Global Information Assurance Certification Paper

This is a GIAC Gold Template

Phishing Defenses for Webmail Providers

GIAC (GSEC) Gold Certification

Author: Rich Graves, rgraves@carleton.edu
Advisor: Rob VandenBrink

Accepted: January 20th 2013

Abstract

This paper surveys common techniques for battling phishing attacks, especially those targeting Internet-accessible webmail servers, and introduces some lesser known countermeasures. We discuss automating the retroactive eradication of phishing messages from user mailboxes, image referrer analysis, phishing your own users as an awareness-raising exercise, and the identification of compromised webmail accounts by monitoring user behavior. Examples focus on open source software, including SpamAssassin and Zimbra, as implemented at colleges and universities. Some lessons may be applicable to proprietary technologies and non-webmail situations, such as financial site and enterprise spear phishing.

Introduction

This paper primarily addresses mass-market, consumer-grade phishing against webmail users. While the disclosure of a single webmail password to an unsophisticated attacker has a far lower single loss expectancy (SLE) than a spear phishing incident or a banking password compromise, the frequency and thus annual loss expectancy (ALE) can be higher. At minimum, each affected user will need to change his or her password, and the help desk and security/systems staff will usually need to intervene. Very often, a phished account is used to send more phish or spam, which can harm the organization's reputation. Finally, efforts to protect against garden-variety phishing attacks can also be effective against more serious threats. Our goals are to prevent phishing attacks if possible; and failing that, to streamline the identification, containment, eradication, and recovery process to reduce the cost.

What exactly is a phishing attack? "Phishing emails are semantic attacks that con people into divulging sensitive information using techniques to make the user believe that information is being requested by a legitimate source" (Downs, Holbrook, & Cranor, 2006). In the literature, much attention has been paid to the users: how to train them, how to create user interface cues with anti-phishing browser toolbars, etc. This research on "securing the human" is, and will remain, necessary and productive. For this paper, however, we are more interested in the technical elements surrounding the psychological attack, and how system administrators and security incident responders might use automated tools to contain the impact of human failures. It is an unfortunate fact that when a large group of people receive a phish, some *will* fall for the con; what then?

The defender has an opportunity for active interception at three stages. Reconnaissance (acquisition of email addresses, development of content) is not discussed here; while the defender has some control over information leakage, these are not active countermeasures. In our model, the first stage is when the message reaches the email gateway, which must then decide whether to accept, reject, or quarantine it. Second, the user sees the message, and might fall for the con. As defenders, we can train the users not to respond to phishing attacks, or attempt to block responses by technical means. Third, if the first and second layers of defense fail and the user "successfully" provides his or her password to the attacker, then we must recognize that incident and contain the abuse of their account. Successful defense in depth requires controls at each of these levels. Which set of controls is most appropriate to your organization depends on your particular

circumstances.

Preventing inbound phishing emails

Although phishing messages can be delivered by instant message, social networking, and voice, the vast majority are delivered by email. Consider some level of counter-reconnaissance. Avoid publishing email addresses and system details helpful for creating more convincing social engineering attacks. This paper assumes that it's too late for that; phishing messages are on their way. Antispam software and services are thus the first active line of defense.

Some of the examples below make reference to scores derived from the open source SpamAssassin program. In order to understand this paper, the essentials are:

> Emails are assigned a score based on how "spammy" they are.
>
> Rules are typically named UPPERCASE_WITH_UNDERSCORES, and can add or subtract from the score when they match.
>
> A summary of rules that matched, and their associated scores, is added to the X-Spam-Status: header.
>
> In our environment, messages scored 6.6 and higher go to the Junk folder. Messages scored 15+ are rejected outright.

Antispam: URIBLs

Uniform Resource Identifier Blocklists (URIBLs) are composed of domain names or specific URLs that, in the opinion of the list maintainer, are "bad." Antispam software scans the body of the message for URIs, checks them against the configured block lists, and scores the message appropriately. The most well-known site, the Spam URI Realtime Blocklist (surbl.org), was publicly launched in 2004, though a few proprietary antispam vendors and ISPs, including BrightMail and AOL, were experimenting with the concept earlier. Most URIBLs have the broad mission of fighting spam. The list ph.surbl.org is free (with registration) and focuses on phishing.

URIBLs can be very effective against sites registered by, or known to have been compromised by, phishing gangs. Possibly convincing look-alike site names like wwwexample.com (spoofing www.example.com) can be added to the URIBL and thus deterred. However, this strategy suffers from the same time lag as antivirus or any other signature-based technology. Heuristics such as "Is the site affiliated with an often-abused registrar like BizCN?" and "Was the site registered within the past month?" can help, but at risk of false positives.

URIBLs are rarely effective against messages like the one below, which abuses a specific URI on a legitimate Google Apps domain. More examples of phishing messages pointing at legitimate "form processing" sites appear later in this paper.

```
Date: Sun, 18 Nov 2012 19:01:28 -0600 (CST)
To: undisclosed-recipients: ;
Subject: mailbox quota exceeded Ref # [1M1QC063GF4xn2r]

Your mailbox storage limit/quota has been exceeded until you re-validate your
mailbox you cannot send or recieve e-mail.To re-validate your mailbox. - > Click
Here: https://docs.google.com/a/nebo.edu/spreadsheet/viewform?
formkey=dFkzcXM5Z3dLUWRmckl5UEQzZzZ6dVE6MQ
```

Figure 1, phish pointing at Google Apps (uninteresting headers removed)

Popular web browsers enable a form of URI blacklisting on the client end. Firefox calls the feature Phishing and Malware Protection; Internet Explorer brands it as SmartScreen Filter; Chrome has SafeBrowsing. Even if the browser-level blocklists are less timely and comprehensive than those running at the antispam gateway, they provide an important second line of defense. The time elapsed between antispam inspection and user click might be enough for a hostile site to be recognized.

