Honeytokens and honeypots for web ID and IH

Rich Graves

Honeypots and Honey Tokens for Webmail ID/IR

GIAC GCIA Gold Certification

Author: Rich Graves, rcgraves@gmail.com
Advisor: Mark Stingley

Accepted: May 13th 2015

Abstract

Honeypots and honey tokens can be useful tools for examining follow-up to phishing attacks. In this exercise, we respond using valid email addresses that actually received the phish, and wrong passwords. We demonstrate using custom single sign-on code to redirect logins with those fake passwords and any other logins from presumed attacker source IP addresses to a dedicated phishing-victim web honeypot. Although the proof-of-concept described did not become a production deployment, it provided insight into current attacks.

1. Introduction

Trends in attack patterns and outsourcing are changing the work of the enterprise intrusion analyst. Skills in packet analysis and intrusion detection system (IDS) alert correlation are still relevant to service providers and locally hosted systems, but increasingly, services are "in the cloud," well outside the analyst's reach. Packet capture is out of the question, and even basic account login data can be difficult to extract. The Software Defined Perimeter (Cloud Security Alliance, 2015) is an intriguing approach to recreate security boundaries on potentially untrusted networks, somewhat reminiscent of Windows Domain isolation with IPSec (Morey, 2005), but both remain unrealistic for small to medium enterprises that simply make use of public services. Imagine an organization that uses Salesforce for CRM, Workday for HR and Finance, Gmail for email, etc. In such an organization, what does an intrusion look like? Are there any remaining chokepoint(s) that an analyst can instrument and watch?

There is little hope of deep visibility into externally hosted web applications. They simply must be trusted – and contractually verified, at least to the minimal level demanded by standards such as PCI DSS 12.8.2-12.8.5 (PCI Security Standards Council, 2015, pp. 106-107). However, users still need to authenticate to the application. If authentication is provided by a single sign-on system under the enterprise's control, then the date, time, User-Agent, IP address (hence geolocation), username, and credentials can be logged and analyzed centrally. In higher education, systems such as GULP have been doing such work for many years (Selsky & Medina, 2005). Ideally, that may be all the information that is necessary to detect compromises due to password compromise. According to the Verizon Breach Report, phishing remains a major threat, playing a role in 2/3 of the incidents investigated. Most web application attacks start with stolen credentials (Verizon, 2015, pp. 12,42). If those logins can be refused based on the attacker's unusual source address, then those attacks simply go away.

If only it were as simple as identifying "unusual source addresses." Users are more mobile than ever. They use multiple devices, from a variety of work, cellular, home, and casual WiFi networks. They travel overseas, to countries that might be

considered "suspicious." From March 19 to April 18, 2015, at least 406 Carleton College users (out of a total 3,000 users) logged on to webmail from 29 different countries including some in Africa and Asia. Below is a Splunk visualization of the geographic distribution.

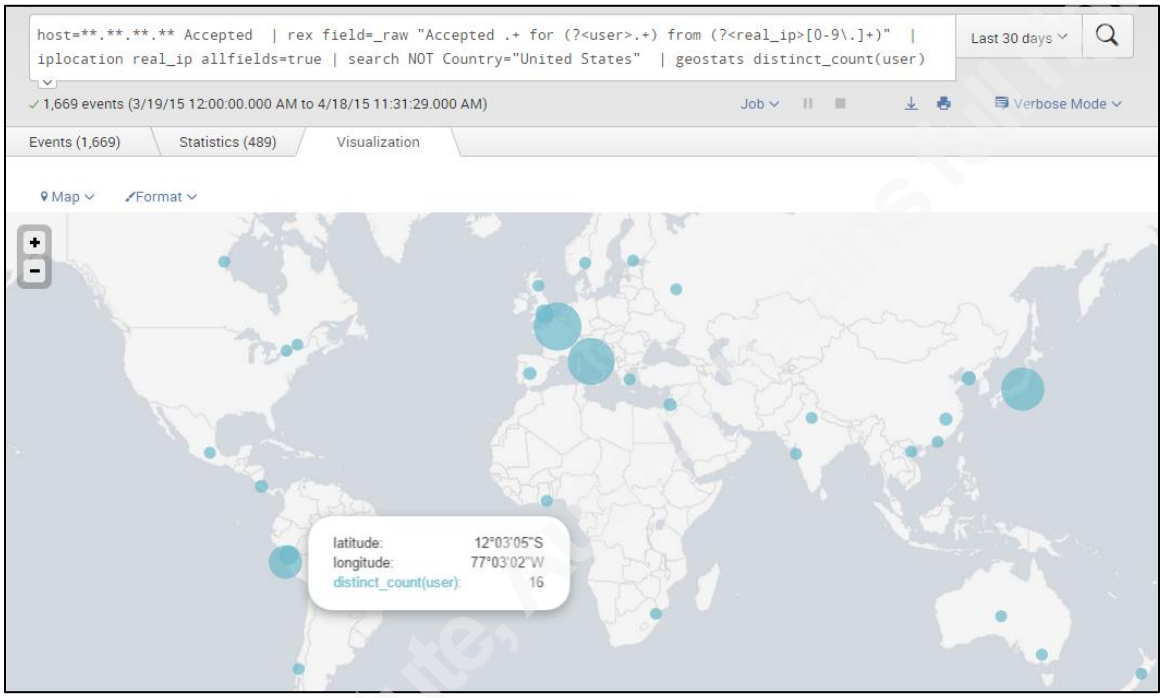


Figure 1 - Splunk search for logins from non-US IP addresses. 16 distinct visitors to Peru, all legitimate, highlighted.

