Malicious Yahooligans

Eric Chien
Symantec Security Response, Ireland

Originally published by Virus Bulletin, August 2006. Copyright held by Virus Bulletin, Ltd., but is made available courtesy of Virus Bulletin. For more information on Virus Bulletin, please visit http://virusbtn.com/
Malicious Yahooligans

Contents
Target Eliminated.......................................................................................................................4
Disinfecting with AJAX................................................................................................................5
Reaping the Harvest.......................................................................................................................7
Rise and Downfall........................................................................................................................7
Ground Zero................................................................................................................................8
Conclusion................................................................................................................................10
On June 11, 2006 we received reports of a worm propagating via email. That in itself was nothing special, but what was more interesting was the fact that this worm appeared to propagate only through Yahoo! Mail email accounts.

Once we obtained a sample, it became clear that this threat wasn’t any ordinary email worm, but was the first webmail worm, later named JS.Yamanner@m.

In the past, we have seen email worms that sent themselves via Outlook and those that communicated with SMTP servers directly, but we hadn’t seen an email worm that actually harnessed a webmail interface. JS.Yamanner@m utilized Yahoo! Mail’s webmail interface both to collect email addresses and to send itself to other users.

**Target Eliminated**

Furthermore, JS.Yamanner@m did not require a user to execute a file. Instead, the worm took advantage of a vulnerability in Yahoo! Mail so that merely by opening an infected mail message for reading, the user would cause the worm to execute and begin sending itself to addresses with which the infected user had corresponded in the past. This functionality would result in both the rise and the downfall of JS.Yamanner@m.

The vulnerability utilized by JS.Yamanner@m was an undisclosed zero-day vulnerability in Yahoo!’s JavaScript- and HTML-filtering algorithms.

Yahoo!, like other web applications, must parse emails for HTML and JavaScript and neuter tag attributes that allow JavaScript execution. If Yahoo! didn’t neuter HTML and JavaScript contained in emails, then emails could contain malicious JavaScript that would be executed under the context of the Yahoo! domain. This could allow one to read other people’s emails, impersonate other users, and create a self-replicating worm.

An example of a tag attribute that Yahoo! neuters is onload. The onload attribute instructs the browser to execute JavaScript code (or another scripting language) as soon as the page is rendered in the browser.

Another example is the target attribute. The target attribute instructs the browser to load the content in a particular page or frame. This page or frame could reside on another domain. If the tag is not filtered, then private content on a yahoo.com page, such as your email address, could be sent to a page outside of the yahoo.com domain.
Malicious Yahooligans

In this case, the worm utilized the following malformed HTML:

```html
<img src='http://us.i1.yimg.com/us.yimg.com/i/us/nt/ma/ma_mail_1.gif' target=""onload="" // malicious javascript here //">
```

This HTML is nonsensical as the information within the quotes after the target attribute should contain the name of a frame or page, but instead includes nothing and is followed immediately by an onload attribute.

The purpose of this malformed HTML becomes clearer when one understands Yahoo!'s filtering algorithms. In particular, Yahoo! filters out the target attribute to prevent information disclosure. However, this results in the following HTML being rendered by the browser:

```html
<img src='http://us.i1.yimg.com/us.yimg.com/i/us/nt/ma/ma_mail_1.gif' onload="" // malicious javascript here //">
```

Note that the target="" text has been removed and this results in proper HTML. However, due to the fact that Yahoo! has removed the target attribute, the onload attribute now resides in a valid position and Yahoo!'s algorithms don't take a second pass at filtering. As a result, the dangerous onload attribute is not filtered out.

This vulnerability allows the JavaScript within the onload attribute to be executed by the browser under the context of the yahoo.com domain, all without any interaction from the user. The user must merely view the page.

The vulnerability has since been fixed by Yahoo! and now results in the following HTML:

```html
<img src='http://us.i1.yimg.com/us.yimg.com/i/us/nt/ma/ma_mail_1.gif' onfiltered="" // malicious javascript here //">
```

Note that, while the target="" text has still been removed, the onload attribute has been neutered properly by replacing it with onfiltered, which is an invalid tag. The JavaScript is no longer loaded and executed after the page load.

