2004

Honeyfiles: Deceptive Files for Intrusion Detection

Yuill, Jim

þÿYuill, J., Zappe, M., Denning, D., and Freer, F., Honeyfiles: Deceptive
þÿDetection, Proc. IEEE Workshop on Information Assurance, U.S. Military
Point, NY, June 2004. (.pdf)

http://hdl.handle.net/10945/37180

Proceedings of the 2004 IEEE
Workshop on Information Assurance
United States Military Academy, West Point, NY June 2004

Honeyfiles: Deceptive Files for Intrusion Detection

Jim Yuill, Mike Zappe, Dorothy Denning, and Fred Feer

Abstract: This paper introduces an intrusion-detection device named honeyfiles. Honeyfiles are bait files intended for hackers to access. The files reside on a file server, and the server sends an alarm when a honeyfile is accessed. For example, a honeyfile named "passwords.txt" would be enticing to most hackers. The file server's end-users create honeyfiles, and the end-users receive the honeyfile's alarms. Honeyfiles can increase a network's internal security without adversely affecting normal operations. The honeyfile system was tested by deploying it on a honeynet, where hackers' use of honeyfiles was observed. The use of honeynets to test a computer security device is also discussed. This form of testing is a useful way of finding the faulty and overlooked assumptions made by the device's developers.

Index terms – deception, intrusion detection, computer security, file servers

I. Introduction

Honeyfiles are an intrusion detection mechanism based on deception. Specifically, a honeyfile is a bait file that is intended for hackers to open, and when the file is opened, an alarm is set off. For example, a file named *passwords.txt* could be used as a honeyfile on a workstation. Hackers who gain unauthorized access to the workstation will be lured by the file's name, and when they open the file an alarm will be triggered.

The concept of deploying bait files against hackers was pioneered by Cliff Stoll during his investigation of the German hackers who had penetrated his system at Lawrence Berkeley Labs, and elsewhere, in search of defense information that could be sold to the KGB [1]. To determine the origin of the attacks, Stoll needed a way of keeping the hackers on-line long enough to trace their connection. This was done by creating bait files that would appeal to the hackers and keep them occupied. The honeyfiles described in this paper extends Stoll's concept to an automated intrusion-detection system for end users. It monitors all file accesses and provides alarms whenever the bait files are accessed.

Honeyfiles are implemented as a file server enhancement, and the file server's users can make any of their files a honeyfile. Alarms are sent by e-mail directly to the user, and services can be used to securely forward the e-mail to a phone or pager. With honeyfiles, detection mechanisms can be effectively deployed, as they are placed by the end users who are intimately familiar with the network's file spaces. In addition, when an alarm is sent, those end users can easily and effectively interpret it.

Honeyfiles can be used to detect unauthorized access to computers whose file space is mounted from a file server. For all but the smallest of organizations, standard industry practice is to store user and application data on file servers. By implementing the alarm system on the file server, honeyfiles provide defense in depth for the file server's clients. Also, in protecting the clients, honeyfiles can detect unauthorized access gained through unknown attacks, as well as unauthorized access gained through unintended file-access permissions.

When effectively deployed, it will be difficult for hackers to avoid honeyfiles, and honeyfiles show potential for avoiding some of the problems frequently encountered by network intrusion-detection systems (NIDSs), such as high false-positive rates and also high false-negative rates for unknown attacks. Honeyfiles offer several additional benefits, such as the opportunity to increase a network's internal security without impairing its normal operations. Further, the honeyfile system can be used to detect unauthorized access to real files (in addition to bait files), and this provides substantial advantages over alternative techniques such as cryptographic checksums for detecting file modification.

A prototype honeyfile system has been implemented on the Network File Server (NFS), and it has been tested by subjecting it to hackers. Honeyfiles, and the prototype, are further described in the following sections.

This research was made possible by funding from The Office of Net Assessment, in the Office of the Secretary of Defense.