Source IP address, concomitant with geolocation, is merely one potential indicator that behind a webmail login is an attacker with stolen credentials. As with any other intrusion detection decision, we want to reduce false positives and false negatives. The challenge, in the absence of widespread acceptance of multifactor authentication (Graves, 2014), is to combine IP source information with other factors including browser user-agent, changes to webmail account preferences (Graves, 2013, p. 14), and attempted spamming activity. Ideally, we want a feedback loop, where past experience with non-IP indicators of compromise influences an IP or network's reputation, and continued abuse from a given set of indicators increases confidence in the decision to open an incident. In this paper, we explore the use of an instrumented web single sign-on system to redirect suspected attacker logins to a high-interaction webmail honeypot system. Some history of

Rich Graves, rcgraves@gmail.com

honeypots is discussed; the setup of the honeypot is outlined; and attacker behavior is observed.

The overall flow of the proposed system is illustrated in Figure 2. A user hits the login page and attempts to authenticate. If the password is valid and their source address (and perhaps other characteristics) pass muster, then they are securely redirected to the desired application server. If a valid password is entered from a suspicious location, the user is redirected to the honeypot. If an invalid password that matches a predetermined pattern is entered, the user is redirected to the honeypot. Finally, if an invalid password that does not match the pattern is entered, the user sees a generic "username/password is invalid" message and is invited to try entering their password again, until account lockout occurs. Any activity within the honeypot is logged for further review. Activity within the real webmail server is also scrutinized for suspicious activity, as discussed in Phishing Defenses for Webmail Providers (Graves, 2013).

