Phighting the Phisher: Using Web Bugs and Honeytokens to Investigate the Source of Phishing Attacks

Craig M. McRae
Master's Degree Student
Mississippi State University
cmm15@msstate.edu

Rayford B. Vaughn
Billie J. Ball Professor of Computer Science and Engineering Director, Center for Computer Security Research
Mississippi State University
vaughn@cse.msstate.edu

Abstract

This paper presents a summary of research findings for a new reacitve phishing investigative technique using web bugs and honeytokens. Phishing has become a rampant problem in today's society and has cost financial institutions millions of dollars per year. Today's reactive techniques against phishing usually involve methods that simply minimize the damage rather than attempting to actually track down a phisher. Our research objective is to track down a phisher to the IP address of the phisher's workstation rather than innocent machines used as intermediaries. By using web bugs and honeytokens on the fake web site forms the phisher presents, one can log accesses to the honeytokens by the phisher when the attacker views the results of the forms. Research results to date are presented in this paper.

1. Introduction

Phishing, which is an acronym for password harvesting fishing, [1] is the use of social engineering and spam e-mails to illicit sensitive information such as usernames, passwords, and financial information from unsuspecting victims. This form of identity theft is a major problem today for business, consumers and the legal community. Phishing attacks cost financial institutions millions of dollars per year. In a one year period that ended in April of 2004, there were an estimated 1.8 million phishing attacks that generated $1.2 billion in losses. When individuals are victimized by phishing attacks, their financial institution must reimburse the individuals for damages just like any other form of identity theft. Even though this institution has done nothing wrong, they are the ones that ultimately pay [2].

Financial institutions are also victimized in another way that is often overlooked. When people become victims of phishing attacks or become aware of the threat of phishing attacks, the financial institution loses the ability to communicate with their customer base through e-mail. Their customers no longer trust e-mails that come from those institutions and usually assume it to be a phishing e-mail [3]. Customers also tend to blame the legitimate institution for phishing attacks.

It is becoming increasingly important to try to protect against phishing attacks. Law enforcement has a need to be able to track a phisher to their location so that attackers can be identified and stopped. The research described in this paper has as its objective to introduce a new way of tracking phishing attacks to the actual IP address of the phisher's machine and not the IP address of one of the computers that the attacker has compromised for use in attacks. The set of compromised machines that an attacker uses is known as a *botnet*. The actual setup of the experiment used is explained later in this paper following a description of web bugs, honeytokens, and current anti-phishing techniques.

2. Web Bugs

A web bug can be defined as any HTML element that is used to, at least in part, secretly gather information about a particular user. Usually these come in the form of images. For that reason, they can also be referred to as pixel tags and clear GIFs [4]. These images could be as large as the web site's logo

Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS'07)
0-7695-2755-8/07 $20.00 © 2007 IEEE
1530-1605/07 $20.00 © 2007 IEEE

or as small as a one by one square pixel.

David Martin et. al. [4] outlines several different things that web bugs can do. A few of these include:

- Count web site hits
- Send demographic data to an internet marketing company such as gender, age, and zip code
- Track web browsing habits of a user
- Report browser information back to a web site.

Web bugs can accomplish some of this by explicitly giving an image a unique file name for that particular web user. An example of a web bug in use can be seen in e-mails as a technique often used by spammers. A one by one pixel white image can be created with a particular e-mail address embedded into the filename such as joesmith_example_com.gif. The spammer will then embed this image into that e-mail to joesmith@example.com. When Joe Smith views his e-mail, the small white image will go unnoticed. The image is loaded from the spammer's web server, and will create an entry on the server log for the filename joesmith_example_com.gif. The spammer now knows that Joe Smith received his e-mail and will continue to use that e-mail address in future spam attacks.

3. Honeytokens

Lance Spitzner, the leader of the Honeynet Project, has defined the term honeytoken as "a digital or information system resource whose value lies in the unauthorized use of that resource" [5]. Honeytokens can be any digital data. They can consist of documents, images, or even data such as a phony login/password combination. A honeytoken may be any data on a system for which accesses can be logged, and whose access automatically implies unauthorized activity.

