2017 25th Euromicro International Conference on Parallel, Distributed and Network-Based Processing

A survey on fake entities as a method to detect and monitor malicious activity

Sampsa Rauti and Ville Leppänen
Department of Information Technology
University of Turku, Finland
Emails: {sjprau, ville.leppanen}@utu.fi

***Abstract*—This paper surveys research concentrating on fake entities as a method to detect and monitor malware. A fake entity is a digital entity (such as a file) no one except a malicious attacker should access. When the entity is accessed, the defender immediately knows there is unwanted activity in the system and can start to monitor it. We discuss both faking different entities on one machine and in a network using virtual groups of fake hosts.**

I. Introduction

Deception is an emerging and promising method to achieve software security, as has been clearly shown with honeypots [26]. Malware can be detected by placing *fake resources* within a computer or a network and its malicious activities can be monitored. The bogus resources are designed to seem as valuable as normal ones and the system waits malware to use these fake resources. A resource can be anything from a simple fake file to a bogus network service. More generally, we can talk about *fake entities*; for example, the attacker can also be fooled by setting up information on nonexistent people.

Stoll described the use of deceptive techniques to improve computer security already in 1989 [28]. Since then, the use of several different fake entities has been suggested in research. It is currently widely agreed that traditional security solutions are not enough to defend against sophisticated attackers [7]. This highlights the need for novel approaches.

The main contribution of this paper is to present a novel survey on how several different fake entities – from small entities residing on one machine to the large entities representing computer networks – could be used in order to detect, observe, or prevent these advanced attacks.

II. Deception and fake entities

A. Deception in computer systems

Current technology is not keeping pace with sophisticated threats like malicious insiders and advanced persistent threats (APT) [32]. As the traditional countermeasures have proven to be ineffective against advanced attackers in many cases, we need new mechanisms to defend the targeted systems. One way to prevent the adversary from reaching his or her goals is to use deception, that is, to employ mechanisms providing false cues that are given to the attacker.

Conventional security approaches often work directly against the adversary's actions in order to prevent them. Deception, in turn, manipulates the attacker's thinking with a goal of making him or her act in a way that benefits the defender. Being fundamentally different from traditional security approaches, deception is able compensate for conventional security's weaknesses. Combining the two is therefore likely to be advantageous from a security point of view.

Yuill [37] states that computer deception consists of *"planned actions taken to mislead attackers and to thereby cause them to take (or not take) specific actions that aid computer security defenses"*. Computer deception defined this way has several important properties and implications:

- *Increase the security proactively.* Introducing fake entities makes everything more difficult for a malicious program because it can no longer trust the deceptive environment. Also, it is not necessary to know the exact mechanisms the attacker uses, because the goal is not preventing malware from infiltrating the system. The attacker just needs to be confused with fake entities.
- *Monitor and analyze the attacker's actions.* It is naturally interesting to see what the attacker does in the system. This has been the goal of honeypots and intrusion detection systems for a long time. Many approaches we see in this paper just raise an alarm on suspicious activity, but it is also interesting to log the attacker behavior and learn from it.
- *Manipulate the attacker's actions.* Analyzing the attacker's actions and manipulating the environment perceived by a piece of malware leads to the possibility to manipulate its actions [3].
- *Waste the attacker's resources.* A great thing about many deceptive approaches is that they waste attacker's resources. For example, analyzing a fake reply from a server requires computational resources.
- *Mitigate the adverse effects and spreading of malware.* In a deceptive environment, changes made by malware may not have any effect and they can often be rolled back. It cannot reliably assess the changes it has caused in the system. Malware can also be lead to believe it has successfully spread while in reality this is not the case.