Rich Graves, rcgraves@gmail.com

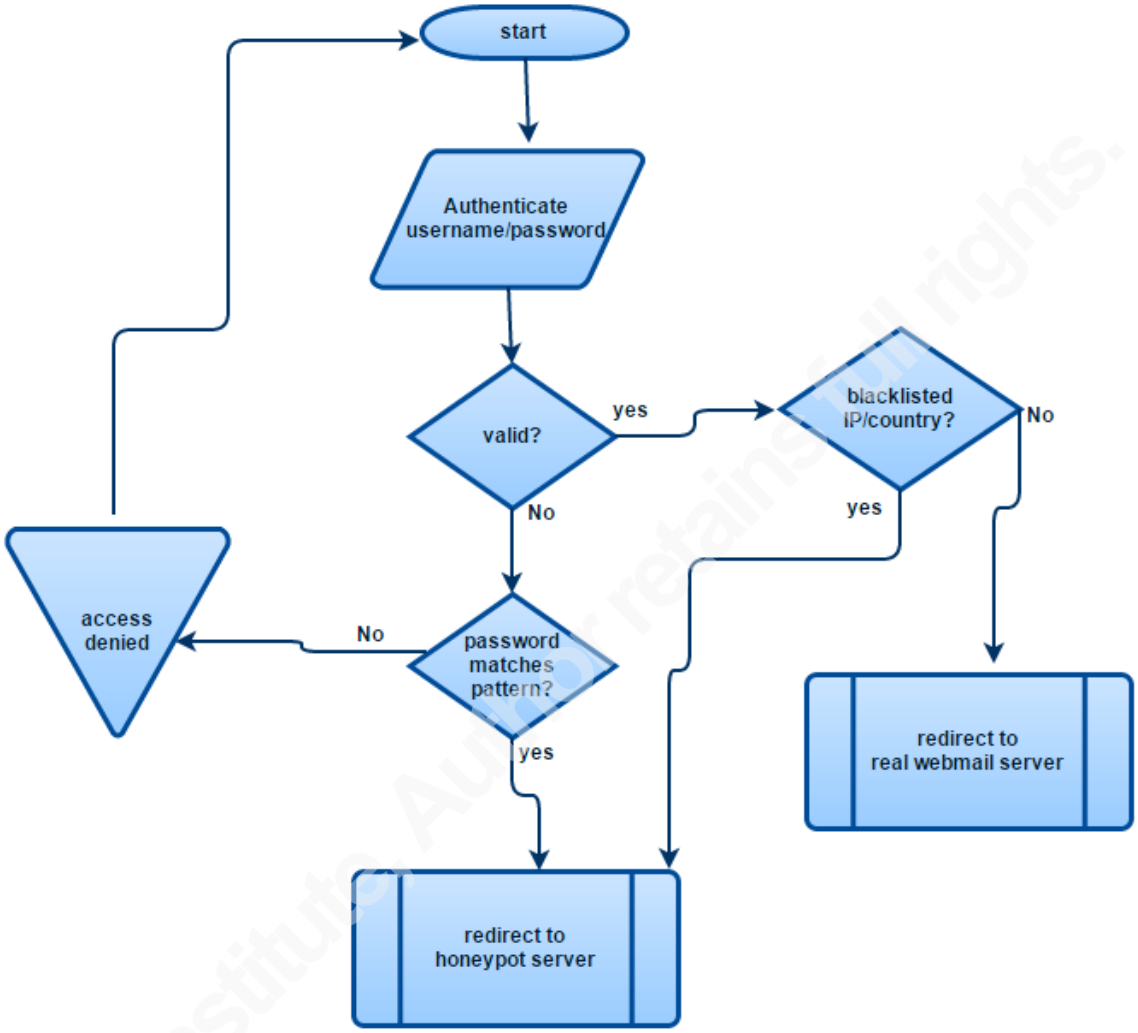


Figure 2 - Honeypot sign-on redirector logic flow

With so many more interesting services out there, why the focus on webmail? Although the likely damage from webmail compromise is low compared to institutional banking (REN-ISAC, 2010) and direct-deposit payroll theft (REN-ISAC, 2014), the rate of attacks on higher-value targets is low and unpredictable. In contrast, phishing emails arrive in spam folders, if not inboxes, nearly every day. Phishing gangs, many of Nigerian origin (Krebs, 2013), are usually diligent and prompt in the use of any credentials given to them. This project took advantage of the ready availability of webmail phishing attacker labor. The techniques and lessons learned can be applied to the protection of any globally available web site. The logic in Figure 2 could be applied to a more sensitive business application or a general-purpose single sign-on system.

Rich Graves, rcgraves@gmail.com

2. A natural experiment in webmail honeypots

Generally speaking, a honeypot is a computer system designed to facilitate the observation of attacker behavior. The many types of honeypots are discussed in Spitzner (Spitzner, 2002). Honeypots can also arise by accident. Over the past two decades, most higher education institutions have migrated email from open-source solutions like Dovecot, Cyrus, and UW-IMAP to proprietary solutions like Exchange and, more recently, cloud providers including Gmail and Office 365 (Mills, 2011). Some discovered that this created a "natural experiment," where legitimate users moved to the new solution, but spammers continued to follow old links to the retired webmail server. The University of Auckland intentionally left their old webmail server online for years as a honeypot, delivering valuable intelligence to the higher education community (Russell Fulton, personal communication, 2015). The Auckland experience was a primary inspiration for this paper.

3. Other related work: phish form stuffing

Since phishing is such a major scourge on the Internet, it has attracted much study. One approach to detecting and deterring attackers is called “form-stuffing.” When phishing emails are received, some process responds with bogus passwords or credit card numbers, as appropriate. An online commenter from 2006 wrote, “Ohhh, and I believe there are already commercial operations that offer distributed, automated fake form-stuffing (among other things) as part of their ‘anti-phishing’ services (and some of them may have filed patents on (variations of) this idea)” (FitzGerald, 2006). The term “honey token” is now used to refer to fake credentials intended to lure, confuse, or overwhelm attackers. The PhiGARo project at Masaryk University implemented a complete network of honeypots to collect phishing emails, respond with honey tokens, and integrate with an integrated intrusion detection system to detect innocent users who responded to the phish (Husak & Cegan, 2014). PhiGARo appears to deliver everything that this project considered, and more; however, the system requirements are very steep. Humboldt 2.0 (Gustafson & Li, 2013) is designed as a scalable solution to the problem of distributing plausible honey tokens. It is intended to defeat the attacker’s efforts to filter out the chaff of honey tokens so as to economize on only “real” credentials. The Phish Feeder Firefox

extension was another academic experiment in enlisting the help of users to stuff phishing forms with honey tokens (Lynch, 2009). Chaff is a new command-line tool to parse and stuff phishing forms with convincing-looking credentials. (Deleon, 2015) What is missing from the literature is a cookbook for a small organization, possibly one that has fully or partially outsourced email, to play the honeypot + honey token game.

4. Implementing a honey token redirector with Zimbra single sign-on

Zimbra is an advanced email and calendar system roughly on par with Microsoft Exchange (Zimbra, Inc., 2015). Although most colleges and universities are migrating to "cloud" email, it was popular for a time and remains in use at Stanford University and elsewhere. After several years of managing Zimbra on local servers and storage, the author migrated Carleton College email to a Merit Networks-hosted service (MeritMail, 2015). The instructions below apply regardless of whether Zimbra is hosted locally or remotely. Although the details are specific to a Zimbra webmail server, think about how the general framework illustrated in Figure 2 could be applied to any internal or cloud service that supports SAML, OpenID, or proprietary single sign-on via browser redirection.

4.1. Isolate the honeypot server

A high-interaction research honeypot needs to be isolated from production networks (Spitzner, 2002, p. 33). Don't let your tool for examining attacker behavior be turned against you. There is no necessary communication between the honeypot server and the real enterprise servers.[1] The sign-on server passes signed authentication assertions only by way of a browser redirect; they never need to talk to each other. The sign-on server should be located in a DMZ, and the honeypot could be located on a separate DMZ or, better, off the enterprise network entirely, perhaps as an Amazon AWS host.

