot, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Computational Science and Its Applications-ICCSA 2007-International Conference, Proceedings, vol. 4706 LNCS, no.PART2, pp 489-502, 2007.

[3] Ciglaric and Mojca, etal, Network intrusions research: Honeypot Simx. Electrotechnical Review, vol77, no4, p173-178, 2010.

[4] Mairh, Abhishek. Honeypot in network security: A survey. ACM International Conference Proceeding Series, Proceedings of the International Conference on Communication, Computing and Security, pp600-605, 2011.

[5] Ghourabi and Abdallah , Honeypot router for routing protocols protection. Post-Proceedings of the 4th International Conference on Risks and Security of Internet and Systems, pp.127-130,2009.

[6] Akiyama, Mitsuaki, etal. Client honeypot multiplication with high performance and precise detection. IEICE Transactions on Information and Systems, vol. E98D, no. 4, pp.775-787, April 1, 2015.