While the term 'honeytoken' is new, the concept is not. The term 'honeytoken' was created by Augusto Paes de Barros on February 21, 2003. He used it in a e-mail that went to a list of security professionals.[6] Spitzner further mentions that some map makers will insert phony roads or cities on their maps. They do this so that they can prove when competitors sell copies of their maps. Spitzner gives other hypothetical examples of honeytokens in use. One such example shows how a honeytoken could possibly help in database security. Hospitals could create a bogus medical record for John F. Kennedy and then track the access to that tuple of the database. Anyone who is viewing that record is violating the privacy of patient data. This is because stored procedures and other layers of database access can be designed to avoid accessing honeytokens. In this situation, access to a honeytoken implies that the database is not being accessed through approved means. Also, financial institutions can create bogus accounts. If one tries to access those accounts, then the institution's system has been compromised [5].

The key to using honeytokens is to give the token unique identifiable elements to guarantee that the only access to that token would be by unauthorized parties. If the token could be viewed in normal interaction with a system, the token's tracking ability is compromised. Honeytokens' greatest advantage lies in their flexibility and their minimal cost.

4. Current Reactive Techniques in Phishing

There are currently numerous approaches to protecting users from phishing schemes. As previously mentioned, these approaches do not get to the root of the problem. These methods do not track down the phishing schemes to the user running the scam.

The simplest of these methods is to attempt to take down phishing sites quickly. Web crawlers similar to those used for search engines can also be used to find phishing sites. The information is then passed on to the appropriate Internet Service Provider (ISP) in order to take the site down. The disadvantage of this method is the difficulty in finding these sites quickly. Also, some web sites hosted in foreign countries do not have similar laws that can justify removing the site.

Another defense is to flood a phisher's database with false information. This flood of information is not intended on being a denial of service (DOS) attack. This flood is designed to make the phisher unable to distinguish correct data from incorrect data in their database. This makes the user's database virtually unusable. This method does nothing to prevent phishing attacks. It simply tries to minimize the ability to carry through with the theft of financial data [2].

One of the few current approaches that can possibly target the root of the problem and prevent further phishing attacks is to watch corporations' web logs for users downloading their images. Creators of phishing web sites usually use the actual images from the corporate web site to make the phishing web site more believable. If users are downloading images on to their personal computer, then their IP address will show up in the server logs. The Corillian Fraud Detection System (CFDS) is a commercial server that

looks for such a behavior in web logs. It then investigates further to find the phishing site that is illegally using those images. Corillian then notifies the administrator of the compromised server and the authorities [2].

Internet browser creators are trying to help as well. This usually involves comparing the address of a web site to a list of 'blacklisted' web sites. This blacklist can come from the browser's creators, users, and other sources. The sites may also have a list of 'whitelisted' web sites as well. If a site does not apply to either list, some security measures may be put in place such as disabling mobile code like ActiveX and JavaScript [2]. Internet Explorer 7.2 disallows mobile code to disable the address bar. Phishers often do this to hide the actual location of the site. They can even include a fake address bar showing a fake URL to further obfuscate their activities. Internet Explorer 7.2 also includes a phishing filter which allows users to report a phishing site. This reporting eventually can lead to the site becoming blacklisted and users of IE are then notified if they visit that site. Users are notified of both confirmed and suspicious phishing sites.

5. Plausibility Investigation

Using web bugs and honeytokens, the research described in this paper attempts to develop a method for tracking phishing attacks to their source. This involves completing the phishers' web forms with HTML image tags of one pixel by one pixel images and HTML web page links. Both the image files and the web page links are hosted on a machine in a laboratory environment with an unrestricted internet connection. This host is on its own dedicated subnet separate from the rest of the network for isolation purposes. If a phisher decides to attack this research system, the departmental and campus network is then protected from compromise. The images used in this project are web bugs since they are uniquely named for each phishing e-mail and can be used to gather information about the individual or group that views the data collected by the phishing scheme. The image files and the HTML files are also honeytokens since they are pieces of data that are unlikely to be accessed by users other than those involved in specific phishing schemes.