[1] For the demonstration at Carleton College, the honeypot server was given limited read-only access to a stub LDAP server was permitted to allow more realistic account creation. This optional feature is discussed in sections 4.4-4.5.

Because the primary aim of the phishing gangs is to send spam, steps must be taken to ensure that the honeypot server can't do that. A simple approach is to block outbound TCP 25 (iptables -I OUTPUT -p tcp -m state --state NEW -m tcp --dport 25 -j REJECT). The only inbound traffic required is to TCP 443, and perhaps from a limited set of source addresses on your enterprise network to SSH on TCP 22.

4.2. Install Zimbra

The basic open-source Zimbra server is freely available for Ubuntu and RHEL/CentOS from https://www.zimbra.com/downloads/zimbra-collaboration-open-source. The license terms are a combination of the GNU General Public License Version 2 and a derivative of the Mozilla Public License. There are no restrictions on commercial use, provided you do not redistribute modified code (Zimbra, Inc., 2014). All the default options are reasonable for this purpose. About 10GB disk space, 3GB RAM, and one CPU core are needed.

4.3. Configure "preauth" for Zimbra login

Zimbra servers can authenticate to a local password store, LDAP, or Kerberos. This won't work for the honeypot because it won't be using production passwords. Zimbra also supports a simple Preauth scheme that authenticates users to the web interface with a transient ticket including a SHA1 HMAC of the subject's email address, a pre-shared secret key, and the time (Holder, 2015). The command-line interface for most domain and account setup tasks is zmprov. First create a random pre-shared key with gdpak (the output must be copied to the login server, see section 4.5), then define the URLs for the externally hosted login and logout pages.

```
[zimbra@zmail ~]$ zmprov
prov> gdpak example.com
preAuthKey: 4e2816f16c44fab20ecdee39fb850c3b0bb54d03f1d8e073aaea376a4f407f0c
prov> modifyDomain example.com zimbraWebClientLogoutURL \
                               'https://login.example.com/sso/?logout'
prov> modifyDomain example.com zimbraWebClientLoginURL 'https://login.example.com/sso/'
```

In operation, the Preauth scheme is conceptually similar to Kerberos, and thus shares some theoretical vulnerabilities. The bearer token passed in the browser redirect remains valid for a few minutes to allow for clock skew. It is not bound to the client IP address or session and does not appear to have a Kerberos-style replay cache (MIT, 2015), so an attacker who can steal a token can log on. Although the 256-bit pre-shared key is robust (assuming good sources of randomness), if it is exposed by either system, login as arbitrary users is possible. Zimbra has nascent OpenID and SAML support that partially mitigates these issues, but they are not necessarily stable enough for production use. We recommend understanding and accepting the (low) risk here. Note that the honeypot server will have a different pre-shared key than the real server, so any compromise of the former will not affect the latter.

4.4. Configure "lazy" account provisioning

The Preauth process takes care of authentication, but each user still needs to be defined in Zimbra. There are at least four ways this could be handled. First, just use a single account. No matter which username is entered by the attacker, always sign on as the same honeypot user. This is undesirable because it makes it more difficult to distinguish artifacts created by overlapping logins. Second, pre-create accounts for every possible user. This can be scripted with zmprov, but carries an ongoing maintenance burden. The solution we eventually chose was to use Zimbra's auto provisioning feature (Zimbra, Inc., 2015) to spawn accounts upon first login. A prerequisite for auto provisioning is that the Zimbra server must be able to contact an LDAP server (ActiveDirectory, ADAM, OpenLDAP, etc.) to confirm that an account should exist and to fetch attributes such as full name. In theory, this requirement could be satisfied by another layer of auto provisioning: run a honeypot LDAP server that creates LDAP entries on demand. One way to do this would be to overload a few methods in perl Net::LDAP::SimpleServer (Znamensky, 2012) so that any entry searched for, is affirmed to exist. For a permanent deployment, this should be done to facilitate complete isolation between honeypot and production systems. For expediency in this proof of concept, however, the honeypot was given an unprivileged read-only account and access through the firewall to one of Carleton College's production LDAP servers.

Rich Graves, rcgraves@gmail.com

The following commands tell the Zimbra to spawn new accounts automatically as needed, and to send an email with the specified subject and body to each new account. That email helps with false positives. If a legitimate user (not an attacker in possession of a stolen password) is redirected to the honeypot server, we want clear communication.

```
zmprov md example.com zimbraAutoProvAuthMech PREAUTH \
 zimbraAutoProvLdapURL ldap://ldap.example.com zimbraAutoProvLdapStartTlsEnabled TRUE \
 zimbraAutoProvLdapAdminBindDn  uid=UnprivilegedReader,ou=People,dc=example,dc=com \
 zimbraAutoProvLdapAdminBindPassword  SuperSecret \
 zimbraAutoProvLdapSearchBase 'ou=People,dc=example,dc=com' \
 zimbraAutoProvLdapSearchFilter '(&(objectClass=account)(uid=%u))'

zmprov md example.com zimbraAutoProvNotificationSubject 'Welcome to ZMail' \
 zimbraAutoProvNotificationFromAddress helpdesk@example.com \
 zimbraAutoProvNotificationBody 'For assistance, contact the help center at 867-5309'
```

4.5. Prepare a sign-on web page that forks for honey tokens

With all that preparation in place, we finally come to the code that decides whether to send the user to the honeypot or the real production server. For brevity, the only detection techniques illustrated here are checking whether the source IP address is in Nigeria or the password matches a honey token pattern (sections 3 and 5.1). For various_other_critera() for considering a login suspicious, see (Graves, Phishing Defenses for Webmail Providers, 2013). The Collective Intelligence Framework (collectiveintel.org, 2013) can also be helpful for sharing threat data and coordinating phishing mitigation efforts across organizations.