B. Fake entities

A fake entity can be anything we want the attackers to interact with. It can be a file, a database entry, or a password stored anywhere in the system. Fake entities come in many shapes and sizes: they can vary from a very small fallacious

2377-5750/17 $31.00 © 2017 IEEE
DOI 10.1109/PDP.2017.34

Fake entity	Type	Example
File	Content	A false business plan
File section	Content	A hyperlink in a file leading to bogus web page
Database	Content	A database full of false information on employees
Database record	Content	A fake patient in hospital's information system
Data in memory	Content	A false SSN in memory as plain text
File system	System data	
Metadata	System data	False file creation time
User account	System configuration	
Registry entry	System configuration	A registry entry for some specific application
OS interface	System configuration	A fake system call interface
Credentials	Personal data	A fake password
Person	Personal data	Fake people on a web page
Reply	Feedback/content	A fake HTTP reply with a bogus web page
Error	Feedback	
Service	System	A fake HTTP server
OS	System	A faked packet having characteristics of a specific OS's TCP/IP stack
Host	Network topology	A fake server storing an organization's data
Network	Network topology	Virtual fake intranet
Internet	Network topology	

Table 1. List of fake entities covered in this study.

piece of data to entire fake internet. Table 1 shows a list of fake entity categories that will be discussed more closely in this paper.

Fake entities have no authorized uses, interacting with them is always suspicious. Therefore they allow the defender to easily monitor malware. As an example, a fake file containing a fictitious business plan can be planted into the system. This entity has no real value and no authorized use. If an attacker tries to access this file, someone is most likely breaching organization's or user's privacy. When this happens, an alert will be generated. Therefore – in its most elementary forms – the fake entity approach is simple, no complicated algorithms or rules are needed. Generally, fake entities are far simpler and inexpensive than many other technologies. The fake entity approach is also one of the few existing methods to detect zero day exploits.

It is worth noting that here we do not actually concentrate on honeypot systems representing a single computer. This is because there are already several surveys on these kinds of honeypots [10]. Instead, our focus is on fake entities that can be seen as the elements that honeypot systems are made of. On the other hand, we also focus on entities larger than one computer such as fake networks.

III. Fake resources on one machine

Fake files

Bowen et al. automatically create decoys that are stored on a file system in order to entice a malicious user [9]. The decoy documents contain several different fake credentials that trigger an alert when used. The systems also keeps track of when and where a particular decoy was opened. Yuill et al. propose using bogus files as baits in order to detect malicious file accesses in [36].

Many of fake file approaches are just simple traps to detect and catch the attacker [34]. However, some approaches also aim at deceiving the adversary by introducing interesting fake content in the files.

Fake directories and file systems

If we can create fake files, it is only natural to generalize this idea and create fake directories and fake file systems [23]. Naturally, the files and file systems have to look interesting to the attacker.

The files shown in the directory listing do not always need to exist at all. We can give the adversary a fake error message with some generic excuse and inform the defender about the fact that suspicious activity is going on in the system.

Passwords and credentials

Juels and Rivest [16] aim to improve the security of hashed passwords by maintenance of additional honeywords, false passwords associated with each user's account. An attacker who succeeds in inverting the contents of password file cannot tell if a certain word is a real password or a honeyword. When the attacker attempts to use a honeyword, an alarm is set off.

Similarly, invalid information about e.g. user accounts [3] and credit cards [17] can also be stored in files. This kinds of little pieces of false information and sometimes also bigger entities such as honeyfiles are called *honeytokens* [26]. A honeytoken has to be enticing, something the hacker considers as valuable information.

Fake entities in databases

Honeytokens can also be used in databases. Much like fallacious pieces of data in files, honey tokens in databases are seen as traps that no one should normally touch [20].

Honey tokens can be effortlessly deployed in order to protect a wide variety of different database systems. It is deemed to be particularly useful when trying to detect privacy violations by employees inside the organization [20].

Memory

Honeytokens can also be placed into the memory [38]. Memory-scraping malware often looks for private information that is usually encrypted but is temporarily held in memory as plain text. Honey tokens are crafted so that they bear resemblance to the data malware is looking for. When a honey token is accessed in memory, an alert can be raised.

Fake metadata

In addition to the actual content (e.g. data in a file or database), metadata is also used by adversaries. This means we should lie about it as well. For instance, we can associate false creation and modification times with files and database entries to deceive the adversary. Unlike the actual content (like a convincing file containing business plans), this information is usually quite easy to create automatically.

As a practical example, Spafford [25] proposes a tool that takes a file and makes it seem really large with the help of the Unix sparse file structure. In reality, the file can be only few thousand bytes long on disk and it can be read in a normal fashion, but a copy program thinks the file size is several gigabytes in length. When the adversary tries to copy them off-site, this results to long copying that will take forever to complete. Meanwhile, it is possible to trace the adversary as the network connection stays open.

