A Forensic Framework for Tracing Phishers

Sebastian Gajek[1] and Ahmad-Reza Sadeghi[2]

Horst Görtz Institute for IT-Security
Ruhr University Bochum, Germany
sebastian.gajek@nds.rub.de[1], sadeghi@crypto.rub.de[2]

Abstract. Identity theft – in particular through phishing – has become a major threat to privacy and a valuable means for (organized) cybercrime. In this paper, we propose a forensic framework that allows for profiling and tracing of the agents involved in phishing networks. The key idea is to apply phishing methods against phishing agents. In order to profile and trace phishers, their databases are filled with fingerprinted credentials (indistinguishable from real ones) whose deployment lures phishers to a fake system that simulates the original service.

1 Introduction

With the advent of the Internet and the continuous growth of electronic commerce, offering new commercial prospects and opportunities, criminals have also discovered the Internet as a ground for illicit business. Cybercrime has entered the digital world and prospered to a severe concern for Internet users. Digital crimes appears in many variations, however, online fraud has proliferated recently and become a major threat. A prominent method of fraud is phishing where, e.g., Internet users are lured to faked web sites and tricked to disclose sensitive credentials such as credit card numbers and passwords. The attacker steals these credentials in order to gain access to users' accounts and services. Federal law enforcement link organized crime to phishing attacks that involve various actors a part of a highly professional criminal network, and that deploy methods similar to those of money laundering. There is not much publicly known about these criminal networks, however, their impact is noticeable.

Phishing has grown to a lucrative criminal business model. A report issued by the Gartner Group [10] totals financial losses to $2.8 billion in 2006 whereas in previous studies Gartner approximated total losses to $1 billion [9, 10]. In comparison to identity theft in real world, phishing attacks are cheap and the potential gains are huge. A standard approach to counter phishing attacks apart from user education is to incriminate and isolate rogue (phishing) servers [13]. In general, the hosting provider is contacted and asked for taking down the site. As most rogue servers are hosted in foreign countries, it is time-consuming to make on-site forensic analysis, and to initiate criminal prosecution. Specifically, different international regulations delay rapid shutdowns, giving phishers enough time to steal vast numbers of identities. Thwarting phishing attacks has become a game of one-upmanship. That is, when a rogue site is taken down, phishers

move to another server. The introduction of countermeasures on client side such as blacklisting phishing mails [7] or improved user authentication schemes have failed to alleviate the threat. Phishers modified the attack vector when new measures have been applied, using a large portfolio of social engineering techniques, mounting and illusion attacks (cf. [1]). Hence, a general belief is that efficiency of countermeasures may be measured in terms of increased costs and risks to mount the attack.

In this paper, we present and discuss aspects of applying methods of phishing against the agents involved in phishing attacks. We propose a forensic framework and discuss the related forensic information sources that can support investigators and service providers to pro-actively fight phishing by identifying and tracing the involved actors. The key idea is to fill phishers' credential databases with fingerprinted credentials (dubbed *phoneytokens*). Upon using phoneytokens, phishers are lured to a fake system (dubbed *phoneypot*). The phoneypot simulates the original service in order to profile phishers' behavior. This approach deters criminals from harvesting valid credentials and increases the risks to track phishers. Moreover, our proposal has much broader coverage since it makes first steps towards the design and realization of an architecture that is interoperable to commodity web applications.

The remainder sections are oganized as follows. Section 2 summarizes the basic idea. Section 3 discusses related work and Section 4 presents an idea for a concrete solution. Section 5 discusses the aggregation of profiles. Section 6 puts into perspective measures which phishers could use to thwart our approach. Finally, Section 7 concludes the paper.

2 Preliminaries

2.1 Model

The main goal of phishing attacks is to gain access to financial assets through identity theft. For this purpose, phishers make use of spoofing attacks to trick users into disclosing their credentials. Phishers use the stolen credentials to impersonate users, and deploy strategies similar to those of money laundering in order to transfer the (electronic) financial assets to the real world. Phishing generally involves several collaborating agents as illustrated in Fig. 2.1.