Rich Graves, rcgraves@gmail.com

```
define("LOGIN_OK",1);
define("HONEYPOT",2);
define("FAILED",3);

function ldap_authed ($user,$pass) {
  openlog('login-mail',LOG_PID | LOG_ODELAY,LOG_AUTHPRIV);
  $ldap_conn = @ldap_connect("ldaps://ldap.example.com/");

  // requires php-pecl-geoip and MaxMind GeoLite databases
  $geoip = @geoip_record_by_name(GetEnv('REMOTE_ADDR'));
  if (isset($geoip) && isset($geoip['country_code'])) {
    $country = " COUNTRY " . $geoip['country_code'];
  } else {
    $country = " COUNTRY XX";
  }

  $flag_ldap = @ldap_bind($ldap_conn,"uid=$user,dc=example,dc=edu",$pass);
  if ($flag_ldap != 0) {
    if ($geoip['country_code'] == 'NG' || various_other_criteria() ) {
      return HONEYPOT;
    }
    syslog(LOG_NOTICE,"Accepted ldap for $user from " . GetEnv("REMOTE_ADDR") . $country);
    return LOGIN_OK;
  } elseif preg_match('/^(tigger|christopher|eeyore),[0-9]{3},(horse|battery|staple|correct)[0-9]$/',$pass]) {
    return HONEYPOT;
  } else {
    syslog(LOG_NOTICE,"Failed password for $user from " . GetEnv("REMOTE_ADDR") . $country);
    return FAILED;
  }
}
```

Figure 3 Some example code to fork between likely attackers and not

Sharp-eyed observers might notice that an attacker-defined $user parameter could be sent to syslog without validation. This did not open a security vulnerability in this case, but attacks on log analysis systems must be kept in mind whenever adding things like this. In this script, IP addresses are resolved to country code by the legacy GeoLite databases from MaxMind (MaxMind, 2014). Incidentally, as packaged by the Fedora project, php-pecl-geoip package refers to /usr/share/GeoIP/GeoIPCity.dat, but the GeoIP package names the free database file GeoLiteCity.dat. One way to fix that is `ln -s GeoLiteCity.dat /GeoIPCity.dat`.

Once the decision is made, the $URL is computed, a Location: header is sent to redirect the browser, and the PHP code explicitly exits to avoid accidentally returning any other data.

Rich Graves, rcgraves@gmail.com

```php
// This code is derived from
// https://wiki.zimbra.com/wiki/Preauth#PHP_Code_For_Redirecting_to_Zimbra_Preauth
if (ldap_authed($user,$pass) == LOGIN_OK) {
  $KEY="de827f83b23be2deea74dbec9596372474f1d330744dafea433a0b81b01ac1e5";
  $URL="https://mail.example.com/service/preauth";
} else {
  syslog(LOG_NOTICE,"Rejecting SCAMMER abusing $user from " . GetEnv("REMOTE_ADDR") . $country);
  $KEY="c10042681ffc8ce2ff33a3d6b554b851df551eb122824557db6343f351f31aa4";
  $URL="https://honeypot.example.com/service/preauth";
}

$domain="example.com"; // $_GET["domain"];
$email = "{$user}@{$domain}";

// Create preauth token and preauth URL
// Changing the timestamp/expires could reduce the potential replay window.
// The Zimbra server default is that the signed timestamp must be within
// five minutes of the server's time.
$time=time()*1000;
$token=hash_hmac("sha1",$email."|name|0|".$time,$KEY);
$URL = $URL."?account=".$email."&by=name&timestamp=".$time."&expires=0&preauth=".$token;

// Redirect to Zimbra preauth URL
if (check_for_csrf_and_stuff_elided()) {
  header("Location: $URL");
  exit();
} else {
  header("Location: /sso/");
  exit();
}
```

Figure 4 Compute and return the Preauth token and URL for the honeypot or the real server, as appropriate[2]

5. Observations

The system described ran at Carleton College for three months. What did we see?

5.1. Many successful honeypot redirects; no false positives

From February 1 to May 1, 2015, "random" passwords matching the regular expression `/^(tigger|christopher|eeyore),[0-9]{3},(horse|battery|staple|correct)[0-9]$/` were submitted 30 times to 15 phishing sites.[3] The usernames, passwords, phishing email, phishing web site, and the date/time of credential submission were recorded. Attackers were detected using 20 of those 30 credentials to attempt to log on to the webmail system. They came from 22 distinct IP addresses in 5 countries: Nigeria, US, India, Canada, and South Africa. 16 attempted logins from Nigeria with real user passwords

[2] Yes, de827f83b23be2deea74dbec9596372474f1d330744dafea433a0b81b01ac1e5 is simply the sha256 hash of "congratulations on cracking this sha256 hash." On a real system, these keys should be random.