ISBN 555555555/$10.00

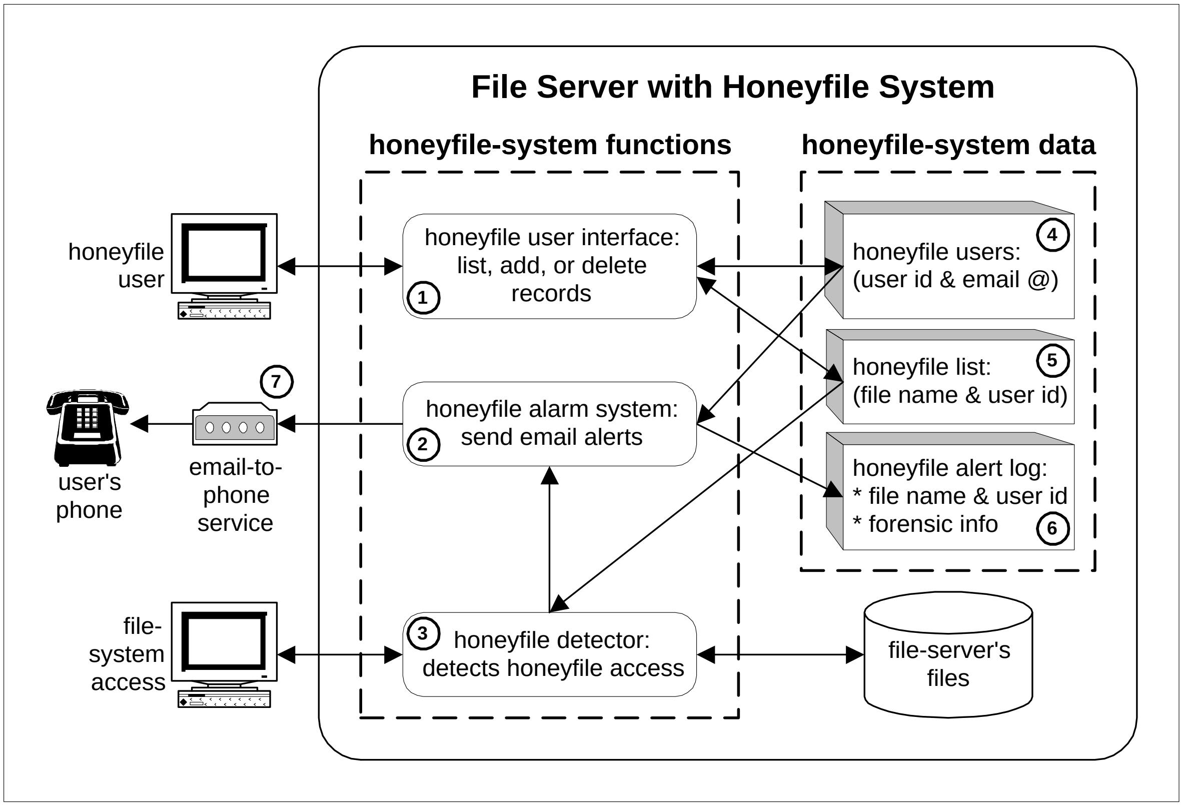
File Server with Honeyfile System
honeyfile-system functions
honeyfile-system data
honeyfile user
honeyfile user interface: list, add, or delete records
1
honeyfile users: (user id & email @)
4
honeyfile list: (file name & user id)
5
7
user's phone
email-to-phone service
honeyfile alarm system: send email alerts
2
honeyfile alert log: * file name & user id * forensic info
6
file-system access
3
honeyfile detector: detects honeyfile access
file-server's files

Figure 1 : The honeyfile system

II. The honeyfile system

Honeyfiles are implemented by a *honeyfile system*, and it provides the necessary file-system and alarm functions. The file-system functions are implemented as an enhancement to a network file server, as illustrated in Figure 1. The system's components are numbered in the figure and their descriptions follow.

Any file within the user's file space can be a honeyfile. The honeyfile system provides an interface whereby file-server users specify their honeyfiles (1). A file records the system's honeyfiles (5). Each record contains a file name and user ID. Honeyfile alarms are sent as email messages, so the user also provides an email address to be used. The email messages are called *email alerts*, or simply *alerts.* A file records the system's users (4). Each record contains a user ID and email address.