Assume a customer U uses a credential A to get access to service S (step 1). S may be a financial institute (e.g., a bank) that allows U to make money transfers to other accounts (step 2). We define the following agents in a phishing attack.[1]: *Mounting attack agents* provide the technology required to lure Internet users to collection servers (e.g., by impersonating the bank in spam emails that contain links to a fake website). *Hosting attack agents* provide the technology for tricking Internet users to disclose their private information (e.g., by designing phishing sites or implementing malware). One main task of hosting attack agents is to hijack vulnerable web servers and turn them into *collection servers* that store the

[1]Note that in some instances, one actor may play several roles.

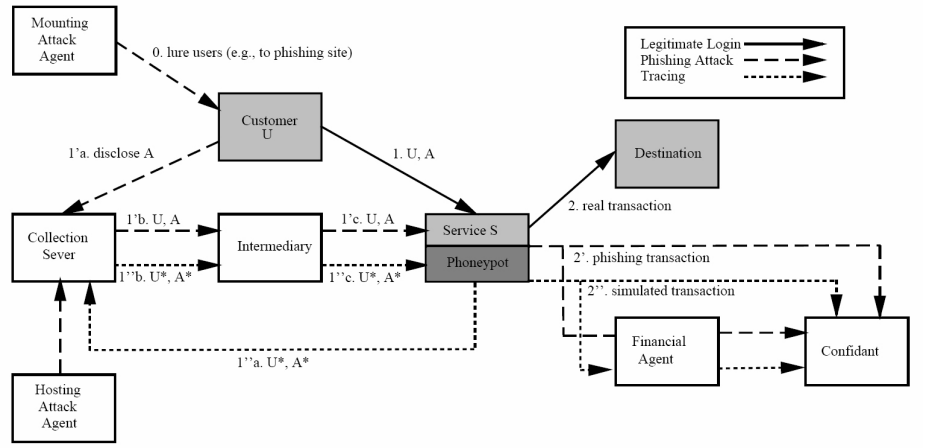

Fig. 1. Tracing the Agents of Phishing

stolen credentials of deceived users. Assuming U has been fooled by the mounting attack agent, and has disclosed credential A, then A is stored in the collection server (step 1'a). The stolen credential is then forwarded to an *intermediary* who is responsible for exchanging or moving financial assets through various entities in order to conceal their true origin and owner (step 1'b). For instance, the intermediary performs transactions on behalf of deceived customers (step 1'c). Transactions are made either to foreign accounts or to *financial agents* whose aim is to hide the illicit origin (step 2'). This is also known as *money layering*. Finally, *confidants* arrange the withdrawal of the money at the transaction's destination.

2.2 Basic Idea of our Approach

The objective of our approach is to deter phishing attacks by enabling the tracing of phishing agents and their associated criminal network, as well as to increase the cost and risks of phishing attacks. We concentrate on collecting information on intermediaries and financial agents that can then be provided to law enforcement for criminal prosecution. By feeding false credentials to phishers' collection server we increase the cost of the attack in two ways. First, the fact that their databases are filled with fake credentials means that phishers have to spend more time and resources in order to get some financial benefit; and second, the risk of a phishing attack is increased due to the tracing capabilities of our scheme.

The basic idea of fingerprinting credentials is analogous to using the serial numbers of bank notes or any other fingerprint to trace the circulation of money. We feed fingerprinted credentials which we call *phoneytokens* (*"phishing honeytokens"*), to the collection servers. A Phoneytoken is the application of

honeytoken [18] to phishing, and represents some data that looks like a valid credential to the phisher, but is traced. To instantiate the phoneytoken approach, one has to detect active collection servers. This can be done by today's known methods such as bounces of phishing mails, alerts of phishing report networks[2], or reverse-engineering of phishing malware. After having sent different phoneytokens to the collection server, one can initiate the server's shutdown (as it is currently practiced). Alternatively, service providers such as financial institutes, may equip customers with phoneytoken credentials to be used when they detect a fake service. For user convenience, service providers may extend anti-phishing toolbars used to warn users when they are visiting a phishing site to submit phoneytokens.