Fake registry keys and configurations

When a computer program is installed into a Windows machine, the program will also register in the Windows registry. Hoglund and Bracken suggest creating a fake registry key in the registry [13]. The idea is that this key is a substitute for the key a malicious program will later try to create when it installs. The piece of malware cannot create the key because it already exist and fails to function correctly. The same idea can be used with configuration files; a file can be created before malware creates it.

Honey patches

Araujo et al. [5], [6] propose transforming software security patches into honey-patches – patches that provide security equivalent to normal patches but rid attackers of ability to see whether their attacks have succeeded or failed. When an attempt to exploit a vulnerability is detected, the honey-patch redirects the adversary to an unpatched decoy. The attack is allowed to succeed against this decoy. Information on attacker's activities can then be collected.

Operating system interfaces

Besides memory, another way to lay a trap for malware on operating system level is the idea of changing the mapping of the system call numbers in the system. All trusted executables are modified to use these new system call numbers, but malicious program that does not know the new mapping still tries to use the original system call numbers and can be monitored.

We have presented a proof-of-concept implementation for this kind of dual-interface [18]. The very same idea of "fake original interfaces" could also be be applied in many different contexts – such as important operating system libraries (by renaming the functions, but leaving the original functions there in order to entrap malware) [19] and command shells (alter the command set of the the command shell, but leave original commands as a trap for the adversary) [29].

False errors

It is not always necessary to have a fake resource or return any real content to deceive the adversary. We can simply give a false reply to the attacker when he or she requests a resource [3]. There are many ways to deny the attacker's request by pretending some kind of error has happened [23]. The system can pretend the command given by the attacker has a syntax error. It can also claim that the requested resource is unavailable or that the command the attacker used was wrong for this resource. The adversary can also be given so much extraneous information or made wait so long that he or she gives up.

Of course, creating too many fake errors might raise the attacker's suspicions. Conversely, we could lie to the attacker that the action has succeeded even though this is not the case.

Systems combining many fake entities

In practice, many systems combine several of the fake entities we have discussed here. In what follows, we will present some examples of solutions involving various fake entities.

Wang et al. [33] present a multilayer deception system that aims to fool an attacker with several bogus entities. This includes honey files with regular honey activity. Network connections are also represented as honey servers and bogus network activity. These fake entities are integrated into a complete system.

Anagnotakis et al. [4] proposed shadow honeypots that are real applications but have some honeypot code integrated in them. Incoming requests are executed as usual but embedded honeypot functionality observes the request to decide whether they are malicious. Any actions performed by a malicious attacker are then rolled back.

Finally, it is interesting to note that because the attackers today generally aim to avoid honeypots, it is actually possible to keep many attackers away from a system by making the system pretend it is a honeypot [22]. These kinds of systems are called *fake honeypots*.

IV. Bogus networks and fake entities in a network

Fake networks

Simulating networks and the services hosts provide for others in a network has also received attention. Alberdi et al. [1] present a solution that deceives bots in a botnet. It is a redirection kit that redirects any outgoing attacks – e.g. the messages bots send to each other to coordinate attacks – to other honeypots so that malicious attacks to real servers through honeypots are prevented. The bot owners will still think that the bots communicate with computers outside the network even though they are in fact sending messages to a honeypot in the same network.

Handling the outbound connections initiated by the attacker is not an easy problem. In other words, how do we present a convincing fake network or fake internet to the adversary. The attacker might notice that the connection is being faked if the reply is not what he or she expects.

Honeyd is a tool used to create a fake internet topology [14]. This small open source daemon creates a group of virtual hosts that appear to run on unallocated addresses of a network. The personality of the hosts can be configured so that they appear to be running specific operating systems. Arbitrary services can be run on the created hosts. A single host can

also use multiple network addresses. Similarly to Honeyd, the Honeynet Project also enables building virtual honeynets [15].

Net-Chaff [37] impersonates computers at the unused addresses of a network. The goal is to detect and stop the attacker's port scans. When the tool detects a scan, it prevents malware's access to the network using the routers in the intranet.