To detect access to honeyfiles, the honeyfile system monitors all file access on the file server (3). When a honeyfile is accessed, an alert is sent (2), and it is logged (6). The alert includes the name of the opened honeyfile, and forensic information for incident response, such as the IP address of the computer that opened the file.

The network can be configured for the email alerts to be sent in a secure manner. Ideally, they will be sent to an automated service that will call the user's cell phone and digitally display the email message (7). This ensures secure delivery of the alert should the user's mail client also be compromised. Phone delivery also enables the user to be notified while away from his computer. We implemented a prototype honeyfile system for NFS, on RedHat Linux 9. We plan to distribute the prototype as open source. The prototype is working, documented and tested. This paper is an abridgement of the prototype's documentation.

III. Using honeyfiles

Honeyfiles can detect the hacker's investigation and copying of files, including:

- the hacker's personal perusal of the file space. Hackers can be tricked into opening files with alluring names that indicate the file is of value.
- the hacker's use of search tools to find particular types of files, e.g., file names containing the string "password". These tools can examine file names or

ISBN 555555555/$10.00 © 2003 IEEE

contents. Honeyfiles can be created to match common hacker searches.

- the hacker's use of tools like *tar* and *zip*, to copy and steal the contents of entire directories. Such copying can be detected by placing honeyfiles in directories that are likely to be stolen, and the honeyfile's name will blend in with the other files, e.g., "sysrun1.dll".

There are four types of files that are generally of interest to hackers, and that can often be used as honeyfiles:

- files with information about accessing and using other systems, such as password files (*passwords.txt)*, user manuals (*customer-accounts-system.pdf*), and security documentation (*vpn- instructions.doc)*,
- system or application programs that the hacker may run, but that authorized users would not run, such as the gcc compiler,
- files that contain evidence of the attack, such as log files, and
- files that contain information of use other than hacking, such as credit card numbers, intellectual property, expected stock market prices, and military intelligence.

A honeyfile should be named and located in such a way that its owner will not be inclined to open it accidentally. One technique is to give a honeyfile a name that appears unusual only to its owner. The unusual name can help jog the owner's memory and recognize the honeyfile. For example, a honeyfile password file could be named *complete-passwords.txt*. Its owner has no partial password files, so the prefix "complete" will help him recognize the honeyfile.[1]

Honeyfiles can contain deceptive content, such as fake user-IDs and passwords. Deceptive file-content can take on a plethora of uses and forms, and it can be used independently of honeyfiles. In order to concentrate on central honeyfile functions, this paper does not address deceptive content in honeyfiles. Instead, it focuses on honeyfile deceptions involving just file system information, i.e., the file's location and its directory entry, including its name.

IV. Honeyfile uses

This section addresses honeyfiles' detection capabilities, tactical capabilities, and ease of use.

A. Detection Capabilities

Honeyfiles' detection capabilities include the following:

- Honeyfiles can detect unauthorized access to computers and file systems:

The primary strength of honeyfiles is their ability to detect unauthorized access to computers whose file-space resides on a file server. For example, a workstation stores its user file-space on a file server, and the workstation automatically mounts the file-space at boot time. If a hacker breaks into the workstation, his presence will be detected if he opens a honeyfile within the user file space.

In general, honeyfiles detect unauthorized access to the file spaces on a file server, including: **1)** compromise of the file space's user ID and password, **2)** compromise of weak or defective authentication mechanisms on the file server, e.g., NFS' notoriously weak authentication, and **3)** exploitation of errors made in granting file-space permissions, e.g., accidentally making the file space "world readable".

- Honeyfiles can be used to detect unauthorized access gained through unknown attacks:

Honeyfiles detect the hacker after he gains unauthorized or unintended access. The detection mechanism is independent of the specific techniques used to gain access. This is one of honeyfiles' primary contributions. Honeyfiles offer a unique opportunity for detecting attackers who are able to defeat conventional defenses. This makes honeyfiles especially useful for protecting high-value systems that are subject to such skilled attacks.