Once we have identified a rogue web site used for phishing, we proceed as follows (see Fig. 2.1): We send phoneytokens to that site and thus insert a set of fake credentials at the collection server. We denote a fake credential with A^* (step 1"a). When the phisher—or more precisely the intermediary—harvests the credentials (step 1"b) and at some stage reuses A^* to get access to our service on behalf of a fake user U^* (step 1"c), we can distinguish U^* from a legitimate customer U and identify U^* as phisher. We then trap the phisher to a virtual system. We call the virtual system a *phoneypot* (*"phishing honeypot"*). The phoneypot simulates the original system and cherishes the illusion that phishers perform valid transactions on a real account; in fact, the phoneypot collects information about phishers' networking, browsing and service-specific behavior. Technically speaking, a phoneypot adapts the notion of honeypots to the problem of phishing; the data collected consists then of attributes typical for a certain intermediary. When the itermediary performs a transaction, the phoneypot simulates the service (step 2") in order for tracing the involved criminal network.

Based on the collected characteristics (see Section 5), we derive three classes of phishing profiles, namely, the *non-phisher*, the *definite phisher*, and the *potential phisher*. The non-phisher profile characterizes an honest user who legitimately authenticates to the service and accesses the real system. The definite phisher is unambiguously identified as an adversary according to the use of phoneytokens and is relayed to the phoneypot. The potential phisher is assumed to be an adversary according to some similarity to a definite phisher. Depending on the degree of similarity, the potential phisher is either relayed to the phoneypot or we delay the transaction and request for authorization (e.g., we call the account owner to confirm that transaction).

2.3 Assumptions

The primary difficulty in our approach is how accurately the phoneypot learns profile patterns. This ultimately affects the accuracy of the phoneypot framework both in terms of whether a non-phisher is flagged (false positive) and whether a phisher will be missed (false negative). In order to reduce false detections, we

[2] e.g., `http://www.phishreport.net/`

assume that some profiles have specific low-biased attributes that we apply to train the phoneypot:

1. The identification of definite phishers is accurate due to the use of phoneytokens. This enables us to use alternative fingerprinting techniques that enlarge the traceability of potential phishers.
2. Non-phishers are conservative regarding the use (and change) of security critical services. For instance, Internet consumers use banking services at a certain time of day and from a certain location. We get a false detection of a non-phisher, if the user changes usual behavior (e.g., requesting access to a service from abroad). However, we expect this behavior to rarely happen.
3. As with real-world organized crime, phishers change their strategy when the costs and risks of attacks increase; otherwise they keep and maintain the techniques and actors. Especially potential phishers address same intermediaries and financial agents since the number of agents is limited.

3 Related Work

Chandrasekaran *et al.* [5] use fake credentials which are submitted to phishing sites and may be seen as phoneytokens. The authors' key idea is to detect phishing sites according to the response of fake input. In contrast, our approach uses forensic methods to identify the involved actors and already assumes phishing sites to be identified. We use fake credentials in order to observe phishers' behavior.

McRae, McGrew, and Vaughn [11] make use of a "web bug", that is a specific kind of phoneytoken: The authors fill forms with HTML code, rather than filling out a phishing site's form with real or faked credentials. The HTML code will cause a web browser or email client to retrieve additional information in order to render the code. The request allows for forensic analysis. This approach is limited by the assumption that phishers render the HTML code. Most email clients prohibit the download of such code, and sophisticated phishers filter the information filled out. Hence, the use of web bugs may easily be foiled. We use phoneytokens that are indistinguishable to real credentials in order to avoid being detected by phishers.

The authors of the honeynet project [20] report on experiences with hosting a honeypot attacked by phishers. The goal of this project was to learn the strategies phishers deploy to mount a phishing attack. The project uses the standard definition of honeypots which is to deposit a "weak" system and wait until an intruder is attracted. This approach collects forensic information on techniques used by phishers and is aimed for thwarting on the technical part of phishing (e.g., setting up phishing sites, sending spam emails, controlling spam botnets). Our approach is different. We collect forensic information in order to trace the usage of stolen (and fingerprinted) credentials. By contrast, we are interested in tracing the phisher's agents and not in the technical means used by phishers.