The image and HTML files are both named using a unique time stamp. That filename is stored, along with the URL of the attack, to associate the files with each particular phishing attack. As previously mentioned, if the phisher views the results in an HTML enabled environment that does not filter or block third party images from being loaded, such as is currently the case with many webmail applications, browsers, and applications such as Outlook, the images will be retrieved from the server by the attacker. The phisher will likely know that something is not right with the data due to the links being displayed and the lack of personal information. By the time the phisher realizes this, the image is already loaded. Any further investigation by the phisher will likely result in more information being logged about them. If the phisher's client software loads the images or the phisher accesses the hyperlinks, a referral will be generated in the web logs on the tracking server. Some of the information included in the log file is:

- The IP address of the actual box where the results were viewed
- The web page where the phisher viewed the results including web mail accounts like GMail, Yahoo Mail, or Hotmail
- The browser type that was used to view these results
- A guess at the operating system the phisher is working from

An analysis of an actual experimental referral that was generated is described later in this paper. Below we first generally describe our procedure test setup followed by a real case experiment. We are currently acquiring additional experimental data beyond what reported in this paper. However, we chose not to present real case data beyond the one case printed here to protect the anonymity of the additional data until the experimentation is complete. Additional data will be presented in the presentation that accompanies this paper.

5.1 Example Test Case

In order to test this concept at tracking phishing schemes, two simple web page forms were created. One form is to test the 'GET' form request, and the second is to test the 'POST' form request. Most phishing attacks begin with a login page, so the two forms were created as login forms for testing. The form was then fed HTML tags similar to the following:

- `myspam@hotmail.com`
- `<img src=``http://192.168.0.0/images/2005_Nov_28_11_20_05.gif''>`

Last, a web page was created that displayed the results of the fictitious phishing site. This web page was opened up in a web browser that allowed the viewing of third-party images. To guarantee that a referral had been made, some of the web page links were accessed as well. At this point the web server log files on the experimental machine were checked, and we confirmed that the referral was made from the phishing investigation.

5.2 Real World Test Cases

After determining that the procedure worked in our lab test cases, e-mails were sent to faculty and grad students asking them to submit phishing e-mails they had received for experiment purposes. Other individuals that were capable of identifying phishing schemes were asked to do the same.

Originally the plan was to automate the process with a single Perl script or series of scripts. The scripts used the UNIX utility *wget t*o automate the form submissions. A problem arose when it was noticed the script was not able to handle redirect pages in the case of nested forms. Due to the limited time available for this test case, it was decided to continue the form submission process manually to achieve results and to prove tracking was possible.

The Perl script used to automate this process was modified to simply take the input of a particular web site URL, generate the appropriate image and HTML files, and store the web site URL with the image and HTML filenames into a comma-separated values (CSV) file. The CSV is simply a text-based file where the values of the file are separated by commas. This file will allow the data to be easily exported in programs such as Microsoft Excel for easier analysis. Each comma separator will export the data into a new table cell in Microsoft Excel. The new lines in the file will dictate the row of the table that the data appears. At this point, the HTML tags linking the HTML web page and image files could then be copied and pasted into the individual fields of the phisher's forms. If the entire HTML tag could not fit into a form field, then the values were simply created with fictitious names, or in the case of number fields such as credit card numbers, the appropriate number of 5s were input into the field. This problem will be bypassed once we have the process fully automated. An automation script could potentially bypass the HTML form restrictions and any possible JavaScript-based error checking. Another alternative solution is to use a proxy server such as Burp Suite [7]. A proxy server basically operates as a man-in-the-middle. When data is submitted to the form, it goes through the client-side error checking and then is sent to the proxy. The proxy can then alter the data before sending it to the server. This will allow the outgoing data stream to be filtered in order to allow the HTML tags to be entered into the outgoing data. After submitting form data, the packet can be changed to include the HTML tags. This would also bypass the HTML form restrictions and JavaScript-based error checking.