Antispam: Statistical Classifiers

There is a large body of computer science literature applying machine learning and statistical analysis, popularly but not always accurately described as Bayesian, to the recognition of phishing messages. In papers such as the exemplary Bergholz, A., De Beer, J., Glahn, S., Moens, M., Paaß, G., & Strobel, S. (2010), researchers assemble a corpus of messages classified by humans as either spam or not-spam. Features including, but not limited to, the content of the headers and body (current research centers on what aspects of the message are most useful), are fed into a statistical model that then "learns" to classify future messages.

When this research is applied to real systems, if the model says the message looks like past spam or phish, then it goes to the user's Junk folder (or may be rejected entirely). Over time, this has proven to be a fairly successful strategy for fighting spam, but as discussed below, general-purpose statistical classifiers have not been so effective for phish. The phishing messages in Figures 2 and 3, below, were recently received at Carleton College. The X-Spam-Status header (portions highlighted in yellow) is a terse summary of SpamAssassin's decision process. The first message was not considered spam (No, score=3.703). The statistical classifier, indicated by BAYES_05, actually had a negative effect. The meaning of RCVD_IN_DNSWL_NONE and CMAE_1 are discussed later.

```
X-Spam-Status: No, score=3.703 tagged_above=-10 required=6.6
 tests=[BAYES_05=-0.5, RCVD_IN_DNSWL_NONE=-0.0001,
```

```
 RCVD_IN_RP_RNBL=1.31, RG_PHISH_FROM=0.5,
 TVD_PH_BODY_ACCOUNTS_PRE=2.393] autolearn=no
From: "Webmail Admin" <webadmin@info.net>
Subject: Messenger from Administrator
Date: Fri, 7 Dec 2012 20:58:08 +0700
X-Mailer: OpenWebMail 2.53
X-OriginatingIP: 69.163.40.212 (widiharto)

Your E-mail® Account needs to be updated with our F-Secure R-HTK4S new(2013)
version anti-spam/anti-virus/anti-spyware. Please click this link below for
update

http://www.formstack.com/forms/?1346988-JLxwDb5BzU

We Are Sorry For Any Inconvenience.

Regards,
WEBMAIL ADMINISTRATOR Copyright© 2012
All rights reserved®.ABN 31 088 377 860
```

Figure 2 - phish pointing at free form builder formstack.com

```
X-Spam-Status: Yes, score=12.093 tagged_above=-10 required=6.6
 tests=[BAYES_80=2, CMAE_1=7.2, RCVD_IN_DNSWL_NONE=-0.0001,
 RG_PHISH_FROM=0.5, TVD_PH_BODY_ACCOUNTS_PRE=2.393] autolearn=no
From: "Webmail Admin" <webadmin@info.net>
Subject: Messenger from Administrator
Date: Sat, 8 Dec 2012 20:45:18 +0700
X-Mailer: OpenWebMail 2.53
X-OriginatingIP: 174.140.166.248 (widiharto)

Your E-mail® Account needs to be updated with our F-Secure R-HTK4S new(2013)
version anti-spam/anti-virus/anti-spyware. Please click this link below for
update

http://www.tuclouds.com/form.php?id=17320

We Are Sorry For Any Inconvenience.

Regards,
WEBMAIL ADMINISTRATOR Copyright© 2012
All rights reserved®.ABN 31 088 377 860
```

Figure 3 - phish pointing at free form builder tuclouds.com

There are two problems with statistical classifiers as applied to the sort of phish discussed here. First, if they are sent from (stolen) accounts on legitimate webmail providers, then the From/Received/X-Mailer headers, which are part of the calculation, are not going to be spam indicators. Second, the words in the body are intended to mimic real tech support messages, and may have specific references to the target organization. As mentioned by Bergholz, et al (2010), if the attackers attempt to obfuscate their hyperlinks, then those very tactics of obfuscation are likely to help the classifier; but, they don't always do that. Figures 1, 2, and 3 are all plain text. In the headers of Figure 2, the formstack.com message, BAYES_05 means that the SpamAssassin Bayesian classifier (admittedly not tuned specifically for phishing detection) estimates only a 5% chance that the message is spam. Figure 3 - phish pointing at free form builder tuclouds.com. The formstack.com message scored only 3.703, which means it was delivered to the user's Inbox. The tuclouds.com message scored 12.093, enough to quarantine it in the Junk folder, because of a combination of CMAE and Bayes learning.

In our experience, the Bayesian filter included in SpamAssassin and derivatives (including Barracuda, Zimbra, SonicWall) is more likely to help phishing messages get through than to stop them. Other tactics are more effective.

Antispam: Fixed Signatures

The SpamAssassin documentation wiki (*WritingRules*, 2011) does not encourage the writing of custom rulesets. That said, some ruleset fragments specifically tailored to webmail phishing messages follow. In SpamAssassin, "meta" rules contain Boolean or arithmetical assertions about which other rules must all fire together. They do what naïve statistical filters cannot: express a possibly complex logical set of conditions, not just correlations. For instance, it is OK for people in Nigeria to send us email. It is not OK for people in Nigeria to send us short messages about passwords. It is OK, and maybe even a sign that a message is *not* spam, if X-headers indicate that the message was sent via webmail. However, it is rare for legitimate technical support bulletins and password reset messages to come from webmail servers; therefore, these two elements together suggest phishing.

```
# They often attempt to forge messages from help desk-ish accounts
# Note that this matches the full name, not just the email address component.
header  __RG_PHISH_FROM        From =~      /(?:update|support|verify|verification|confirm|help|helpdesk|
team|carleton|center|service|upgrade|webmail|manage|admin|info|webmaster|customer|login|database|activate|
verif|secur|account)[^\s]{0,20}\@\b/i

# We receive legitimate webmail from Nigeria. But ANDed with other rules…
header  RG_XIP_PHISHY   X-Original-IP =~ /^(?:41|80|84\.12|92|200\.215|202\.106|189\.79|
196)\.\d{1,3}\.\d{1,3}/
describe RG_XIP_PHISHY  Appears to be webmail from a spammy source
score   RG_XIP_PHISHY   0.4

# Similar rationale to webmail... mail from Exchange is fine, but should not be talking about passwords.
header __RG_THREAD_INDEX Thread-Index =~ /./
describe __RG_THREAD_INDEX Exchange-looking

meta    RG_PHISH_EXCH          (__RG_THREAD_INDEX && __RG_BODY_SHORT &&
(RG_PHISH_BODY_1 || RG_PHISHY_REPLY || RG_PHISH_SIG || RG_PHISH_SIG_2))
describe        RG_PHISH_EXCH           Phishy Exchange-sourced message
score           RG_PHISH_EXCH           2

body RG_PHISH_FORM1     /\/form1\.html/
describe RG_PHISH_FORM1 PHP formgen form1.html
score RG_PHISH_FORM1 6

body RG_PHISH_GDOC_FORM /\/\/docs.google.com\/spreadsheet\/viewform\?formkey=/
describe RG_PHISH_GDOC_FORM Google Doc spreadsheet form
score RG_PHISH_GDOC_FORM 1.6
```