For most approaches using fake entities in a network, the aim seems to be to monitor the adversary's activities and defend the network against threats. By doing this, the potential attackers are lured away from critical systems.

Operating systems

Operating systems are important potential fake resources. As mentioned before, Honeyd can impersonate different operating systems. Honeyd can imitate the appearance of over 1,000 operating systems and their variants [35]. It uses knowledge from an extensive database in order to mimic fingerprints of a wide variety of operating systems. Every operating system has a unique TCP/IP stack fingerprint, and each stack has a different response [30]. When these fingerprints are simulated, the attacker can be fooled.

Deceptive services

Traditional computer deception, such as honeypots, often fool the adversary by creating fake copies of production systems and trying to lure adversaries to them. However, it is also possible to deceptively change replies of a public-facing service (e.g. a HTTP server) without actually having any fake resources in the system. Deceptiver [2] applies deception to resources either by creating a new deceptive response on the fly or by modifying the response (possibly containing system's resources) on the fly before it is sent to the attacker. A proof-of-concept has been implemented for an Apache web server.

Fake services can be set up in several virtual hosts. For instance, the pieces of malware often need to communicate via DNS, HTTP and SMTP [27]. In many cases, emulating only part of the protocol is enough for malware to function normally. It is also worth noting that we often do not know beforehand what protocol a malicious program is going to use, or a custom protocol written by malware author can be employed. In these cases, we can simply simulate a generic protocol (by for example listening on a TCP port and sending a reply to each received packet) and try to draw out as much conversation from the malicious party as we can.

Honey people

Information on nonexistent people, *honey people*, can be set up for instance in a social network [31], [33]. These fake persons are created to appear realistic and have connections with other people inside and outside their organization. Fake email addresses can also be created for these people [21]. Interaction with these fake persons is then monitored and an alarm can be set off when they are contacted.

V. Challenges

Fake entities are a simple and inexpensive method to detect and study malware, but we have already seen there are some challenges with this approach. Naturally, fake entities require some space and can also have some effects to the performance of the system. Still, these effects are not usually significant.

There is also a problem of false positives when an employee unintentionally opens a honey file. No one else except the attacker should interact with it in order to avoid false positives. However, there are many cases (e.g. for privacy-critical databases) where even insiders are not allowed to access some entities, and fake entities are a good method to detect this kind of data breaches.

The main problem with fake entities is arguably creating them. Naturally, creating data content that appears genuine and is difficult to distinguish from real entities is one of the main challenges when doing this. Bercovitch et al. [8] present HoneyGen, a method for automatic generation of fake entities. It extrapolates the characteristics of real data items. Still, creating large files or email messages with convincing contents automatically is obviously difficult. On the other hand, convincing fake files created by humans would require lots of work.

We have also seen – in context of fake networks and fake internet – that granting malware uncontrolled access to the Internet is not a good idea [39]. There is always a danger of malicious network traffic and self-spreading worms. We have already seen some solutions to this problem presented in literature. Moreover, we believe the convincing exchange of messages between malware and fake entities (such as fake services) should be further studied. This would better convince malware that it is indeed communicating with a genuine entity and elicit more interesting activity from the malicious programs.

VI. Conclusions and future work

We have seen that there are many interesting ideas on fake entities in the literature. A fake entity is a very simple and flexible concept that can be applied to many assets in computer systems and networks. Fake entities are also inexpensive, easy to deploy and lightweight compared to many other security approaches.

Still, in our view, the answer to the question "what should we do after malware is detected" is often incomplete or wrong. In future work, we plan to study "record and play" approaches where we deceptively interact with malware by using earlier genuine interaction in some service as a basis [12]. This poses a problem of data creation: we have to create convincing fake data to fool the malware. If this is created based on the genuine data (which would be preferable, of course), we have to be careful so that we do not disclose anything secret or important to the attacker. Still, it would be enticing to further study how machine learning approaches could be used to create the data used by fake entities.

In practice, it would be interesting to build a proof-of-concept implementation of our framework and test it. Exper-

iments could be performed on some relatively simple service to demonstrate the feasibility of deceiving a malicious piece of code while exchanging messages with it with the goal of keeping the fake communication convincing as long as possible so that we can learn more and more about the attacker's malicious actions.