During the week of November 11 through November 18, 2005 eleven phishing sites were investigated and two referrals were found. While this was not a particularly successful result, we believe we can greatly improve it with modification we are now making to the process. We believe that some of the other nine attacks may not have generated referrals since the only fields that would accept the full HTML tag were the login fields. If the phisher was not worried about the login parameters and only looked at the financial data from later forms, then the referral would not occur. In other cases, none of the fields would accept the full HTML tag. This problem can be fixed by script automation or by the use of a proxy server. One of the referrals that were generated during this week is shown below:

```
82.79.137.22 - - [11/Nov/2005:12:08:12 -
0600] "GET
/images/2005_Nov_12_11_52_53.gif
HTTP/1.1" 404 309
"http://mail.google.com/mail/h/15zatsmc8x
2ql/?th=1078083bcd28d7d6&v=c"
"Mozilla/4.0 (compatible; MSIE 6.0;
Windows 98) Opera 7.50 [en]"
```

We can infer the following from this referral:

- 82.79.137.22 is the IP address that was logged by the referral. This IP could be the traced to the ISP used by the phisher's machine.
- The time stamp given (11/Nov/2005:12:08:12 can show us how quickly the phisher gained access to the data when compared to the time stamp of the filename. In this case, it was approximately 15 minutes after the file was created.
- /images/2005_Nov_12_11_52_53.gif is the web bug and honeytoken that was accessed.
- The results of the attack were e-mailed to the attacker via a GMail e-mail account. Attempts to reproduce the GMail session by inputting the URL into a browser were unsuccessful.
- The Opera web browser was used by the attacker to view his information.
- The '[en]' at the end of the log represents English

as the native language of the computer being used.

Actually tracking the source of the phishing scheme occurs with a WHOIS [8] query of the IP address given in the referral. Using the IP address from the previous referral returned the following:

```
inetnum:        82.79.137.0 - 82.79.137.63
netname:        RO-BZ-METRONETWORK
descr:          Metronetwork SRL
country:        RO
admin-c:        IS1460-RIPE
tech-c:         IS1460-RIPE
tech-c:         RDS-RIPE
status:         ASSIGNED PA
remarks:        +--------------------------+
remarks:        | ABUSE CONTACT:           |
remarks:        | abuse@rdsnet.ro          |
remarks:        | IN CASE OF HACK ATTACKS, |
remarks:        | ILLEGAL ACTIVITY,        |
remarks:        | VIOLATIONS, SCAMS,       |
remarks:        | PROBES, SPAM, ETC.       |
remarks:        +--------------------------+
mnt-by:         AS8708-MNT
mnt-lower:      AS8708-MNT
source:         RIPE # Filtered
role:           Romania Data Systems NOC
address:        71-75 Dr. Staicovici
address:        Bucharest / ROMANIA
phone:          +40 21 30 10 888
fax-no:         +40 21 30 10 892
e-mail:         contact-tech@rdsnet.ro
admin-c:        CN19-RIPE
tech-c:         CN19-RIPE
tech-c:         GEPU1-RIPE
nic-hdl:        RDS-RIPE
mnt-by:         AS8708-MNT
remarks:        +--------------------------+
remarks:        | ABUSE CONTACT:           |
remarks:        | abuse@rdsnet.ro          |
remarks:        | IN CASE OF HACK ATTACKS, |
remarks:        | ILLEGAL ACTIVITY,        |
remarks:        | VIOLATIONS, SCAMS,       |
remarks:        | PROBES, SPAM, ETC.       |
remarks:        +--------------------------+
source:         RIPE # Filtered
person:         Iordache Silviu
address:        Str Traian Vuia, bl 16,
                sc. C, et3, ap.54
address:        Buzau / Romania
phone:          +4-0744821745
fax-no:         +4-338401143
e-mail:         iorsior@yahoo.com
nic-hdl:        IS1460-RIPE
mnt-by:         AS8708-MNT
mnt-by:         AS8708-MNT
remarks:        +--------------------------+
remarks:        | ABUSE CONTACT:           |
remarks:        | abuse@rdsnet.ro          |
remarks:        | IN CASE OF HACK ATTACKS, |
remarks:        | ILLEGAL ACTIVITY,        |
remarks:        | VIOLATIONS, SCAMS,       |
remarks:        | PROBES, SPAM, ETC.       |
remarks:        +--------------------------+
source:         RIPE # Filtered

% Information related to
'82.76.0.0/14AS8708'
route:          82.76.0.0/14
descr:          RDSNET
origin:         AS8708
mnt-by:         AS8708-MNT
source:         RIPE # Filtered
```