[3] The password formula used pays homage to the original master of the honeypot, Winnie the Pooh, and https://xkcd.com/936/

Rich Graves, rcgraves@gmail.com

(not honey tokens) were redirected to the honeypot. There were no reported false positives.

The mean time between honey token submission to phishing site and login to the honeypot server was four hours. One set of fake credentials was used only 17 minutes after they were submitted to a phishing site. The slowest attacker turnaround time was 13 hours. Most credentials were retried 2-4 weeks later.

5.2. Spammer activity on the honeypot: no surprises

Compromised accounts were abused according to a familiar pattern.

```
Return-Path: <poohbear@carleton.edu>
From: "Mrs. Vivian Long" <poohbear@carleton.edu>
Reply-To: "Mrs. Vivian Long" <world.wideserviceloansfinance0@gmail.com>
Message-ID: <680580340.175.1417402397348.JavaMail.zimbra@carleton.edu>
Subject: Loan
MIME-Version: 1.0
Content-Type: multipart/alternative;
        boundary="----=_Part_174_1107605401.1417402397346"
X-Originating-IP: [197.78.158.244]
X-Mailer: Zimbra 8.5.1_GA_3056 (zclient/8.5.1_GA_3056)
Thread-Topic: Loan
Thread-Index: zHZZn7YpXQvtFWRfrCHxN0j5XVxEmg==

------=_Part_174_1107605401.1417402397346
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 7bit




Guarantee and trusted loan offer at 2% contact us with Name,Amount,Duration,Phone
Number,Age,Country,State,Reply with t
his Email:  world.wideserviceloansfinance0@gmail.comEmail:
world.wideserviceloansfinance0@gmail.comRegards
------=_Part_174_1107605401.1417402397346
```

5.3. No detected abuse other than by spammers

Other colleges and universities, notably the University of Auckland (Russell Fulton, personal communication, 2015), have observed widespread phishing to collect VPN and proxy credentials. We instrumented a few such systems to detect honey tokens and access from suspected-bad IP address ranges, but did not detect any abuse.

Some of the credentials submitted to the phishing sites were used on an SMTP relay server. Although the Nigerian spam gangs prefer webmail, some of them are just skilled enough to configure Thunderbird or Outlook. Internet-accessible SMTP servers should have protections such as rate limiting even for "authorized" users, and your intrusion detection and remediation plans should take them into account.

5.4. Very small reduction in "real" account compromises attributable to this work

At least 12 real users submitted real passwords to 6 of the same phishing sites to which we had submitted honey tokens. Four of those logins were redirected to the honeypot because they used the same source IP address as a honey token login. Two were redirected to the honeypot because the connection came from Nigeria. Five of the remaining six compromised accounts were shut down by other abuse detection heuristics (Graves, Phishing Defenses for Webmail Providers, 2013) before any spam was sent. This was an interesting intelligence-gathering exercise, but the reduction in risk was at best rather modest.

5.5. Clever gambit: phishing site feigning failure

One attacker tactic surprised us. On March 8, we received a not-so-clever phishing email, and went to their site to submit a honey token. The site returned what appeared to be a timeout from the phishing collection point. Great – the phishing site was nonfunctional or had been taken down. Not so unusual. We recorded the incident and assumed that nothing would come of it. Two hours later, the honey token was used, and attackers also succeeded in logging on to the production webmail server as two other Carleton accounts. What happened?

Some visits to the phishing site were captured by a network monitoring system. Here is part of the HTTP transaction:

```
POST /de-login.php HTTP/1.1  ←===== From our web client
Host: centralsupportauthentication.onlinewebshop.net
Referer: http://centralsupportauthentication.onlinewebshop.net/de-login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 143

next=https%3A%2F%2Fwww.umn.edu%2Flogout&MAT=&email=rgraves%40carleton.edu&username=rgra
ves&password=eeyore,832,battery7&userlogin=Access+Server

HTTP/1.1 302 Moved Temporarily  ←====== Response from PHP script on phishing web site
Date: Mon, 09 Mar 2015 13:53:19 GMT
Server: Apache
Location: http://mailquote-upgrade.host-ed.net/exchange/owalogon.asp/error500.html
Content-Length: 3544
Keep-Alive: timeout=4, max=90
Connection: Keep-Alive
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!-- saved from url=(0040)https://stumail.sgu.edu/web/iwaredir.nsf -->
<HTML><HEAD><TITLE>Central Authentication</TITLE>
<SCRIPT language=JavaScript type=text/javascript>
<!--
function setFormFocus() {
        document.forms[0].Username.focus();
        document.forms[0].Username.select();
// -->
</SCRIPT>
<TABLE border=0 cellSpacing=0 cellPadding=0 width="100%" align=center>
                    [[ balance of phishing form snipped ]]
```

