Sources of Error When Identifying Misuse of Corporate Internet Services - A Cautionary Note

Lawrence Stewart
Centre for Advanced Internet Architectures. Technical Report 040224A
Swinburne University of Technology
Melbourne, Australia
lastewart@swin.edu.au

***Abstract*-This technical report aims to investigate the ease with which unsolicited content from the Internet can be fetched by an email client triggered by specially formatted HTML email. It also looks at the default behaviour of a number of different popular email clients across the Windows and FreeBSD platforms, and their configuration options. We found that it was surprisingly easy to get the tested email clients to download images from the Internet with no prompting. Such activity could leave traces in corporate ITS server logs or on a user's local machine which could be misconstrued as breaking company IT policy and result in an employee's dismissal. The results of this investigation are meant to educate IT personnel about the need to exercise caution before jumping to conclusions about the activities of an individual. They are also meant to educate email users about the risks involved when using email and how to avoid walking into traps by being aware of the configuration options of the email client they use and understanding what they do.**

Keywords- Lawful interception, framing, inappropriate content, email, caching

I. Introduction

The widespread use of electronic mail (email) as a fast, reliable means of communication has worked its way into every facet of modern day life and work. For these reasons, it has been widely adopted in the work place. Combined with the ever increasing use of the Internet in day to day work, corporate information technology departments are having to keep up to date and manage their resources effectively and efficiently.

Part of these responsibilities is to ensure that resources are being used for work related purposes and not being abused by employees. There have been a number of instances in recent years of employees losing their jobs for inappropriate use of their corporate information technology services.

IT departments have gradually been addressing these issues by taking a “big brother” stance and monitoring the traffic flowing over the network, using proxy servers with ban lists to make it difficult for employees to visit websites considered inappropriate, and so forth. Many corporations have gone to great lengths to ensure accountability in their networks. The net result of this has resulted in everything flowing through the network being traced and logged in one form or another. With simple, freely available tools, network administrators are able to keep logs of traffic patterns for employees, and in the event suspect behaviour is taking place, use this evidence to identify employees and possibly take disciplinary action.

A couple of recent incidents involving the dismissal of employees over the sending of emails containing pornography [1] has further stirred the debate.

The important question one must ask in all this is does an entry in a server log or a picture stored in an employee's local cache signify intent to contravene company policy?

With a couple of quick experiments, we were able to prove that the answer to this question is in fact “no”, and that it is very easy to make an honest mistake, or even possibly maliciously try to frame someone using the current email technology as it is.

II. Methodology

For this report, we decided to focus on how to trigger the downloading of images to a client machine. There are other forms of content that could have been investigated, but all of them end up with the same net result of the client's machine having to make requests to a foreign host which is the all important factor.

There are two main ways an image can be incorporated into an email. It can be directly attached to the email, or it can be linked to the email from a remote location and loaded on opening the email. The second option is of interest for this report, as it results in the client's computer following the embedded link and downloading the content the moment the email is opened. This can be accomplished by writing the email in hypertext markup language (HTML), which is now a standard email format recognised by most popular email clients.

A. Email format

There were two simple ways identified to load images on a user's machine without their consent. First was to embed a HTML <img> tag inside the body of the email. The second was to use some javascript in the body of the email to do the fetching of the pictures. This method would be ideally suited to attempting to frame someone, as you can fetch images without even displaying them, so unless the client is savvy enough to inspect th source of the email, they would not even know content is being downloaded.

In order to make the above two methods difficult for the average user, most email clients (and all that were tested for this report) do not allow you to compose emails in raw HTML. They take your text and generate the HTML tags behind the scenes. In order to generate our own raw HTML emails, a small python [2] script was written to talk directly to a simple mail transport protocol (SMTP) server. We were then able to transfer any information we liked in an email without the limitations imposed by an email client.

We sent two different emails in order to test the two different methods outlined above. The first was a standard HTML formatted email with a single embedded <img> tag. The email contents that were sent are listed below.

```
   <html>
   <head>
   <meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
   </head>
   <body>
   Hi there!
   <p align="center"><img
src="http://www.stanford.edu/~robevans/DSCF005
6.JPG" name="robjump" alt="robtest"
border="1"></p><br>
   </body>
   </html>
```

It should be noted that this method renders the embedded image at its original size. It is also possible to add height and width attributes to HTML <img> tags to render an image at a user specified pixel size. This means that you can include normal sized images in a HTML email and make them one pixel large (rendering them essentially invisible).