Figure 4 - example custom SpamAssassin rules.

The RG_PHISH_FORM rule refers to a basic form processor that is often abused in low-end phishing attacks (further discussed in section , Check logs for access to suspect phishing sites). The score of 6 is just short of the 6.6 needed to land the message in the Junk folder. Usually, phishing messages match some other rule. The RG_PHISH_GDOC_FORM has a far lower score of 1.6 because we (occasionally) see Google Doc spreadsheet forms used legitimately. Another interesting characteristic of Google Docs as a phish collection site is that the links tend to be https, hence opaque to IDS/IPS/full packet capture (unless doing full SSL intercept, which is unlikely in an ISP or residential college environment).

Antispam: Fuzzy Checksums

CloudMark, BrightMail, and free exchanges such as DCC and Pyzor, can block phishing messages, but only after many similar messages have already been delivered to other sites. We often find that phishing messages that were “clean” at delivery time are flagged by CMAE (CloudMark Authority Engine, a proprietary plug-in for SpamAssassin) a few hours later. Why the difference? Figure 3 was delivered a full day later. CloudMark had time to add the phishing message to their signature database. Multiple spam complaints regarding the first wave of phishing messages changed the Bayes classification of the second wave to 80% probable spam.

Antispam: IP Reputation Blacklists

IP reputation blacklists, such as Spamhaus Block List, tend not to be helpful when phish is sent from legitimate webmail servers, abusing accounts that have themselves been phished. In both Figure 2 and Figure 3, the "Messenger from Administrator" phish above, the highlighted RCVD_IN_DNSWL_NONE rule hit means that they came from a known, legitimate email server. IP blacklisting is not helpful for "real-time" response to new attacks. They remain relevant to this paper, though, for another reason. Sites that fail to control webmail phishing are likely to find themselves on the wrong end of a blacklist. To avoid being tarred as a spam source, pay attention to the next two sections: stopping users from replying, and identification and containment of compromised accounts.

Stop users from replying to delivered phish

Even with the best antispam filter, some messages are going to be delivered. It's not too late to make a difference.

Don't trust passwords for authentication

Ideally, a phished password would be useless, because a second authentication factor is required. Alas, this is not currently realistic for ISPs or most universities. In addition, an advanced attacker might be able to defeat some authentication token schemes by acting as a man in the middle, relaying the challenge/response between the user and the legitimate web server.

Security awareness training

Security awareness training is critically important and promises to be the single most effective countermeasure, with the possible exception of two-factor authentication. However, both are beyond the scope of this paper, and both are difficult to deploy to the customers of public ISPs and schools. It is appalling how many otherwise intelligent college students, staff, and faculty fall for the crudest of phishing messages, but this is not unique to education sector. One would hope that RSA Security has some level of security awareness, yet their well-publicized breach in 2011 was caused by one or more employees who pulled a not-terribly-clever message out of their Junk folder and opened the attachment (Hypponen, M., 2011). In June 2011, Carleton College, among other sites, received thousands of phishing messages from an attacker logged on to an account at uGov.gov, a Zimbra webmail server dedicated to unclassified information sharing among US intelligence agencies. The messages were unsophisticated, and the X-Originating-IP

header resolved to Nigeria. If employees of RSA Security and the intelligence community can be fooled by phishing messages, what hope do we have?

Phishing your own users

An increasingly common method of both training and assessing users is to run simulated phishing campaigns. Spransy (2012) reports a 40.9% improvement (i.e., decrease) in the proportion of users responding to phishing attacks after three rounds. That is a very impressive achievement, but it is still worth pursuing other measures. Many organizations, such as public ISPs, would have serious ethical and public relations difficulties with such a program. For the balance of this paper, we return to technical countermeasures available to system administrators working on their own, without such communication and policy concerns (except where noted).

Block access to collection forms

Network security staff should block access to known phishing forms, through whatever means are available. If the password collection form is on an otherwise legitimate high-volume site, like Google Docs or SurveyMonkey, it is best to use a filtering proxy or IPS that can act on the full URL. If that is not available, or if the site is used for no other purpose, then a DNS sinkhole (i.e., configuring the organization's DNS server to return wrong answers to requests for bad domains; the commercial OpenDNS service is often used for this) or even an IP address block at the border router should be considered.