Some financial institutes do fraud auditing which is closely related to our approach (see, e.g., [6]). However, fraud auditing is in general a post hoc method,

i.e., after fraud has occurred investigations are initiated. Our approach is proactive. It helps to expose criminals without apriori knowledge of malicious activities. Further, our approach bears no financial risks, i.e., no customer becomes a victim of fraud. Hence, the presented approach may be used to complement existing fraud auditing mechanisms.

4 Framework

Fig. 2 illustrates our proposed framework in conjunction with a real system. The framework comprises two main components: The phoneypot that simulates the real system to infer phishing profiles and a phoneytoken machine that generates and distributes phoneytokens.

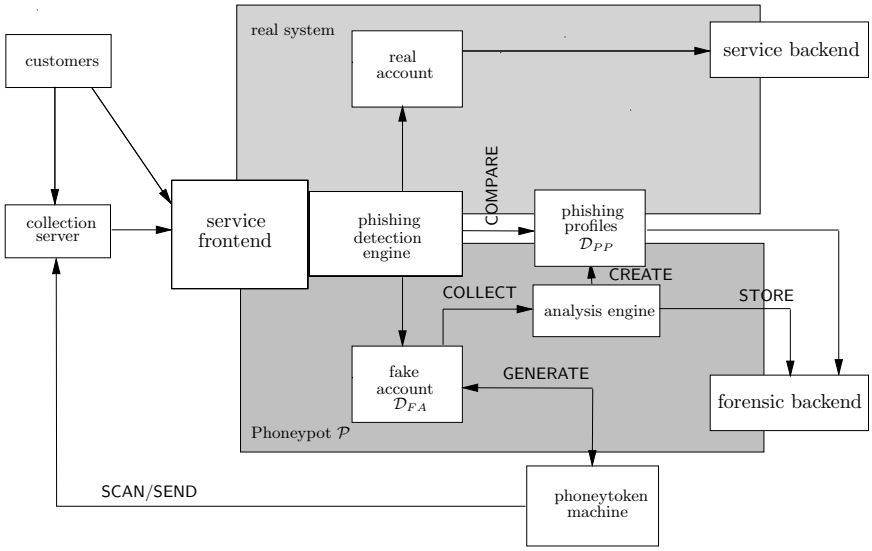

Fig. 2. The Architecture of a Phoneypot

4.1 Phoneytoken Machine

The interfaces SCAN, GENERATE and SEND constitute the phoneytoken machine. We use SCAN to analyze and extract the attributes required to fill a collection server with phoneytokens. We apply web crawling and form scanning techniques to extract attributes from phishing sites (e.g., [14]); in case of modern web technologies (e.g., Flash) or malware, an analyst has to manually look for the attributes. Typically, the attributes requested are email address, username, password and some service specific credentials, such as credit card number, PIN and

TAN or social number. Let $\mathcal{C}$ be the set of attributes requested by a collection server denoted by the address URL, then SCAN generates a new table in fake account database $\mathcal{D}_{FA} := (URL, \mathcal{C})$. We use GENERATE to create values for a fake account which ordinary users would normally submit (to a collection server). The phisher is unable to distinguish between user and machine-generated values. We call the entry of $\mathcal{D}_{FA}$ a *phoneytoken* $\mathcal{T}$ where $c_1, c_2, ..., c_n \in \mathcal{C}$ are the n attributes requested by a collection server URL.