The second was a standard HTML formatted email with some javascript that preloaded five images but did not display them. The email contents that were sent are listed below.

```
   <html>
   <head>
   <meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
   </head>
   <body>
   <script language="JavaScript"
type="text/JavaScript">

       images = new Array();
       imgSrcs = new Array();
       imgSrcs[0] =
"http://www.stanford.edu/~robevans/robbiebigju
mp1SM.jpg";
       imgSrcs[1] =
"http://www.stanford.edu/~robevans/DSCF0012.JP
G";
       imgSrcs[2] =
"http://www.stanford.edu/~robevans/DSCF0018.JP
G";
       imgSrcs[3] =
"http://www.stanford.edu/~robevans/DSCF0056.JP
G";
       imgSrcs[4] =
"http://www.stanford.edu/~robevans/DSCF0057.JP
G";

       for(i = 0; i < imgSrcs.length; i++)
       {
              images[i] = new Image();
              images[i].src = imgSrcs[i];
       }

   </script>
   </body>
   </html>
```

B. *Testing platform*

We decided to test on both Microsoft Windows 2000 [3] and a Unix based operating system, FreeBSD [4]. Unix derivatives are well known for putting great care into security related matters, where as Windows has developed a reputation for being somewhat careless when it comes to such matters, which makes for interesting comparison. The K Desktop Environment (KDE) [5] was the window manager run on the FreeBSD client machine.

The email clients tested under Windows 2000 were: Outlook Express [6], Netscape 7.1 [7], Mozilla 1.6b [8] and Novell's Groupwise 6.5 [9]. The email clients tested under FreeBSD were: Kmail 1.5 [10] and Mozilla 1.2b [8].

Two web based email interfaces, Yahoo mail [11] and Hotmail [12], were also tested in the roundup.

C. *Gathering the information*

In order to determine if the email clients were following the links and downloading the content specified, we had to run some packet capturing software on the client machine as the email was being opened. Under FreeBSD, we ran tcpdump [13] with the -s0 and -w *filename* flags to capture the entire packet contents and save them to file *filename*.

Once the raw data had been gathered, we used Ethereal [14] to inspect the contents of the packets and scan for hyper text transfer protocol (HTTP) get requests to the universal resource locator(s) (URL) we had embedded in the email.

Under Windows 2000, we used Ethereal and WinPCap [15] to do all the packet capturing and followed the same inspection method as above.

III. Analysis

A. *Outlook Express 6 email client under Windows 2000*

Outlook Express (OE) is the default email client installed with the Windows 2000 operating system and is one of the most widely used email clients [16]. OE is much less configurable than most of the other tested email clients. It only allows the user to select whether they read email in plain text or HTML, the latter of which is the default. It is immediately apparent that this poses a security exploit in itself, as uninformed users will have HTML email being interpreted and with no option to disable external content downloading, will be open to compromise.

There is a security settings tab that has a toggle option for less or more secure. It is extremely vague, but switching to "Internet zone" (less secure option) allows emails with javascript to run. With the more secure option, javascript is disabled but images and external content can still be reached.

We were able to get OE to make a HTTP get request for the regular HTML email without changing any settings. By changing the security option to "Internet zone", we were able to get javascript to load external images as well.

The caching to disk of downloaded content for OE is dependent on the caching options specified in Internet Explorer in the Tools->Internet Options->Temporary Internet Files Settings configuration panel. The default setting is to allow caching to the folder `C:\Documents and Settings\<`*`username`*`>\Local Settings\Temporary Internet Files` (where *username* is your Windows login name). Note that images preloaded using javascript are also cached to the above location.

OE does aim to be a bare bones simple to use email client, and with this in mind it does serve its purpose. However, the lack of specific security settings to govern HTML email makes it extremely unclear exactly what the client is or is not doing for the user, which makes it dangerous from a security point of view.

B. *Netscape 7.1 and Mozilla 1.6b email clients under Windows 2000*

These two clients are grouped together as they are both built on the Mozilla open source engine, which results in them having the same configuration options, defaults and visual looks.

Mozilla is an open source web software suite that contains an email client. It has more configuration options than you can poke a piece of CAT 5 at, which can be confusing for the uninitiated. That said, its configurability is excellent for those that have a modest idea about what they are doing.

Both clients default to allowing the loading of external references and the reading of HTML email. Javascript functionality is disabled by default, but can be turned on. You can also control what a script can and cannot do if you do enable javascript for email.

The caching to disk of downloaded content for both clients is dependent on the caching options specified in the Edit->Preferences->Advanced->Cache configuration panel. The default setting is to allow caching to the folder `C:\Documents and Settings\<`*`username`*`>\Application Data\Mozilla\Profiles\default\<`*`profilename`*`>.slt\cache` (where *username* is your Windows login name and *profilename* is a randomly generated set of characters created by the client program). Note that images preloaded using javascript are also cached to the above location.

C. *Novell Groupwise 6.5 email client under Windows 2000*

Novell Groupwise 6.5 is an enterprise email system that runs on Windows 2000. Being a third party application, it runs with its own proprietary server and this adds degrees of complication to the testing process.