From this whois query, quite a bit of information can be gathered:

- The entire range of IP addresses that this organization owns is shown.
- The actual street address, city, state and zip of this organization that owns that IP address is given. In this case, the owner of this IP address resides in Bucharest, Romania.
- Phone numbers for the organization are given.
- A generic e-mail contact is provided.
- Another e-mail contact where abuse is reported.
- A named contact who is a Senior IP engineer in this organization is given with his e-mail and phone number.

At this point, the investigation moves to the ISP that owns the particular IP address. In order to trace the source of the IP, the ISP would need to cooperate and check their logs to try to determine what machine was logged in at the particular IP address at a given time.

There are limitations of tracking by IP addresses. Many ISPs use dynamic IPs with their customers. Every time a machine logs on to the ISP's network, they could have a different IP address. This makes it very difficult to determine who had a particular IP address at a particular time. Some ISPs do not actually log IP assignments; therefore, the machine that was using a particular IP at a particular time cannot be determined. Other limitations include the use of public wireless access points at coffee shops and other public venues and the use of anonymous proxies which hides the actual IP address of a particular machine.

6. Formal Investigation

For a formal investigation, we are automating scripts to get around the problem of error-checking as mentioned before. Also, automated scripts will allow the experiment to be run on a larger scale. The scripts will perform the tasks that were done manually in the initial experiments. These scripts should take the input of a phishing URL or a series of phishing URLs. The script can then create images and HTML pages named for the current system time. These images and HTML files are the honeytokens and web bugs to be used for the experiment. The filenames will then be logged in a CSV file with each URL so

that the filename can be connected to the URL. The filenames are stored into a CSV file just for simplicity. The logs can be easily ported into Excel or other spreadsheet software for charting and other analysis.

After setting up the files for that particular URL, the script will then open up each phishing URL one at a time. The script will then analyze the HTML on the page to determine all of the form variables and types. The script will search for these tags in the HTML code such as '<input>' tags. These tags contain variables inside the tag such as the variable name associated with that field, the type of input field, and any particular value it may currently have. The current value is useful in the case of hidden form variables that are used to submit data that does not appear in the form. The script will then add the HTML and image links to the values of the variables with a text data type. The script will simply select an option from other fields such as drop-down menus, select boxes, and check boxes. Once the script submits the data, it will analyze the next page to determine if it contains a form. This can be accomplished by simply looking for the beginning of a '<form>' HTML tag. If the page does contain a form, then the process is repeated. Some phishing sites actually forward the user back to the legitimate company's login page after they have completed their attack. To keep the script from going into an infinite loop at this point, a limit of how many pages it analyzes will be set. This should allow the script to process the entirety of the phishing site without flooding a legitimate site in bad login attempts. The script should be able to handle referral pages that caused the initial experiment automation attempts to fail.

We plan a second script that will frequently monitor the web server logs for referrals from the investigation. This can be done by looking for key values in the referral that would only come from this investigation. Once a referral is found, then the script should log the filename found in the referral, reference the URL associated with that filename, and log both together in a separate CSV file.