In GENERATE we create random values of dictionaries or public lists for alphanumerical values, such as names, streets, cities. For (sufficiently long and randomly looking) numerical values, such as PIN, TAN, bank account number, we conceal sequence number and submission date to trace the time until the phisher applies this phoneytoken. The jitter time may be used to determine the submission frequency of phoneytokens. If f_k is a function that conceals the sequence number seq and the date of submission $date$, and k a secret key linked to a certain phoneytoken, then numerical values are calculated as concatenation of $date$ and seq. For example, let TAN be an integer, then $TAN = f_k(date \| seq)$. There are many possibilities to instantiate the function f (see e.g., [12]). The secret k also authenticates a phoneytoken $\mathcal{T}$. This prevents a phisher from creating a phoneytoken $\mathcal{T}^*$ and using $\mathcal{T}^*$ to fool the forensics. A further effect is that we do not have to store more information in a database where, e.g., only username and password pairs are queried in order to detect a phoney user. GENERATE also creates additional entries for each phoneytoken $\mathcal{T}$ that are mandatory to build up a fake user account (cf. Section 4.2).

We use SEND to submit the phoneytokens $\mathcal{T}$ to the collection server. The phisher should be unable to notice that $\mathcal{T}$ has not been submitted by phishing victims, as an unusual collection of phoneytokens may raise suspicion. Therefore, we require that SEND emulates the submission behavior of legitimate customers. We meet this requirement by inferring typical behavior profiles from log files of the real system (service frontend) and by sending the phoneytokens from different networks. To the authors' knowledge, service providers that are currently target to phishing attacks, especially financial institutes, have the capabilities to provide the phoneytoken machine with different IP addresses, allowing to send numerous phoneytokens from several addresses.

4.2 Phoneypot

The design goal of the presented architecture is to separate the real web application from the phoneypot. We isolate the phoneypot in order to ensure that our analysis do not negatively impact the original application. This design goal makes our approach appealing for high-performance and security critical applications where modifications to the backend are crucial (e.g., financial institutes). The phoneypot is coupled with a phishing detection engine to the service fontend of the web application. We integrate the phishing detection engine into the login and authentication mechanisms of the web application in order to recognize users logging in using a phoneytoken. The phishing detection engine then redirects this

session to the phoneypot. This can be easily achieved with *load-balancing hardware*[3]

The phoneypot needs to exactly replicate the functionality and the look and feel of the real web application; otherwise phishers may raise suspicion. We call this a fake account. This seems at first an involved task, particularly because the modeling of the real web application has to be redone for every application which should be accompanied by a phoneypot. Fortunately, most security critical web applications have only very few interactive and dynamic elements as a result of non-interoperable browsers (e.g., the web application has to be completely renderable without active content). Therefore, most web sites are purely static and can simply be copied to the phoneypot. Requests for dynamic content is to be proxied by the phoneypot, however, with the slight difference that in order to complete the simulation of a fake user account (e.g., account balance, list of last activities) additional data is derived from the fake account database $\mathcal{D}_{FA}$.

For profiling, we log all traffic to and from the phoneypot. We use STORE to pass the data to the *forensic backend* which stores the data in a well documented and secure manner to meet with forensic standards (e.g., as specified by Scientific Working Group on Digital Evidence (SWGDE)[4]). Also, all components involved in evidence preservation must be thoroughly audited by independent parties.

5 Phishing Profiles

5.1 Data Collection

We use the interface COLLECT to collect information on phishers. This information is passed to the *analysis engine*, where we infer profiles of phishers. The profiles are stored in the *phishing profile database* $\mathcal{D}_{PP}$, using CREATE. To build our profiling information, we terminate the communication between the phisher and the phoneypot. We then have access to various link and application characteristics provided by ISO/OSI layers that we take into account to build phishing profiles. We apply standard passive fingerprinting techniques [21, 3, 17, 8]; we do not use active fingerprinting techniques to avoid being detected by phishers. Then we may fingerprint the phisher's geolocation, operating system, browsing application and user behavior.

In addition, we have access to characteristics provided by the service. Whenever a user—be it a phisher or be it an honest costumer—makes use of the service (e.g., make a donation), one can profile that behavior. As phishers prefer services that are appealing to clean digital goods into real funds, such as online-banking or auctions, we gain information dealing with aspects of money laundering (see e.g., [4]). For instance, in case of online banking, one can intercept the account

[3]Load balancers spread work between many computers, processes, hard disks or other resources in order to get optimal resource utilization and decrease computing time. The phishing detection engine instructs the load-balancer to transfer the session to the phoneypot.