The Groupwise client program has fairly minimal security options. It does allow the reading of HTML emails by default.

For unknown reasons, our HTML email with the embedded <img> tag could never be received by the Groupwise recipient. One possibility is that there is some filtering occurring at the server which deletes or quarantines emails that could have inappropriate content such as pictures. As we were unable to find the cause of this anomaly, we cannot conclude if it is the Groupwise software being extra secure or some other alternative.

We were able to send the javascript email, but it had no effect when opened, so although there is no explicit mention of disabling javascript, it is clearly not allowed to function.

D. *Netscape Communicator 4.78 email client under Windows 2000*

Although slightly outdated, the Netscape Communicator suite is still used quite widely. As we had it already installed on our Windows 2000 testing machine, we decided to run its email client through our tests.

The first thing to note is the lack of configuration options as compared to Netscape 7.1. There are no options to disable the rendering of HTML email or loading of external references. There are no options to configure javascript functionality.

We found that by default, both foreign images and javascript are able to function, and the downloaded content is cached to `C:\Program Files\Netscape\Users\default\Cache` (assuming you installed the program to C:\Program Files). The caching to disk of downloaded content is dependent on the caching options specified in the Edit->Preferences->Advanced->Cache configuration panel.

The fact that javascript support in email is enabled by default and there are almost no useful security configuration options makes Netscape Communicator an extremely poor choice for use as an email client.

E. Kmail 1.5 email client under FreeBSD

Kmail is the default email client installed with KDE. For comparison, it is quite similar in look and feel to Outlook Express in Windows. Kmail defaults to reading all email in plain text. If a HTML email is received, Kmail displays the message in plain text and offers the option to then display the email in HTML if the source is "trusted" by the client. This is an extremely good feature, as it allows HTML source code to be screened by the client and displayed properly if deemed to be safe. Kmail has configuration options to allow reading email in HTML and to prevent HTML email, when viewed, from downloading external content. The default is to not allow emails to download external content. Kmail cannot be influenced by javascript and other plugins and it has no support for these.

Kmail also gives suitable warnings next to the options controlling HTML email viewing which alerts the user to the possibility of having their machine compromised.

This combination of defaults makes Kmail extremely secure right out of the box and we could only run our tests by specifically changing the defaults. We were able to get Kmail to make a HTTP get request by turning on the option that allowed external references to be loaded. The caching to disk of downloaded content is dependent on the caching options specified in Konqueror in the Settings->Configure Konqueror->Cache configuration panel. The downloaded content was cached to `/home/<username>/.kde/share/cache/http/` (where *username* is your FreeBSD login name).

F. Mozilla 1.2b email client under FreeBSD

Although I slightly older version, there is almost no noticeable difference between the Mozilla 1.2b for FreeBSD and Mozilla 1.6b for Windows.

Mozilla defaults to allowing HTML messages to be read and allows the loading of external images, which meant the email with <img> tags was able to load the image immediately. There was a configuration option to allow javascript in email as well, and with this turned on, we were able to preload some images using the Mozilla client.

The caching to disk of downloaded content is dependent on the caching options specified in the Edit->Preferences->Advanced->Cache configuration panel. The default setting is to allow caching to the folder `/home/<username>/.mozilla/lstewart/<profilename>.slt/Cache/` (where username is your FreeBSD login name and profilename is a randomly generated set of characters created by the client program). Note that images preloaded using javascript are also cached to the above location.

Although Mozilla allows image downloads by default, which makes it less secure than Kmail, it does have more configuration options to govern every possible aspect of the client and this makes it a fantastic email client for experienced users.

G. Yahoo web mail

Web based email services are extremely popular, especially for personal use. There are countless such services, and we decided to test two of them to see how they deal with HTML formatted email, and what security options they provide and default to.

Yahoo provides an option to not load images from external sources, but it is not enabled by default. It has no option to disable reading HTML email.

We were able to trigger our browser to download an external image without intervention.

However, it is important to note that web based email clients are greatly influenced by the settings of the browser being used to access the client. For instance, the Mozilla web browser (which we used for testing the web based email clients) has a configuration option to disallow the loading of images that are not from the originating server, which would stop this behaviour from occurring and being a problem. The caching of web content is also controlled by browser settings, which by default will store downloaded images in the browser's local cache.

With the javascript email, the Yahoo client simply converted it into plain HTML text which showed up when the email was viewed. This is a good feature as it allows the user to see what the email was intended to do.

H. Hotmail web mail

Different company but very similar look and feel to Yahoo and same options available to users. Hotmail does not provide the choice of HTML or plain text email reading and does supply an option to allow and image to be screened by the client before being displayed. However, it is turned off by default.