Image referrer analysis

Sometimes, in order to appear more legitimate, the attackers will incorporate references to images stored on the target organization's web server into their phishing emails or password collection web sites. For example, here's a fragment from a recent phishing campaign targeting users at carleton.edu:

```
<body><div style='font-family: Arial; font-size: 12pt; color:
#000000'><img class="yiv3404942decoded"
src="http://apps.carleton.edu/global_stock/css/homepage/images/banner.g
if" height="90" width="508"><br><br>Dear User,<br><br>This
 is to notify all our Zimbra email user on our recent SQL Server
upgrade.
Kindly follow the instruction below to update your
account.<br><br><b><a
href="http://uksim.5gbfree.com/carleton.html">CLICK HERE</a> </b>to
begin.<br><br><br>Best Regard<br>Carleton College</div></body>
```

Figure 4 – HTML source of phishing email referencing the spoofed site's banner image

Both the phishing email, portions shown above, and the password collection site

behind the CLICK HERE, http://uksim.5gbfree.com, incorporate a banner from a legitimate site at the target organization, apps.carleton.edu. This creates traffic and log entries on carleton.edu servers, assisting in both retroactive investigation and live response. Retroactive investigation is discussed later, in section . In terms of response, Apache can be configured to send a different response if the referring web site is unexpected:

```
# Redirect requests for CarletonLogoSmall to a phishing warning graphic if the referring
# page is not one of ours. Note wrapped line at the end.
RewriteCond %{HTTP_REFERER} .
RewriteCond %{HTTP_REFERER}!^https?://(apps|wiki|www)\.carleton\.edu [NC]
RewriteRule ^/departments/ITS/Images/CarletonLogoSmall.png \
   http://www.carleton.edu/departments/ITS/Images/phish-banner.png [L,NC,R=307]
```

Figure 5 - Apache mod_rewrite rules to change image based on referrer

When a phishing victim follows the link to uksim.5gbfree.com, their web browser sees the IMG HREF to an image on apps.carleton.edu. When the browser requests the image, it includes “Referer: uksim.5gbfree.com” in the header. Code on apps.carleton.edu can see that header, act on it, and effectively change the look of the attacker’s web site:

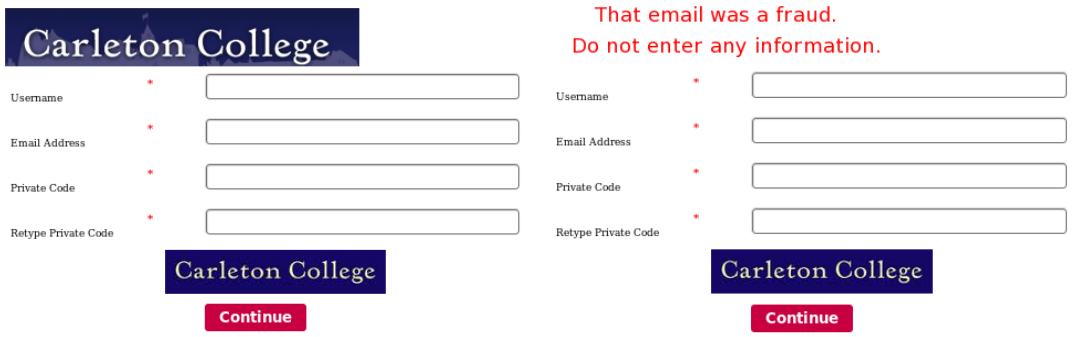

Figure 6 - phishing site before (left) and after (right) image rewrite

That’s great, but we would like to be more proactive. Jim Clark of the University of Chicago is credited with the concept and some code for monitoring a web log for “interesting” referrers, allowing incident responders to begin quarantine and takedown operations even before the first phishing email is received (Rossell, 2011). Below, we see a user on 41.151.70.7, in South Africa, coming from a phishing site that redirects to webmail.carleton.edu. If noticed in time, this information could have been used to sinkhole that site, or to create a spam filter for it, before any damage was done.

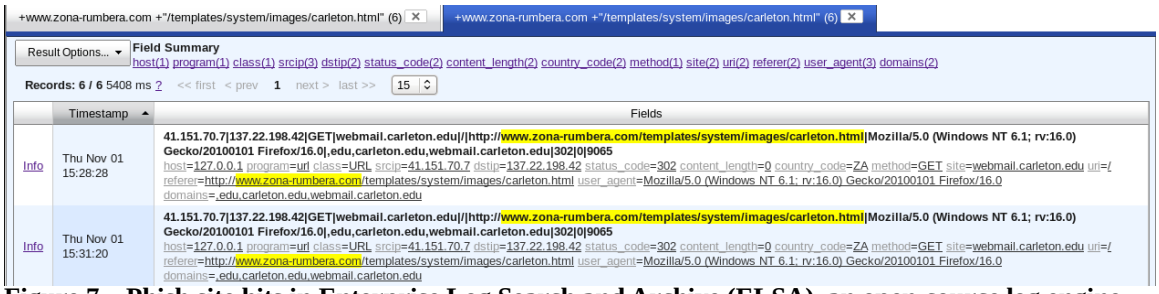

Figure 7 – Phish site hits in Enterprise Log Search and Archive (ELSA), an open-source log engine

Filtering Spam "At Your Leisure"

It is usually assumed that webmail providers have only one opportunity to classify messages as spam: at the point of delivery. This isn't true; providers could reexamine and reclassify messages up to the point that the user actually opens them (St. Sauver, J., 2006). As mentioned previously, URIBLs, Bayesian statistical classifiers, and fuzzy signature databases such as Pyzor and CloudMark improve with feedback. Therefore, a follow-up check is likely to catch some phish that was missed by the original delivery-time check. As far as we know, the concept of delayed or repeated spam checking is not widely deployed at the server side. Client-side tools do exist – Outlook, MacOS Mail, and Thunderbird all include Bayesian-style spam filters – but the number of users is small and falling (Kendall, E., 2006; Graves, R., 2002). This is an area for future research. However, the next, related, idea can be deployed today.

Purge reported phishing messages from inboxes

Webmail suites usually have a "Report Spam" button to leverage end-user assistance in alerting administrators and/or directly training server-side spam filters. In the Zimbra Collaboration Suite, this is implemented by sending a MIME-encoded copy of the original message to a central mailbox. Since 2008, this has worked not only for webmail, but also for desktop and mobile clients, by watching for moves to the folder named "Junk" (*IMAP/Outlook move to junk doesn't train anti-spam*, 2006-2010). Although the Zimbra developers intended the central spam mailbox only as a temporary holding area between runs of zmtrainsa (essentially a wrapper around sa-learn, the SpamAssassin Bayes database utility), we can use that corpus of reported spam for other purposes.