[4]`http://ncfs.org/swgde/index.html`

number, amount due, the date and the recipient of an illicit transaction, where in case of online auctions, one can collect information about goods purchased or sold, contacted persons and messages exchanged using the internal instant messaging functionality, or the methods used to pay for the auction.

5.2 Profile Aggregation

In order to build a phishing profile, we need to find the distinguishing characteristics that allow us to distinguish a normal user from a phisher. Let $\mathcal{P}$ be the set of attributes stored in the phishing profile database $\mathcal{D}_{PP}$ that identifies known phishers. $\mathcal{P}$ includes both transaction information (e.g., destination accounts of financial agents and confidants), as well as network and application information.

Let $\mathcal{A}$ be the set of attributes that we infer on the customer U—be it a phisher or a legitimate user—who accesses the service S, and let $\Pi_i \in \mathcal{A}$ be a certain characteristic, then we use a simple rule as follows in order to instantiate the COMPARE function:

> if U uses a phoneytoken $\mathcal{T}$, or if the remaining attributes $\mathcal{A} \setminus \{\mathcal{T}\}$ match the attributes $\mathcal{P}$ of one of the profiles, then we say U is a definite phisher.

As a phoneytoken is constructed such that $\mathcal{T}$ is indistinguishable from a real token, we can be sure that we unambiguously expose a phisher. Hence, the rule can be used to collect viable profile information on the phisher. This allows us to use supervised methods to infer profiles of potential phishers. Supervised methods are those where attributes of both definite and potentials phishers are present. Several techniques have been proposed in the literature for the construction of user profiles in different applications, such as intrusion detection, credit card fraud, telecommunications fraud based on statistical modeling like rule discovery, clustering, Bayesian rules, neuronal network classification, etc. (see e.g., [6] for an overview).

Similarly, we have a profile of each user U, containing information on her past behavior. If the attributes of the transaction match those of the user profile (e.g., money is transferred to a destination account already known by that user), the transaction is assumed to be legitimate. Note that, if U claims that a given transaction has not been performed by herself, the attributes of that transaction will be added to $\mathcal{D}_{PP}$. This would still leave a gray area of transactions with attributes that neither identify the user as a definite phisher, nor correspond to known behavioral patterns of the owner of the credentials that are being used. We note that, given the high volume of transactions processed by financial institutions, it is very important that the phishing detection engine is efficient in telling apart legitimate from illegitimate transactions.

Let us assume that we use the information collected from a given user U to define the probabilities $\Pr(\Pi_i|U)$ of U performing a transaction with attribute values $\{\Pi_i\}$. Note that these probabilities will vary for each user; for example, a user U_1 who travels often and makes transactions from different locations is much more likely to initiate a transaction from a new location than another

user U_2 who has always made transactions from the same computer at her home address.

Analogously, we can define the probabilities $\Pr(\Pi_i|\mathcal{P})$ of how likely a phisher is to perform transactions with attribute values Π_i. For example, let us assume that Π_1 stands for the destination account on a financial transaction. $\Pr(\Pi_1|\mathcal{P})$ will be higher if Π_1 corresponds to a bank operating in a country with regulations that phishers can take advantage of, and which they have consequently used in the past. Based on this information, we can use Bayes' theorem to find the probability of a transaction having been made by the user or by a phisher. Given the probabilities $\Pr(\Pi_i|U)$, and given that the transaction has a set of attributes $\{\Pi_i\}$, we can compute the probability $\Pr(U|\{\Pi_i\})$ of those attributes belonging to a transaction made by U. Similarly, we can compute the probability $\Pr(\mathcal{P}|\{\Pi_i\})$ of those attributes being part of a phishing transaction.