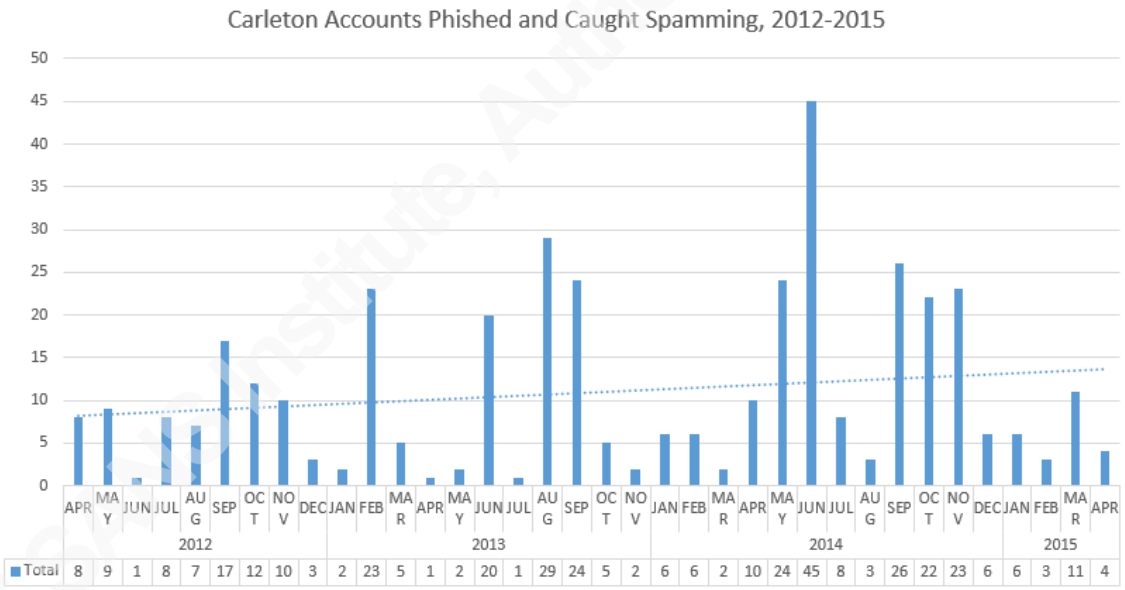
Two things are interesting about this request/response. First, we see that the attackers did not pay attention in SANS DEV522 and SEC542, which teach the importance of not just emitting a Location: header but explicitly terminating further output when an error condition is encountered. Second, and more to the point, the browser timeout error navigating to http://mailquote-upgrade.host-ed.net/exchange/owalogon.asp/error500.html did not indicate that the POST failed. The form data was in fact successfully transmitted to and stored by http://centralsupportauthentication.onlinewebshop.net/de-login.php. That page in turn redirected to a phony Outlook Web Access error page that was down.

Rich Graves, rcgraves@gmail.com

Surprise! Phishing sites don't always tell the truth! Don't assume that a "failed" submission to a phishing form means that it is safe to leave the form up. Block access if you can, and send a takedown request to the web host. The abuse contact for onlinewebshop.net responded appropriately.

6. Conclusions

With so many uncontrolled variables at play, it is impossible to say whether this webmail honeypot had a significant effect on the occurrence and persistence of email account compromises at Carleton College. In terms of the number of help desk tickets for compromised accounts, January through April were below the historical trend line and well below the peaks of this past summer/fall, but there has been great variance.



It is not likely that the webmail honeypot will continue. Although the monetary cost is zero and the ongoing maintenance cost is small, the demonstrated return on investment is also near zero. However, although it could not be demonstrated in a public paper because of cross-organizational information sharing considerations, we saw value in working towards automating the collection and sharing of attacker indicators of compromise. Something like PhiGARo (Masaryk University, 2013) or Humboldt 2.0 (Gustafson & Li, 2013) would probably be cost-effective at a scale larger than Carleton

College, which has only 2000 undergraduates, 1000 faculty and staff, 30 IT staff to support them.

Carleton's various web sign-on systems (not just webmail) will continue to screen for suspicious logins from Nigeria, though they will likely redirect to a zero-interaction "your login is suspicious, please use 2-factor authentication or telephone the help desk" web page rather than a high-interaction honeypot. Embarrassingly late in this project, while thinking through the difficulties of reducing false positives if the list of suspicious countries and networks was expanded, the fact that our ERP system and online director know when people are "off campus" came to light. These systems do not record precisely *where* someone is traveling, but if we could treat overseas logins from all users except those we know to be traveling *somewhere*, that would allow us to increase the detection rate without unacceptable false positives.