Appendix A and B contain stub source code for scripts to check a central spam reporting mailbox and, if more than a threshold number of distinct users have complained about the same envelope from address, to retroactively move all messages from that sender to Junk folders. In our experience, if a phishing message is sent to 1000 users

during business hours, it takes about 15 minutes for 5 people to report it as spam. This triggers the scripts, which move the other 995 copies to Junk folders. A high percentage of users never see the phish. The script sends the original RFC822 message source – which is often difficult to get from end users – to the system administrator along with notification that automated remediation has occurred. No human action needs to be taken to move the messages out of mailboxes.

The same concept should be applicable to systems other than Zimbra. Requirements include a way to receive spam complaints; a very fast way to find recent messages from the same sender, and if possible, with "substantially similar content"; and a programmatic way to Junk or delete those messages, as desired. In the Zimbra environment, we search mailbox.log to correlate senders to mail item IDs (roughly analogous to the file numbers in Cyrus or Maildir), and the provided zmmailbox utility to administratively manipulate the mailbox. A very similar script could certainly be developed for Cyrus IMAP; Microsoft Exchange, perhaps.

Identify and contain compromised accounts

Your user received a phishing email, and clicked on the link, and submitted his or her password. You have limited time to identify that this has occurred and to contain the damage.

Check logs for access to suspect phishing sites

Even if we retroactively remove phishing messages from inboxes, some people might have visited the phishing site in the interim. We need to search for access to the site, especially HTTP POSTed data. If all users were forced through a proxy or other device that logs web traffic, then those logs are the obvious place to look. In ISP and university environments with open borders, a tap port running httpry (http://dumpsterventures.com/jason/httpry/) or Bro (http://www.bro-ids.org/), logging to a system like ELSA (http://code.google.com/p/enterprise-log-search-and-archive/), will suffice. Here, we see a POST to the known phishing site uksim.5gbfree.com. Full-packet logging with Vortex-IDS and StreamDB (Holste, M., 2011) confirmed that the user had submitted a valid password. The user's compromised account was shut down.

Figure 8 - ELSA search for a known phishing URL finds POSTed data (source IP address elided)

In some cases, log searches can detect accesses to phishing sites that haven't been reported even once. For example, from 2009 to mid-2012, it was common for phishing gangs to abuse vulnerable phpFormGenerator installations, creating phishing URLs ending in form1.html (one instance from 2009: http://www.creative--solutions.com/phpFormGenerator/use/solutions/form1.html). Setting an alert on POSTs from such URLs was an effective way to find otherwise unknown phishes (Holste, M. 2012). In the future, if you notice a rash of phishing URLs on compromised WordPress installations, then you might want to tell your SEIM, or ELSA, or perhaps your IDS to raise an alarm on unexpected POSTs to /wp-admin/, /wp-content/, or whatever the common element might be.

If full URL query logging is not compatible with your local privacy policy, consider truncating URLs at the hostname, but always push for permission to log the method (GET versus POST). In most cases, a GET indicates a page view, but only a POST is evidence that the user might have disclosed their password. DNS query logging is better than nothing, but note that most current web browsers, in the interest of improving perceived performance, will issue DNS lookups for all hyperlinks immediately upon page load. Thus a DNS query log hit is merely an indication that a phishing email was viewed, not that the user clicked on the link, much less submitted data.

Detect unusual logins

When attackers log on to a compromised account, they tend to do so from an "unusual" location. Large parts of 41.* and 196.220.* are well known to mail administrators as West African sources of spam. Users can and do travel to West Africa, but if the same account logs on from Nigeria and the USA on the same day, that is *probably* a sign of compromise. (VPNs and cellular networks complicate matters.) Some years ago, Columbia University came up with the Grand Unified Logging Program (Selsky, M., & Medina, D., 2005) to normalize heterogeneous sources of login data and detect anomalous logins. Any SEIM worth its salt should be able to run GULP-like searches for anomalous logins. Read that report; investigate and whitelist "known exceptions" so that the report contains only actionable data; and contain compromised

accounts as described below.

Detect changes to webmail preferences

When a phishing gang compromises an unprivileged webmail account, they're often going to want to send phish or spam. To begin, they usually make some changes to preferences. They set the preferred name to "Help Desk" or "UK Lottery." They set forwarding and reply-to addresses. They might disable "Save to Sent Folder" and "Remember New Contacts" features. They set the compromised account's signature to the body of the spam message they want to send.

These actions can be detected. At Carleton College, we detect 80% of all successfully phished accounts by searching Zimbra's LDAP store once per minute for the phishing indicators below. Accounts that match are then locked; a help desk ticket is created; the IP address used to change the preferences is identified; the Zimbra audit log is searched for other accounts used from the same IP address; and the previous account preferences are restored using Zimbra's zmprov utility, all without human intervention.

```
(&(|(zimbraPrefReplyToAddress=*globomail.com)
(zimbraPrefReplyToAddress=*reminder.com)
(zimbraPrefReplyToAddress=*skymail.mn)(zimbraPrefReplyToAddress=*kimo.com)
(zimbraPrefFromDisplay=*help*)(zimbraPrefFromDisplay=*Service*Provider)
(zimbraPrefFromDisplay=*Reminder*)(zimbraPrefReplyToAddress=*reply*)
(zimbraPrefMailForwardingAddress=*reply*)
(zimbraPrefMailForwardingAddress=*kimo.com)
(zimbraPrefMailForwardingAddress=od@*)
(zimbraPrefMailForwardingAddress=no@*)(zimbraprefreplytoaddress=no@*)
(zimbraPrefFromDisplay=web*)(zimbraPrefFromDisplay=*admin*)
(zimbraPrefNewMailNotificationAddress=*reply*)(zimbraPrefSaveToSent=FALSE)
(zimbraPrefMailSignatureHTML=*sir/madam*)
(zimbraPrefMailSignature=Hello*My*)
(zimbraPrefMailSignatureHTML=*million*us*)
(zimbraPrefMailSignatureHTML=*good*day*)(zimbraPreffromDisplay=*lottery*)
(zimbraPrefReplyToDisplay=*lottery*))(zimbraAccountStatus=active))
```

Figure 9 - LDAP query filter to find compromises via changes to webmail preferences

This idea could probably be implemented in other systems. Find the database where these preferences are stored and monitor changes.