As mentioned previously, we believe several operating system interfaces also have lots of potential as fake entities. The idea of changing the system call numbers, library functions or shell language command sets and leaving the "fake original interface" as a decoy for malware could be developed further.

As malware keeps increasing at a staggering pace and advanced persistent threats and insider attacks pose a serious challenge, novel approaches are needed. We believe fake entities have lots of potential and room for further development in this field of research.

ACKNOWLEDGMENT

The authors gratefully acknowledge Tekes – the Finnish Funding Agency for Innovation, DIMECC Oy and Cyber Trust research program for their support.

REFERENCES

[1] Alberdi, I., Philippe, E., Vincent, O., Nicomette, K.M. Shark: Spy Honeypot with Advanced Redirection Kit, Proceedings of the IEEE Workshop on Monitoring, Attack Detection and Mitigation. IEEE, 2007.

[2] Almeshekah, M.H. Using Deception to Enhance Security: A Taxonomy, Model, and Novel Uses. Dissertation. Purdue University, 2015.

[3] Almeshekah, M.H., Spafford, E.G. Planning and Integrating Deception into Computer Security Defenses, Proceedings of the 2014 workshop on New Security Paradigms Workshop, pp. 127–138. ACM, 2014.

[4] Anagnostakis, K.G., Sidiroglou, S., Akritidis, P., Xinidis, K., Markatos, E., Keromytis, A.D. Detecting Targeted Attacks Using Shadow Honeypots, Proceedings of the Conference on USENIX Security Symposium, pp. 9–23. The USENIX Association, 2005.

[5] Araujo, F., Hamlen, K. W., Biedermann, S., Katzenbeisser, S. From patches to honey-patches: Lightweight attacker misdirection, deception, and disinformation. In Proc. ACM Conf. Computer and Communications Security (CCS), pp. 942–953. ACM, 2014.

[6] Araujo, F., Shapouri, M., Pandey, S., Hamlen, K. Experiences with honeypatching in active cyber security education. In 8th Workshop on Cyber Security Experimentation and Test (CSET 15), 2015.

[7] Bejtlich, R. The Practice of Network Security Monitoring: Understanding Incident Detection and Response. No Starch Press, 2013.

[8] Bercovitch, M., Renford, M., Hasson, L., Shabtai, A., Rokach, L., Elovici, Y. HoneyGen: An Automated Honeytokens Generator. IEEE International Conference on In Intelligence and Security Informatics (ISI), pp. 131–136. IEEE, 2011.

[9] Bowen, B., Hershkop, S., Keromytis A.D., Stolfo S.J. Baiting Inside Attackers Using Decoy Documents. Security and Privacy in Communication Networks. Volume 19 of the series Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 51–70. Springer, 2009.

[10] Bringer, M.L., Chelmecki C.A., Fujinoki, H. A Survey: Recent Advances and Future Trends in Honeypot Research. International Journal of Computer Network and Information Security, Volume 4, Issue 10, pp. 63–75. MECS, 2012.

[11] Cheswick, W.R. An evening with Berferd, in which a cracker is lured, endured, and studied. In Proceedings of the USENIX. The USENIX Association, 1992.

[12] Cui, W., Paxson, V., Weaver, N., Katz, R.H. Protocol-independent adaptive replay of application dialog. Proceedings of the 13th Annual Network and Distributed System Security Symposium, 2006.

[13] Hoglund, M.G., Bracken S.M. Inoculator and antibody for computer security. Patent US 20120110673 A1.

[14] Honeyd homepage. http://www.honeyd.org

[15] Honeynet homepage. http://www.honeynet.org

[16] Juels, A., Rivest, R.L. Honeywords: Making passwordcracking detectable. In Proceedings of ACM CCS, pp. 145–160, 2013.

[17] Kambow, N., Passi, L.K. Honeypots: The Need of Network Security. International Journal of Computer Science and Information Technologies, Volume 5, Issue 5, 2014.

[18] Lauren, S., Rauti, S. Leppänen, V. An Interface Diversified Honeypot for Malware Analysis. Accepted to MeSSa 2016.