The way Hotmail handled our javascript email was interesting. It parsed the entire email and added <comment> tags around the javascript code, thus disabling it. This is similar to the Yahoo client's approach to handling javascript, except that to see the added tags we had to view the web page source and find the email source in order to see what happened to the javascript.

I. Summary of results

Table 1 summarises the results for each email client tested with default settings. Every email client that defaulted to allowing an external image to be loaded, cached the image locally and thus left evidence in both the proxy server log as attempting to access a particular site, and on the local machine. This cements that fact that users cannot afford to be naïve about the specifics of which email client they are using and what configuration options are offered.

	Platform	***Leaves traces in proxy log with default settings***	***Leaves traces on local machine with default settings***	***Can be configured to NOT leave traces in proxy log***	***Can be configured to NOT leave traces on local machine***
Outlook Express	Windows	Yes	Yes	No	No
Netscape 7.1	Windows	Yes	Yes	Yes	Yes
Mozilla 1.6b	Windows	Yes	Yes	Yes	Yes
Groupwise 6.5	Windows	N/A	N/A	N/A	N/A
Netscape Communicator 4.78	Windows	Yes	Yes	No	Yes
Kmail 1.5	FreeBSD	No	No	Yes	Yes
Mozilla 1.2b	FreeBSD	Yes	Yes	Yes	Yes
Yahoo	N/A	Yes	Yes	Yes	Yes
Hotmail	N/A	Yes	Yes	Yes	Yes

Table 1. Summary of results for tested email clients

IV. Research Limitations and Further Work

Whilst Outlook Express was tested, its parent program, Microsoft Outlook, was not. According to surveys [16], Outlook is the most widely used email client for work related purposes, which would make it worthwhile to run some tests on.

It would also have been worthwhile tracing why the Groupwise server was not forwarding our HTML email with <img> tags.

V. Conclusion

By sending specially formatted HTML emails to a number of different popular email clients, we were able to show how easy it is to instigate the loading of external unsolicited content from the Internet. We discovered the strengths and weaknesses inherent in each of the tested clients.

Clients developed for cross platform and FreeBSD use were by default more secure and more configurable. Outlook Express and Netscape Communicator were found to be lacking in security options and Groupwise was somewhere in the middle, although unknown configuration settings out of our control disabled the transmission of HTML email containing embedded <img> tags. The web based clients were dependent on the web browser settings for security.

It was found to be extremely easy to send HTML email with embedded <img> tags that downloaded content from external websites, as every client except one allowed this by default. Javascript was generally not supported by default except for Netscape Communicator.

As mentioned previously, the ability to send “invisible” images by making them render at one pixel, combined with the fact that almost every email client supports loading external images by default, means that any email user using these clients is vulnerable straight out of the box.

These results should demonstrate the worth of educating email users about the risks associated with using email and how to avoid walking into traps. The choice of a good email client and basic understanding of its configuration options would go a long way to protecting users.

We would recommend the use of the most recent version of Mozilla or Netscape (1.6b and 7.1 respectively at the time of writing) as a highly configurable email client for Windows.

The education of IT personnel about the need for careful consideration of the facts before pointing the finger should also be paramount in ensuring both users and administrators can do their job properly.

There is a clear case for implementing a policy that content found in the cache directories listed throughout this paper on a user's local machine should be ignored and not used as evidence against them.

References

[1] Mehlman, J., "Porn hysteria strikes again", Zdnet Australia News, 4 December 2003, http://www.zdnet.com.au/news/communications/0,2000061791,20281622,00.htm (as of February 2004)

[2] Python Programming Language, http://www.python.org/ (as of February 2004)

[3] Microsoft Windows 2000, http://www.microsoft.com/windows2000/ (as of February 2004)

[4] The FreeBSD Project, http://www.freebsd.org/ (as of February 2004)

[5] K Desktop Environment, http://www.kde.org/ (as of February 2004)

[6] Microsoft Outlook Express, http://www.microsoft.com/windows/oe/ (as of February 2004)

[7] Netscape, http://www.netscape.com/ (as of February 2004)

[8] Mozilla, http://mozilla.org/ (as of February 2004)

[9] Novell Groupwise, http://www.novell.com/products/groupwise/ (as of February 2004)

[10] Kmail – The KDE mail client, http://kmail.kde.org/ (as of February 2004)

[11] Yahoo! Mail, http://mail.yahoo.com/ (as of February 2004)

[12] MSN Hotmail, http://www.hotmail.com/ (as of February 2004)

[13] Tcpdump, http://www.tcpdump.org/ (as of February 2004)

[14] Ethereal, http://www.ethereal.com/ (as of February 2004)

[15] WinPCap, http://winpcap.polito.it/ (as of February 2004)

[16] Grossman, E., "Real-World Email Client Usage: The Hard Data", http://www.clickz.com/em_mkt/infra/article.php/1428551, 19 July 2002 (as of February 2004)