We can thus define a "phishiness" metric σ for determining whether a transaction has been initiated by U or by a phisher impersonating U. Based on ideas from the field of anomaly detection [2, 16], we choose the odds ratio as metric for measuring "phishiness." The odds ratio is computed as:

$$\sigma = \frac{\Pr(\mathcal{P}|\{\Pi_i\})(1 - \Pr(U|\{\Pi_i\}))}{\Pr(U|\{\Pi_i\})(1 - \Pr(\mathcal{P}|\{\Pi_i\}))}$$

If $\sigma > 1$, that means that it is more likely that the transaction was made by a phisher than by the user. Conversely, if $\sigma < 1$, then the transaction is likely to have been initiated by the user U. The rationality behind odds ratio is that it is fast to compute (i.e., it does not require expensive computations and can therefore be used in a context where high throughput is required) and takes into account both the (independently derived) profile information of the phisher and of the credential owner. Especially taking into account attributes of the real credential owner allows to reduce false positives. These attributes are uncorrelated with the phisher's behavior and may reliably be inferred from past sign ons.

6 Security Considerations

We are aware that phishers are not completely defenseless against phoneytokens and phoneypots. In the following, we discuss the measures that phishers may apply to foil our approach.

6.1 Strategies to Counter Phoneytokens

Let us assume the phisher is aware that his credential server may be filled with phoneytokens. We see two ways he might strike back:

First, the phisher might analyze the network traffic, identify the phoneytoken machine (using methods, we discussed in Section 5), and launch a denial-of-service attack. However, this countermeasure can be prevented by using adequate

anti-flooding mechanisms (e.g., [15]) to protect the phoneytoken machine or by simply changing the machine's IP address.

Second, the phisher might personalize mounting attacks. These attacks are called *context aware phishing* [19]. The attacks have in common that they are unique and address individuals, i.e., context aware phishing is tailored to attack a certain user. For instance, the phisher could encrypt a pattern, consisting of the recipient's email address and the time, and use the cryptogram as nonce in a phishing mail. In order to feed the phishing collection server with sufficient phoneytokens, we would have to collect numerous phishing mails (since the mails are fingerprinted). This is not crucial. As mentioned before, many phishing report networks and warehouses exist that might help us to collect a vast number of fingerprinted phishing mails.

6.2 Strategies to Detect Phoneypots

In our model, we assume that phishers know the existence of phoneypots and try to detect and avoid them. A countermeasure for a phisher would be to first test for phoneypot presence, we call the *phoneypot oracle attack*. The phisher queries the phoneypot and observes each interaction with the system to extract hidden data or implementation details. The test may be done with the same methods we described in Section 5.1 in order for enabling the phisher to distinguish the real system from the phoneypot. The queries yield no meaningful results (under the assumption that the phoneypot has been correctly implemented) since the phoneypot is hidden behind the same frontend as the real system is. Therefore, the phisher infers indistinguishable characteristics from and is incable of deciding whether he is on the real system or the phoneypot.

A more realistic countermeasure would be to query the phoneypot oracle whether it really provides the real service by observing the outcome. For instance, one strategy would be to initiate a small transaction as a donation to some organization which lists its donators online. The phisher could use this action to verify that a transaction was successful without revealing his identity or the identity of one of his agents. Nevertheless, such a test would mean considerable time and increase the probability that the deceived customer notices the fraud and the bank freezes the account. There might be a good argument for actually allowing the phoneypot to carry out the simulated service requests to the real world. While the service would have to bear the costs, the prospect of luring criminals into extensively usage of the phoneypot which would allow to collect enough data and to catch them more easily, might out-weight the associated costs. Therefore, there should be a way to allow near real-time review of the service requests initiated on the phoneypot by an analyst empowered to approve certain activities.

7 Conclusion

We have introduced an approach to proactively deter phishing-based identity fraud and proposed a forensic framework of luring, trapping and analyzing phish-

ers in order to profile their sources. We have argued how to trace the agents involved in phishing, and identified the measures criminals may employ to thwart our approach. Even with the constraint that the phoneypot is detected in the long run, it will be able to collect considerable amount of information on phishing activity, and thus forces the phishers to commit resources to detecting phoneypots. Based on this, we believe that phishers who detect the use of phoneypot techniques by an organization will avoid its accounts in future.