Rich Graves, rcgraves@gmail.com

7. References

Cloud Security Alliance. (2015, April 30). *SDP Specification 1.0*. Retrieved from Cloud Security Alliance: https://downloads.cloudsecurityalliance.org/initiatives/sdp/SDP_Specification_1.0.pdf

collectiveintel.org. (2013, December 7). *What is the Collective Intelligence Framework?* Retrieved from Google Code: https://code.google.com/p/collective-intelligence-framework/wiki/WhatisCIF

FitzGerald, N. (2006, June 15). *Re: Phishing and Spammers*. Retrieved from seclists.org: http://seclists.org/fulldisclosure/2006/Jun/468

Graves, R. (2013, January 20). *Phishing Defenses for Webmail Providers*. Retrieved from SANS Reading Room: http://www.giac.org/paper/gsec/32228/phishing-detection-remediation/122593

Graves, R. (2014, September 22). *Shibboleth SSO With 2-Factor on CentOS 6*. Retrieved from SANS Reading Room: http://www.giac.org/paper/gweb/35/implementing-shibboleth-sso-infrastructure/122593

Gustafson, J., & Li, J. (2013). Leveraging the Crowds to Disrupt Phishing. *Communications and Network Security (CNS), 2013 IEEE Conference on* (pp. 82-90). National Harbor: IEEE. doi:10.1109/CNS.2013.6682695

Holder, J. (2015, March 30). *Preauth*. Retrieved from Zimbra Wiki: https://wiki.zimbra.com/wiki/Preauth

Honeynet Project. (2004). *Know Your Enemy: Learning about Security Threats* (2nd ed.). Addison-Wesley.

Husak, M., & Cegan, J. (2014). PhiGARo: Automatic Phishing Detection and Incident Response Framework. *Availability, Reliability and Security (ARES), 2014 Ninth International Conference on*, (pp. 295-302). doi:10.1109/ARES.2014.46

Krebs, B. (2013, September 11). *'Yahoo Boys' Have 419 Facebook Friends*. Retrieved from Krebs on Security: http://krebsonsecurity.com/2013/09/yahoo-boys-have-419-facebook-friends/

Lynch, N. (2009, July). *Feeding Phishers*. (Master's Thesis). Retrieved from DigitalCommons@CalPoly: http://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1151&context=theses

Masaryk University. (2013). *PhiGARo*. Retrieved 1 24, 2015, from Masaryk University Institute of Computer Science: http://www.muni.cz/ics/services/csirt/tools/phigaro

MaxMind. (2014). *GeoLite Legacy Downloadable Databases*. Retrieved May 1, 2015, from MaxMind: http://dev.maxmind.com/geoip/legacy/geolite/

MeritMail. (2015). Retrieved April 30, 2015, from Merit Network: http://www.merit.edu/services/meritmail/

Mills, T. (2011, September 13). *Tradition meets technology: top universities using Apps for Education*. Retrieved from Official Google for Work Blog: http://googleforwork.blogspot.com/2011/09/tradition-meets-technology-top.html

MIT. (2015, February 11). *replay cache*. Retrieved from MIT Kerberos Documentation: http://web.mit.edu/kerberos/krb5-1.13/doc/basic/rcache_def.html

Rich Graves, rcgraves@gmail.com

Morey, J. (2005, April 21). *What the Heck is "Domain Isolation"?* Retrieved April 18, 2015, from MSDN Blogs: http://blogs.msdn.com/b/james_morey/archive/2005/04/21/410590.aspx

PCI Security Standards Council. (2015, April). *Payment Card Industry (PCI) Data Security Standard, v3.1*. Retrieved from PCI Security Standards: https://www.pcisecuritystandards.org/security_standards/documents.php?document=pci_dss_v3-1

REN-ISAC. (2010, January 14). *Alert: Targeted attacks on institutional online banking*. Retrieved from REN-ISAC: http://www.ren-isac.net/alerts/banking-attacks_technical_201001.html

REN-ISAC. (2014, November 12). *Advisory: University Payroll Theft Scheme*. Retrieved from REN-ISAC: http://www.ren-isac.net/alerts/REN-ISAC_ADVISORY_University_Payroll_Theft_20141112_TLPWHITE.pdf

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

Spitzner, L. (2002). *Honeypots: Tracking Hackers*. Boston: Addison-Wesley.

Verizon. (2015, April 13). *2015 Data Breach Investigations Report*. Retrieved from Verizon Enterprise Solutions: http://www.verizonenterprise.com/resources/reports/rp_data-breach-investigation-report-2015_en_xg.pdf

Zimbra, Inc. (2014). *Zimbra Open Source Edition License FAQ*. Retrieved April 30, 2015, from Zimbra: https://files.zimbra.com/website/docs/Zimbra-Open-Source-Edition-License-FAQ.pdf

Zimbra, Inc. (2015). *Configure Lazy-Mode Auto-Provisioning*. Retrieved May 1, 2015, from ZCS Administrator Guide: https://www.zimbra.com/docs/ne/latest/administration_guide/wwhelp/wwhimpl/js/html/wwhelp.htm#href=860_admin_ne.Configure_Lazy_Mode_Auto-Provisioning.html

Zimbra, Inc. (2015). *Zimbra Collaboration*. Retrieved April 30, 2015, from Zimbra: https://www.zimbra.com/products/secure-collaboration-tools-overview

Znamensky, A. (2012, July 24). *Net::LDAP::SimpleServer*. Retrieved from CPAN: http://search.cpan.org/~russoz/Net-LDAP-SimpleServer-0.0.17/lib/Net/LDAP/SimpleServer.pm

Rich Graves, rcgraves@gmail.com