- Hackers can be highly vulnerable to honeyfile deceptions:

Honeyfiles take advantage of several deception vulnerabilities in most hackers' intelligence collection and analysis: **1)** when hackers initially access a file space, they must search it in order to locate valuable data. If the hacker's search can be anticipated, honeyfiles can be placed where he is likely to encounter them. **2)** The hacker's limited knowledge of the file space makes it difficult for him to discern what truly belongs there, and his naiveté makes it easy to create deceptive honeyfiles. **3)** It can be very difficult for the hacker to detect a honeyfile before opening it. The honeyfile deception is created using a small amount of information, i.e., the file's directory entry, and usually, there is no way for the hacker to cross-verify the information, and **4)** In most instances, if the target wants to know a honeyfile's contents, his only option is to open the file and trigger the alarm.

[1] Unless stated otherwise, this paper's masculine pronouns refer to both men and women.

ISBN 555555555/$10.00 © 2003 IEEE

- Honeyfiles can be used to protect a wide variety of files and computer systems:

A honeyfile can be almost any file stored on a file server. In addition to regular data files, they can be files used by application programs, such as attachments within a mail client. For example, a company's executive email discloses corporate plans that will predictably affect the company's stock price. Such information could be extremely valuable to hackers. Security personnel can work with the executives to place honeyfiles within their mail clients. Honeyfiles can also be used to protect application programs. For example, a web-server's cgi-bin directory can be populated with empty honeyfiles named after notoriously vulnerable scripts.

- Honeyfiles show potential for avoiding some of the problems encountered by network intrusion-detection systems (NIDSs):

NIDSs are typically very weak at detecting unknown attacks, whereas honeyfiles can detect unknown attacks and even access gained through unintentional file-access permissions. Also, NIDSs can generate an exorbitant volume of false alarms. In contrast, honeyfiles show the potential for having a much lower false alarm rate. Further, with NIDSs, false alarms are often investigated by a centralized security group that does not work directly with the protected data, making investigation difficult. In contrast, honeyfile users can accurately and easily dismiss many false alarms because of their familiarity with the protected data.

Honeyfiles make it possible for alarms to be deployed by the personnel who create and manage information assets. In contrast, when NIDSs are deployed by a centralized security group, it can be difficult for them to accurately understand the network's changing information assets.

- The honeyfile system can be used to detect unauthorized access to real files, and it offers some substantial advantages over cryptographic checksums:

In addition to detecting access to deceptive honeyfiles, the honeyfile system can be used to detect access to real files. For example, when a workstation user leaves work for the day, he could use the honeyfile system to set alarms for all of his files.

The honeyfile system can be easily extended to provide alerts for honeyfiles when they are changed. A popular technique for detecting file changes involves creating and storing cryptographic checksums. The files' checksums are periodically recalculated to detect changes to the files. Tripwire is a commercial product that uses this checksum technique.

For detecting changed files on a file server, the extended honeyfile system provides two substantial improvements over the use of checksums: **1)** the honeyfile system detects changes when they occur, whereas the checksum technique detects changes during periodic, and often infrequent, execution, and **2)** the honeyfile system is simpler than the checksum technique. The honeyfile system resides on the file server and an end user only has to specify the honeyfiles. With the checksum technique, the checksums must be periodically calculated by the end user, or he must grant file access to a separate system that calculates the checksums. If the user calculates the checksums, he must securely store the binaries and checksums.

For a balanced assessment of checksums, it should be noted that checksums can protect local file systems, whereas honeyfiles can not. Also, the use of both checksums and honeyfiles can provide defense in depth for detecting file changes.

B. Tactical Capabilities

Honeyfiles' tactical capabilities stem mostly from: **1)** decentralized deployment: the network's end users create and place alarms, and **2)** centralized implementation: the alarm mechanism resides on the file server rather than on its clients.

- By enabling end-users to create alarms, the detection mechanisms can be effectively deployed and the alerts effectively interpreted:

If honeyfiles are created and placed well, it can be difficult for hackers to avoid them, resulting in a low false negative rate. End users are intimately familiar with the data they create and manage. Honeyfiles make it possible for users to create and place alarms where they are most needed and where they will be most effective. With some basic instruction on security and honeyfile tactics, users can effectively deploy honeyfiles. Also, end users can evaluate and improve their alarms' effectiveness because they receive alerts directly. Further, end users can adapt their honeyfile use as the network and its threats change.