Enforce outbound rate limits

After the above, our most effective way of detecting the abuse of compromised webmail accounts is per-user outbound rate limits. Allow any given account to send fifty messages per minute, for example, and reject/alert on anything beyond that level. (This seems very high, but spammers will try to send hundreds per minute, so there should be few false negatives. You do not want false positives.) In the open source realm, Mailfromd (http://www.gnu.org.ua/software/mailfromd/) and Policyd (http://www.policyd.org/) are used to rate limit in this manner. Be sure to configure and

act on alerts when the rate limit policy is violated. Technology that silently slows your outbound spam rate to a trickle is better than nothing, but you really want to identify and stop the abuse entirely.

Detect downstream antispam deferrals

When Yahoo, AOL, and Hotmail get a large number of spammy messages from an IP address, they start deferring messages from that source. This leaves telltale signs in the logs. In sendmail, they match patterns like:

```
stat=Deferred.+http://postmaster.yahoo.com
stat=Deferred.+http://postmaster.info.aol.com
stat=Deferred.+mail.live.com/mail/troubleshooting.aspx
```

Appendix C is a simple but effective script to look for evidence that major ISPs are throttling connections, to identify the top user and defer any more messages from that source, to clear the queue, and to notify the system administrator.

Anti-Phishing-Email-Reply project (APER)

Email administrators discovered some time ago that although spammers and phishing gangs cycle through an endless supply of email accounts, often stolen from phishing victims, the outbound spam has a Reply-To address chosen from a smaller, more stable set. In addition, the first message sent from a compromised account often goes to the gang's "real" email address. The Anti-Phishing-Email-Reply project (http://code.google.com/p/anti-phishing-email-reply) maintains a list of addresses useful for identifying phishing victims. As an example of how this list is applied, consider a phishing campaign that hit Carleton College in December 2012. One of our users received a phishing email (i.e., it got through all antispam filters) and chose to submit her password to a crude phishing form tacked on to a compromised WordPress blog. An hour later, someone in Nigeria (41.203.67.50) configured an Outlook Express client for our authenticated relay and sent a test message:

```
Dec  6 07:35:18 sendmail[15144]: AUTH=server, relay=[41.203.67.50],
authid=********, mech=LOGIN, bits=0
Dec  6 07:35:21 sendmail[15144]: qB6DZFoA015144: X-Mailer:  Microsoft Outlook
Express 6.00.2600.0000 From: gdfs@metu.edu.tr
Dec  6 07:35:21 sendmail[15144]: qB6DZFoA015144: from=<gdfs@metu.edu.tr>,
size=611, class=0, nrcpts=2,
msgid=<201212061335.qB6DZFoA015144@smtp.carleton.edu>, proto=ESMTP,
daemon=TLSMTA, relay=[41.203.67.50]
Dec  6 07:35:22 sendmail[15171]: qB6DZFoA015144:
to=<greatcbz14life@hotmail.com>, delay=00:00:02, xdelay=00:00:01, mailer=esmtp,
pri=150611, relay=mx1.hotmail.com. [65.55.92.168], dsn=2.0.0, stat=Sent
( <201212061335.qB6DZFoA015144@smtp.carleton.edu> Queued mail for delivery)
Dec  6 07:35:24 sendmail[15171]: qB6DZFoA015144: to=<ojo4live@yahoo.com>,
delay=00:00:04, xdelay=00:00:02, mailer=esmtp, pri=150611,
relay=mta7.am0.yahoodns.net. [98.136.217.202], dsn=2.0.0, stat=Sent (ok dirdel)
```

Figure 10 - /var/log/maillog lines relating to the first use of a stolen account (authid elided)

The address ojo4live@yahoo.com was known to APER from previous spam runs. This raised an alarm, and we shut down the account responsible. The results of a Google search for ojo4live@yahoo.com are interesting:

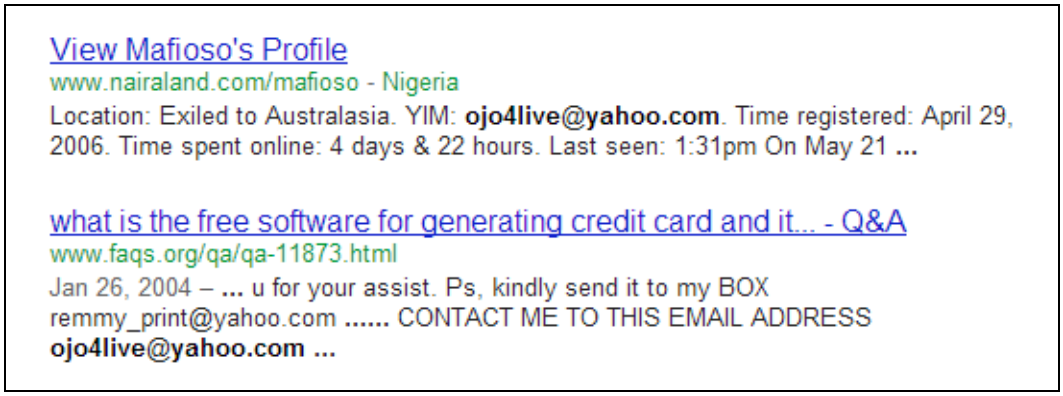

Figure 11 - Google search results for Nigerian phish-and-scam contact address

He chose to call himself "Mafioso." He was trying (rather ineptly) to get involved in credit card fraud in 2006. This is not a person with whom we wish to communicate. The spam he wished to send (which we quarantined) was:

```
Received: from User ([41.203.67.50])
        (authenticated bits=0)
        by smtp.carleton.edu (8.14.4/8.14.4) with ESMTP id
qB6FlTjQ026402
        (version=TLSv1 cipher=DHE-RSA-AES256-SHA bits=256 verify=NOT);
        Thu, 6 Dec 2012 09:47:33 -0600
Reply-To: <yee.leegrace4@yahoo.com.hk>
From: "Grace Lee Yee"<********@carleton.edu>
Subject: Genuine Deal.Interested?
X-Mailer: Microsoft Outlook Express 6.00.2600.0000
X-Antivirus: avast! (VPS 121206-1, 12/06/2012), Outbound message
X-Antivirus-Status: Clean

Deal worth USD45,275, 000.00. Are you interested? Please get back for
more details.via email: yee.leegrace1@zing.vn
```