[19] Lauren, S., Mäki, P., Rauti, S., Hosseinzadeh, S., Hyrynsalmi, S., Leppänen. V. Symbol Diversification of Linux Binaries. In Proceedings of World Congress on Internet Security (WorldCIS-2014). IEEE, 2014.

[20] Rietta, F.S. Application layer intrusion detection for SQL injection. In: Annual Southeast Regional Conference, Melbourne, Florida, pp. 531–536. ACM, New York (2006).

[21] Rowe, N.C. Deception in defence of computer systems from cyber-attack. In: Colarik, A., Janczewski L. (Eds.): Cyber War and Cyber Terrorism, pp. 97-104. The Idea Group, 2007.

[22] Rowe, N.C., Duong B. T., Custy, E. J. Fake Honeypots: A Defensive Tactic for Cyberspace. IEEE Information Assurance Workshop, pp. 223–230. IEEE, 2006.

[23] Rowe, N.C. A model of deception during cyber-attacks on information systems. First Symposium on Multi-Agent Security and Survivability, pp. 21–30. IEEE, 2004.

[24] Small, S., Mason, J., Monrose, F., Provos, M., Stubblefield, A. To Catch a Predator: A Natural Language Approach for Eliciting Malicious Payloads. USENIX Security Symposium, 171–184. The USENIX Association, 2008.

[25] Spafford, E. More than passive defense. https://www.cerias.purdue.edu/site/blog/post/more_than_passive_defense/

[26] Spitzner, L.: Honeypots: Tracking Hackers, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 2002.

[27] Stewart, J. Behavioural malware analysis using Sandnets. Computer Fraud and Security, Volume 2006, Issue 12, pp. 4-–6. Elsevier, 2006.

[28] Stoll, C.P. The Cuckoo's Egg: Tracing a Spy trough the Maze of Computer Espionage. Doubleday, 1989.

[29] Uitto, J., Rauti, S., Mäkelä, J.-M., Leppänen, V. Preventing Malicious Attacks by Diversifying Linux Shell Commands. In Proceedings of the 14th Symposium on Programming Languages and Software Tools (SPLST'15). CEUR Workshop Proceedings 1525. CEUR, 2015.

[30] Valli, C. Honeyd - a fingerprint artifice. Paper presented at the 1st Australian Computer, Information and Network Forensics Conference, Scarborough, Western Australia, 2003.

[31] Virvilis, N., Serrano, O. Changing the game: The art of deceiving sophisticated attackers. In Proceedings of the 6th International Conference on Cyber Conflict (CYCON-014), pp. 87–97. IEEE, 2014.

[32] Virvilis, N., Gritzalis, D. The Big Four – What we did wrong in Advanced Persistent Threat detection? In Eighth International Conference on Availability, Reliability and Security (ARES), pp. 248–254. IEEE, 2013.

[33] Wang, W., Bickford, J, Murynets, I., Subbaraman, R., Forte, A., Singaraju, G. Catching a wily hacker: A Multilayer Deception System. Proceedings of 35th IEEE Safnoff Symposium, pp. 1–6. IEEE, 2012.

[34] Whitham, B. Canary Files: Generating Fake Files to Detect Critical Data Loss from Complex Computer Networks. Presented at the Second International Conference on Cyber Security, Cyber Peacefare and Digital Forensic (CyberSec2013), 2013.

[35] Willis, N. Weekend Project: Use Honeyd on Linux to Fool Attackers. The Linux Foundation, 2011. https://www.linux.com/learn/weekend-project-use-honeyd-linux-fool-attackers

[36] Yuill, Y., Zappe, M., Denning, D., Feer, F. Honeyfiles: deceptive files for intrusion detection, in Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC, pp. 116–122. IEEE, 2004.

[37] Yuill, J. Defensive Computer-Security Deception Operations: Processes, Principles and Techniques. Dissertation. Computer Science, North Carolina State University, 2006.

[38] Zeltser, L. Detecting memory-scraping malware. Patent US 20150381655 A1.

[39] Zhang, F., Zhou, S., Qin, Z., Liu, J. Honeypot: a supplemented active defense system for network security. Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'2003), pp. 231–235. IEEE, 2003.