Honeyfile users can accurately discern between true and false positives because they create the honeyfiles and receive the alerts. For instance, if a user accidentally opens a honeyfile, the resultant alert can be recognized as a false positive. If an alert is sent when the user is not accessing his file space, the alert can be recognized as a true positive.

- Honeyfiles support defense-in-depth for the file server's clients:

ISBN 555555555/$10.00 © 2003 IEEE

Honeyfiles provide the file server's clients with an alarm system that resides outside of the client itself, and this adds a layer of depth to the client's defenses. When a hacker breaks into a client, the honeyfile's alarm mechanism is on the file server, not on the client. If the honeyfile's alarm mechanism was on the client, the alarm would be vulnerable to attack or detection by the hacker, especially when the hacker has "rooted" the client computer. Alerts are sent by e-mail, and they can be made to travel over a secure channel.

- Honeyfiles can provide the security function of deterrence, and they can support incident response:

In addition to detecting attacks, honeyfiles can deter attacks. Honeyfiles have an affect that is similar to landmines: if hackers know honeyfiles are being used, the use can dissuade them from hacking, and the use can slow hackers down by making them cautious and uncertain. Honeyfiles are also useful for incident response. Investigators can view all of the alerts for a network, and collectively, they may reveal a hacker's capabilities, intentions, or courses of action.

C. Ease of Use

Honeyfiles' ease of use is advantageous to both end users and security administrators:

- Honeyfiles can enhance a network's internal security without impairing normal operations:

Networks typically use a relatively low level of internal security, as additional security is burdensome and costly. For example, extra access controls make resource sharing difficult, and making IDSs more sensitive increases their false alarm rates. Honeyfiles can provide a means of increasing internal security without impairing operations. Honeyfiles have little adverse affect on legitimate computer use. Also, honeyfiles can be an effective deterrent for insider hackers because they, like all other network users, will have been informed of the honeyfiles' availability and use.

- Honeyfiles are an effective deception because they can be easily created, require little falsehood, and involve little risk:

A honeyfile is integrated within a real file space, and this real context makes the honeyfile deception easy to create and difficult to detect. Also, honeyfiles themselves involve little falsehood—just a directory entry. Further, honeyfiles involve little risk.

- Implementing the alarm system on the file server makes honeyfiles available to almost all network computers:

Honeyfiles can be created by any computer that uses the file server. Honeyfiles can be used by computers with a wide variety of operating systems and file systems. The alarm system does not have to be ported to the network's various operating systems, e.g., Windows and Unix. Also, having a single alarm system makes it easier to train users.

- Implementing the alarm system on the file server centralizes security management functions:

Having a single alarm system makes the system's maintenance and defense easier, as the system resides in one place, rather than on each of the client computers. Further, having a single alarm system makes it easier for network security personnel to monitor the alarm system's overall use and effectiveness.

V. Enhanced functions

Earlier sections described basic honeyfile functions, and this section describes some enhancements that greatly improve honeyfile use. These improvements have to do with maintaining realism and controlling alarms.

Operational systems change over time, and so too must most honeyfiles if they are to be believable. A file's MAC times record when it was created, last modified, and last accessed. Honeyfiles that portray in-use files must have their MAC times periodically updated. The honeyfile system can solve this problem by periodically updating MAC times, within user-defined parameters.

If deceptive content is being used, it may also need to change over time. Although deceptive content is not addressed here, there is a noteworthy technique for automatically updating a file's deceptive content. The honeyfile's contents can mirror a source file that is hidden from the target, and the honeyfile system can periodically update the honeyfile from the source.

Honeyfile use can also be improved by providing controls for selectively generating alerts. Some processes and users must be permitted to open honeyfiles without setting off alarms, such as tape-backup processes and the root user.

VI. Honeyfile limitations

Honeyfiles' primary limitations are as follows:

- Honeyfiles may not be viable in file spaces that require regular searching:

Honeyfiles will not be viable if file search tools generate frequent and unavoidable false alarms. The honeyfile system could accommodate searches by enabling users to temporarily suspend their honeyfiles' alerts. However, a suspension function introduces vulnerabilities: users may forget to resume honeyfile alerts, and the function could be hacked.