Figure 12 - Advance fee fraud (419 scam) via authenticated SMTP submission (email elided)

This is an international affair. Based on his comments on Nigerian politics in the airaland forums, "Mafioso" is clearly Nigerian, but he says he was "exiled" to Australasia (*View Mafioso's Profile*, 2013). The Reply-To is Yahoo Hong Kong. The drop box for the scam is a Vietnamese social networking and free email site, though "Grace Lee Yee" could easily be Nigerian.

In his discussion of Nigerian 419 scams, Herley, C. (2012) suggests, counter-intuitively, that the very crudeness of the message helps. Only an idiot would submit his or her password to an obvious phishing site or hand bank account details to an obvious scammer. The phish-and-scam gangs pre-screen for idiocy by sending messages that aren't very convincing. An alternative explanation is that they are doing their best with

limited talent. Either way, it is clear that their campaigns are quite prolific. Alas, some of them succeed.

Conclusion

At each stage of a phishing attack, the techniques discussed in this paper will reduce the likelihood and impact of resulting account compromises. The (apparently) Nigerian phish-and-scam gangs are remarkably persistent. End-user security awareness is probably the best defense, but they are relentless; defense in depth is required. Most of these techniques are "free," but take time and effort. Concentrate on those with the lowest cost and highest expected payoff. Monitoring webmail preferences for indications that spam content is being staged in signatures is easy and effective. The various forms of log collection and analysis (i.e. finding "unusual" logins, data-mining web proxy or httpry logs for phishy web sites) are expensive in time, dollars, and potential privacy exposure, but are likely to pay off operationally. Definitely investigate ways to remove previously delivered messages from your email store. Phishing is never going to go away, but the risk can be managed.

This paper was primarily written for email service providers and enterprise email administrators. It did not discuss the growing trend towards outsourced email. Most recommendations still apply to an outsourced environment, but responsibilities will be divided between the service provider and the contracting organization. Communication and log analysis across organizational boundaries can be challenging. That's a topic for another paper.

References

Bergholz, A., De Beer, J., Glahn, S., Moens, M., Paaß, G., & Strobel, S. (2010). New filtering approaches for phishing email. *Journal of Computer Security, 18*(1), 7-35.

Chan, J. (2004-2012). *Surbl*. Retrieved from http://www.surbl.org/

Downs, J. S., Holbrook, M. B., & Cranor, L. F. (2006, July). Decision strategies and susceptibility to phishing. In *Proceedings of the second symposium on Usable privacy and security* (pp. 79-90). ACM.

Google. (2012, December 27). *Ojo4live@yahoo.com – Google Search*. Retrieved from https://www.google.com/search?q=ojo4live@yahoo.com

Graves, R. (2002). *UNet 2002 Survey Results*. Retrieved from http://www.brandeis.edu/mybrandeis/archive/bboard/q-and-a-fetch-msg%3Fmsg_id=0000U4

Herley, C. (2012). Why do Nigerian Scammers Say They are from Nigeria? In *Proceedings of the Workshop on the Economics of Information Security*

Holste, M. (April 2011). *Lizamoon: Knowing is half the battle*. [Web log message]. Retrieved from http://ossectools.blogspot.com/2011/04/lizamoon-knowing-is-half-battle.html

Holste, M. (March 2012). *Detection is the new prevention*. [Web log message]. Retrieved from http://ossectools.blogspot.com/2012/03/detection-is-new-prevention.html

Hypponen, M. (2011, August 26). [Web log message]. Retrieved from http://www.f-secure.com/weblog/archives/00002226.html

IMAP/Outlook move to junk doesn't train anti-spam (2006-2010). Retrieved from https://bugzilla.zimbra.com/show_bug.cgi?id=9532

Kendall, E. (2006). *2006 UNet Survey Results*. Retrieved from http://www.brandeis.edu/mybrandeis/archive/bboard/q-and-a-fetch-msg%3Fmsg_id=00051l

Rossell, Shelley (2011, April). *Phishing technical controls: beyond Proofpoint.* Presentation delivered at the EDUCAUSE Security Professionals Conference, San Antonio, Texas.

Selsky, M., & Medina, D. (2005, December). GULP: a unified logging architecture for authentication data. In *Proceedings of the 19th conference on Large Installation System Administration Conference-Volume 19* (pp. 1-5). USENIX Association.

Spransy, Derek (2012, May). Phishing ourselves to raise awareness. Presentation delivered at the 10th EDUCAUSE Security Professionals Conference, Indianapolis, Indiana. Retrieved from http://www.educause.edu/events/security-professionals-conference/phishing-ourselves-raise-awareness

St Sauver, J. (2006, June). *Filtering spam "at your leisure:" a MAAWG briefing/look ahead*. Presentation delivered at the 7th general meeting of the mail anti-abuse working group,

Brussels, Belgium. Retrieved from http://www.uoregon.edu/~joe/maawg7

View Mafioso's Profile (2013, January 8). Retrieved from http://www.nairaland.com/mafioso

WritingRules. (2011, December 31). Retrieved from http://wiki.apache.org/spamassassin/WritingRules