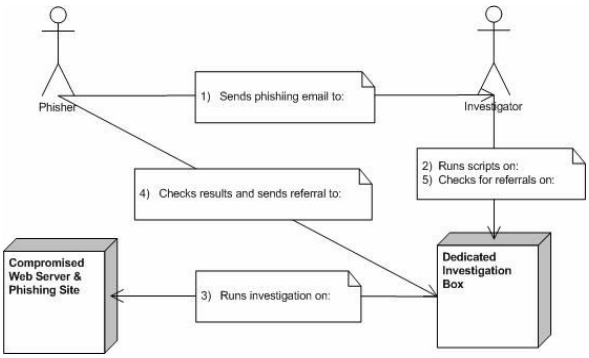

Figure 1: UML diagram of investigation

The steps of the invesigation are shown above in figure 1. Initially the phisher sends a phishing email to the investigator. The investigator then uses the dedicated investigation box to run the automation scripts. The dedicated box through the automation scripts runs the actual investigative steps on to the compromised web server that contains the phishing site. After the investigation has been run, the phisher checks the results of their web site forms. If he checks his results in a HTML environment, a referral is sent to the dedicated box. The investigator later checks and sees the referral on the dedicated box. At this point further investigation into the IP address can be done.

The investigation will be run over a three month period. The amount of data that is collected will depend on the number of phishing attacks that are received during that time frame. If more data is needed after the three month period, then the investigation can be continued for a longer period of time.

7. Conclusions

In conclusion, our initial experiments show that this investigative method can be a valid way of tracking phishing attacks. With this type of investigation, the attacks can actually be traced to the source of the attack. Our early results are encouraging. We have discovered the source of some problems and we have improved our success rate as well as the process we use.

Since most of the data in this project came from only a few individuals (students and faculty), most of the attacks were repeated. When the data set is expanded, more attacks can be explored.

As previously mentioned (as with all law enforcement techniques) there are countermeasures available to the attacker. The simplest

countermeasure would be to only view the results of phishing forms in text-only viewers. This would disable the HTML code and prevent any referral from being made in the web server log. The other simple countermeasure is to disable the viewing of third-party images in whatever browser is used in viewing the results. If the image is never loaded, then a referral is never made on the web server. We believe at the current time, many (if not most) phishers are not employing these countermeasures.

8. Acknowledgements

The authors whish to acknowledge the assistance and support of Mr. Robert Wesley McGrew who first conceived of this technique and provided much of the technical talent to design and implement this experiment.

9. References

[1] The Honeynet Project Research Alliance, "Know Your Enemy: Phishing - Beyond the Scenes of Phishing Attacks," 2005, http://www.honeynet.org/papers/phishing/index.html (current Feb.6, 2006).

[2] D. Geer, "Security Technologies go Phishing," *Computer*, vol. 38, no. 6, June 2005, pp. 18-21.

[3] G. Goth, "Phishing Attacks Rising, but Dollar Losses Down," *IEEE Security and Privacy Magazine*, vol. 3, no. 1, January-February 2005, p. 8.

[4] D. Martin, H. Wu, and A. Alsaid, "Hidden Surveillance by Web Sites: Web Bugs in Contemporary Use," *Communication of the ACM*, vol. 46, no. 12, 2003, pp. 258 – 264.

[5] L. Spitzner, "Honeytokens: The Other Honeypots," 2003, http://www.securityfocus.com/infocus/1713 (current Nov. 18, 2005).

[6] N. Thompson, "New Economy, the 'Honeytoken' an Innocuous Tag in a File Can Signal an Intrusion in a Company's Database," *NY Times*, April 2003.

[7] "PortSwigger.net – web application hack tools," Aug. 2005; http://portswigger.net/suite/.

[8] "WHOIS Protocol Specification," Aug. 2005; http://www.rfc-editor.org/rfc/rfc3912.txt