Currently, we implement the framework and evaluate methods used to conduct meaningful statistical results in order to define phishing profiles and to achieve accurate detection rates. An interesting topic for future work would be to tweak the profiles to detect freeriders. Freeriders are dishonest customers who claim to be a victim of phishing attacks in order to get back the loss from the service provider.

Acknowledgement

We would like to thank Claudia Diaz, Maximillian Dornseif, Dominik Birk and Felix Gröbert for fruitful discussions.

References

1. A. Adelsbach, S. Gajek, and J. Schwenk. Visual Spoofing of SSL Protected Web Sites and Effective Countermeasures. In *Information Security Practice and Experience Conference*, 2005.
2. D. Agarwal. An empirical bayes approach to detect anomalies in dynamic multidimensional arrays. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 26–33. IEEE Computer Society, 2005.
3. R. Beverly. A robust classifier for passive TCP/IP fingerprinting. In *Passive and Active Network Measurement*, LNCS, pages 158–167, 2004.
4. D. Birk, S. Gajek, F. Gröbert, and A.-R. Sadeghi. Phishing phishers—observing and tracing organized cybercrime. In *ICIMP'07: Proceedings of the Second International Conference on Internet Monitoring and Protection*. IEEE Computer Society, 2007.
5. M. Chandrasekaran, R. Chinchani, and S. Upadhyaya. Phoney: Mimicking user response to detect phishing attacks. *wowmom*, 0:668–672, 2006.
6. T. Fawcett and F. Provost. Fraud detection. In W. Kloesgen and J. Zytkow, editors, *Handbook of Knowledge Discovery and Data Mining*. Oxford University Press, 2002. CeDER Working Paper #IS-99-18, Stern School of Business, New York University, NY, NY 10012.
7. D. Florencio and C. Herley. Stopping a Phishing Attack, Even when the Victims Ignore Warnings. Technical Report MSR-TR-2005-142, Microsoft Research (MSR), 2005.
8. T. Kohno, A. Broido, and K. C. Claffy. Remote physical device fingerprintin. *IEEE Trans. Dependable Sec. Comput*, 2(2):93–108, 2005.
9. A. Litan. Increased Phishing and Online Attacks Cause Dip in Consumer Confidence. Gartner Study, June 2005.

10. A. Litan. Phishing Attacks Leapfrog Despite Attempts to Stop Them. Gartner Study, November 2006.
11. C. M. McRae, R. W. McGrew, and R. B. Vaughn. Honey tokens and web bugs: Developing reactive techniques for investigating phishing scams. *Digital Forensic Practice*, 1(3):193–199, 2006.
12. R. Molva and G. Tsudik. Authentication method with impersonal token cards. In *SP'91: Proceedings of the Symposium on Research in Security and Privacy*, pages 55–65, May 1991.
13. T. Moore and R. Clayton. An empirical analysis of the current state of phishing attack and defence. In *Workshop on the Economics of Information Security*, 2007.
14. M. Najork and A. Heydon. On high-performance web crawling. Technical report, Compaq Systems Research Center, 2001.
15. V. Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *SIGCOMM Comput. Commun. Rev.*, 31(3):38–47, 2001.
16. S. L. Scott. A bayesian paradigm for designing intrusion detection systems. *Computational Statistics & Data Analysis*, 45(1):69–83, 2004.
17. M. Smart, G. R. Malan, and F. Jahanian. Defeating TCP/IP stack fingerprinting. In *USENIX Security Symposium*, 2000.
18. L. Spitzner. Honeytokens: The Other Honeypot, 2003. `http://www.securityfocus.com/infocus/1713`.
19. M. J. T. Jagatic, N. Johnson and F. Menczer. Social phishing, 2007. To appear in Communications of the ACM.
20. The Honeynet Project and Research Alliance. Know your Enemy: Phishing, Identifying remote hosts, without them knowing, 2005. `http://www.honeynet.org/papers/phishing/`.
21. M. Zalewski and W. Stearns. Passive os fingerprinting tool, 2006. `http://lcamtuf.coredump.cx/p0f/README`.