Appendix A
check-spam-complaints.pl

```
#!/usr/bin/perl -Tw

# If there have been more than 5 spam complaints about the same sender in
# a 24-hour period, retroactively junk all messages from that sender.
# The state files in /var/cache/spamscore are cleaned daily by logrotate.

use Mail::IMAPClient;
use DB_File;

use strict;

my %last_user;
my %complainers;
my %complaint_count;
my %already_censored;

tie %last_user, 'DB_File', '/var/cache/spamscore/last_user.db';
tie %complaint_count, 'DB_File', '/var/cache/spamscore/complaint_count.db';
tie %complainers, 'DB_File', '/var/cache/spamscore/complainers.db';
tie %already_censored, 'DB_File', '/var/cache/spamscore/already_censored.db';

sub alertmail {
  my ($subject,$body) = @_;
  open SENDMAIL, "| /usr/sbin/sendmail -oi -t";
  print SENDMAIL "From: root\@example.com\nTo: root\@example.com\nSubject:
       $subject\n\n$body\n";
  close SENDMAIL;
}

my $imap = Mail::IMAPClient->new(
           Server   => 'mail.example.com',
           User     => 'spam.XXXXXXX@mail.example.com',
           Password => 'XXXXXXXXXX',
           Ssl      => 1,
           Uid      => 1,
);

$imap->select('inbox')
           or die "Select 'inbox' error: ", $imap->LastError, "\n";

foreach my $msgid ($imap->search("recent")) {
  my $headers = $imap->parse_headers($msgid,"Subject");
  next unless ( $headers->{"Subject"}->[0] =~ /zimbra-spam-report: ([a-zA-Z0-
       9\.\-\_]+)\@example.com: spam/);
  my $user = $1;
  my $part2 = $imap->bodypart_string($msgid,2,7800,0) or die "Could not get
       bodypart string: ", $imap->LastError;
  if ($part2 =~ /^Return-Path: <?([A-Za-z0-9\.\=\-\*\_\+]+\@[A-Za-z0-9\.\-]+)>?
       [\r\n]/) {
    my $spammer = $1;
    next if ($spammer =~ /\=/);
    next if ($already_censored{$spammer});
    next if (defined $last_user{$spammer} and $last_user{$spammer} eq $user);
    $last_user{$spammer} = $user;
        next if (defined $complainers{$spammer} and $complainers{$spammer}
      =~ /\b$user\b/);
    $complainers{$spammer} .= " $user";
```

```
    #print "$1\n";
    $complaint_count{$spammer}++;
    if ($complaint_count{$spammer} > 4) {
      $already_censored{$spammer}++;
      # We don't censor ourselves or other whitelisted addresses.
      if ($spammer =~ /example.com$/) {
        alertmail "alleged spammer $spammer NOT censored", "Who complained:
       $complainers{$spammer}\n$part2";
        next;
      }
      close STDERR;
      open CENSOR, "/usr/local/bin/spam-censor-by-email.pl $spammer |";
      my @censored = <CENSOR>;
      close CENSOR;
      # The body of the mail to the sysadmin includes all the accounts
      # and mail item IDs, so the action can be reversed if necessary.
      alertmail "alleged spammer $spammer censored", "Who complained:
       $complainers{$spammer}\n$part2" . join("\n",@censored);;
    }
  } else {
    alertmail "could not parse spam complaint", "$part2";
  }
}

$imap->logout
           or die "Logout error: ", $imap->LastError, "\n";
```

Appendix B
spam-censor-by-email.pl

```
#!/usr/bin/perl

if ($ARGV[0] =~ /\@/) {
  $email = $ARGV[0];
} else {
  print "This script purges any email received from a given address.\n";
  print "Enter email address to purge: ";
  $email = <STDIN>;
  chop $email;
}
$email =~ s/\@/\\\@/ or die "no \@ sign in $email\n";
if ($> = 670) {
  open ZMMAILBOX, "| /opt/zimbra/bin/zmmailbox -z";
} else {
  open ZMMAILBOX, "| sudo su - zimbra -c '/opt/zimbra/bin/zmmailbox -z'";
}
open LOG, "< /opt/zimbra/log/mailbox.log" or die "canot open mailbox.log";
while (<LOG>) {
  if (/LmtpServer.+Delivering message: size=.+sender=$email, msgid=(<.+>)/) {
    $id{$1} = 1;
    warn "will junk msgid $1\n"
  } elsif (/LmtpServer.+\[name=([a-zA-Z0-9]+)\@example\.com;mid=.+mailop -
Adding Message: id=([0-9]+), Message-ID=(.+), parentId.+folderName=.+/ and
$id{$3}) {
    print ZMMAILBOX "sm $1\nmm $2 /Junk\n";
  }
}
close LOG;
close ZMMAILBOX;
```

Appendix C
deferred-spam-detect.sh

```
#!/bin/sh
export PATH=/bin:/usr/bin

# /etc/mail/deferred-spam-patterns contains regular expressions such as:
# stat=Deferred.+http://postmaster.yahoo.com
# stat=Deferred.+http://postmaster.info.aol.com
# stat=Deferred.+mail.live.com/mail/troubleshooting.aspx

test `tail -400 /var/log/maillog|egrep -c -f /etc/mail/deferred-spam-patterns`
-gt 10 || exit 0

TOPSPAMMER=`/usr/bin/mailq|awk '/^[a-zA-Z0-9]+-? +.+<.+\@.+>/ {print $7}'|sort|
uniq -c|sort -nr | head -1 | perl -ne 'print if s/\s+[0-9]{2,6}\s+<(.+\@.+)>.*/
$1/'`

test -z "$TOPSPAMMER" && exit 0

/sbin/service sendmail stop > /dev/null 2> /dev/null
killall sendmail
echo "From:$TOPSPAMMER ERROR:4.7.1:\"421 Your address is temporarily restricted
due to external complaints\"" >> /etc/mail/access
/usr/sbin/makemap hash /etc/mail/access < /etc/mail/access

mkdir /var/spool/mqueue/$TOPSPAMMER
mv /var/spool/mqueue/* /var/spool/mqueue/$TOPSPAMMER 2> /dev/null
killall sendmail
/sbin/service sendmail start > /dev/null 2> /dev/null

date|mail -s "Likely spammer $TOPSPAMMER tempfailed at $
(hostname):/etc/mail/access by $0" page-the-sysadmin@example.com
```