- Honeyfiles are appropriate for file spaces that are accessible to one person or a small group:

Honeyfiles are likely to be problematic if placed in a file space that is used by many people. Honeyfile information would have to be communicated to the group. Also, false alarms may be frequent and difficult to investigate.

- Honeyfiles have tactical weaknesses that limit their use:

Like most deceptions, honeyfiles provide uncertain effectiveness against an individual attack. Many other security measures, such as strong encryption, are much more certain. Also, honeyfile use will be limited if the target does not tend to explore the file system.

There are some circumstances in which honeyfiles can be defeated. There are ways in which a hacker can identify real files, and if he opens only them, he will avoid honeyfiles. For example, a hacker can use a keystroke logger to learn what files are being used, and then open only them. Another honeyfile vulnerability is overloading of the alert mechanism.

- Honeyfiles require end-user involvement and skill:

Effective honeyfile deployment requires user participation. It cognitively taxes users by requiring them to manage and track honeyfiles. Also, it requires users to have some security savvy as well as adeptness with computers. Not all users will have the time or skills needed to use honeyfiles. However, security personnel can provide some simple training that will be sufficient for many users. Another potential cost of honeyfiles is the inadvertent deception of friendly personnel.

VII. Using a honeynet to test honeyfiles

The honeyfile system was tested by deploying it on the honeynet and thereby subjecting it to hacking. Three hacking incidents were observed, and each involved a different hacker. The three hackers were students from North Carolina State University who are skilled in computer security. The hackers accessed a honeyfile system containing error reports and manuals for a mainframe system. Each of the hackers was detected by at least one honeyfile. The hackers did not find the file space very interesting, and they did not search it diligently. This suggests that honeyfiles are more likely to be detected if they are near the file space's root, where the hacker will start searching.

Honeynets show much promise as a means for testing security devices. They provide a realistic setting in which hackers can test the device. The testing can be performed unwittingly by real hackers or by those recruited for the task. There is a significant advantage in using testers from outside of the security device's development team. Outside testers may reveal the developers' faulty and overlooked assumptions. Such errors are a common source of security vulnerabilities, and they are very difficult for developers to find themselves.

Building the honeynet was non-trivial and substantially more time consuming than we expected. Constructing deceptive files, file content, and system footprints (e.g., file time-stamps) was especially challenging, as all the falsehood had to be made consistent and believable.

VIII. Conclusion

After years of research and development, computer security remains an error-prone task and, in some respects, a loosing battle. Computer security's chronic problems call for wholly new approaches. Deception works in a fundamentally different way than conventional security. Conventional security tends to work directly with the hacker's actions, e.g., to prevent them or to detect them. Deception manipulates the hacker's thinking to make him act in a way that is advantageous to the defender. Being fundamentally different, deception can be strong where conventional security is weak. Honeyfiles are a promising tool for intrusion detection. They offer significant advantages where conventional intrusion detection is weak. A prototype honeyfile system has been constructed and tested, and we plan to distribute it as open source.

IX. References

[1] Stoll, C. *The cuckoo's egg*, Doubleday, 1989.

X. Authors

ISBN 555555555/$10.00 © 2003 IEEE

Jim Yuill is a PhD student in the Computer Science Department at North Carolina State University. This paper is part of his thesis research. Jim previously worked at IBM in operating systems development. *jimyuill-at-pobox.com*

Mike Zappe built the honeyfile prototype. He is currently a Unix kernel developer at Seclarity. *zapman-at-zappe.us*

Dr. Dorothy Denning is a Professor in the Department of Defense Analysis at the Naval Postgraduate School. She is an ACM Fellow, and the recipient of several awards, including the National Computer Systems Security Award. *dedennin-at-nps.navy.mil*

Fred Feer is retired from a career with the U.S. Army counterintelligence, CIA, RAND and independent consulting. Deception has been an interest and area of professional specialization for over 40 years. *ffeer-at-comcast.net*

ISBN 555555555/$10.00 © 2003 IEEE