Disinfecting with AJAX

Once the user reads the email, the JavaScript code of the worm begins executing via the unfiltered onload handler. The worm utilizes AJAX (Asynchronous JavaScript and XML), which is another first. Now running under the context of yahoo.com and the currently logged on user session, the worm has the ability to parse the web page and make the same HTTP queries as if the user had clicked on items in the webmail interface.
Malicious Yahooligans

Smartly, the worm uses AJAX for the HTTP queries. If the worm had not used AJAX, any HTTP queries would have resulted in another page loading, which would be more likely to be noticeable to the user, as well as putting the calling JavaScript code out of scope.

By using AJAX, the worm can issue multiple HTTP queries in order to find email addresses and send itself all under the covers without changing the page.

JavaScript is global in scope across an HTML page, so script in one block has full access to variables and functions in other blocks or included JavaScript files. JS.Yamanner@m takes advantage of this in order to perform some of its actions.

For example, the first thing JS.Yamanner@m does is to determine which server is being utilized. Yahoo! serves the web application from many different servers. The URL is stored in a variable called url0, and from this variable JS.Yamanner@m can parse out the domain.

To collect email addresses, JS.Yamanner@m sends an HTTP query as if someone had selected the QuickBuilder functionality in Yahoo! Mail using AJAX so the page does not refresh. QuickBuilder is a Yahoo! Mail feature that searches all your mail in selected folders for any email addresses that are not already part of your address book. The purpose of this feature is to allow you to build your address book quickly.

JS.Yamanner@m takes advantage of this feature to get Yahoo! to find viable email addresses. JS.Yamanner@m requests the first 100 (alphabetically by folders) email addresses in all folders and parses these for any that match @yahoo.com or @yahoogroups.com. In addition, the worm attempts to filter out the user and sender’s email addresses, thus preventing the threat from resending itself back to the user. The worm obtains the user’s and sender’s email addresses from form fields already populated by the Yahoo! Mail application.

With a list of viable email addresses, JS.Yamanner@m sends an AJAX HTTP request to compose a new message, but doesn't actually use this request to send the message. Instead, this request is used merely to generate a new ‘crumb’, which is similar to a session-tracking cookie, but a form value within the page.

JS.Yamanner@m then sends a second AJAX HTTP request with a variety of POST variables set in order to forward the open message (which contains itself) to the list of email addresses discovered via QuickBuilder. One email address is set as the ‘To:’ email address and the entire list is set as the BCC: field. Since this is a forwarded message, the From: address will be set to whomever opened the infected message. The subject line is set to ‘New Graphic Site’ and the message body is set to ‘Note: forwarded message attached’.
Malicious Yahooligans

JS.Yamanner@m also needs to set a variety of administrative values, including a tracking number for the message being forwarded by parsing the HTML page, the crumb value, and set a parameter so the message isn't saved in the Sent folder.

When the message is received, at the bottom of the chain of forwarded messages, the infected HTML attachment will automatically be rendered and contain the text 'Please wait while loading the site'.

Reaping the Harvest
Few users will actually see the content of the message since as soon as the page is loaded, the worm executes and after the worm forwards itself to further targets, JS.Yamanner@m then calls window.navigate to redirect the page to another website (www.av3.net), along with a variety of GET parameters.

The purpose of the redirection wasn't completely clear at first. The redirection included GET parameters which appear to have been used for debugging purposes to show the Sent folder view in Yahoo! Mail after the worm finished sending itself, but before being released, the yahoo.com domain was replaced with www.av3.net. Also, appended to the parameters was the list of email addresses, which would not be needed when displaying the Sent folder view.

While many assumed that the purpose of this last step was to harvest email addresses and send them to av3.net, av3.net has existed for a long time and is referenced and utilized on a variety of websites. Av3.net hosts multimedia content commonly used on MySpace.

In addition, the form of the GET parameters with unnecessary Yahoo! Mail parameters didn’t quite add up. Furthermore, the author could easily have used AJAX instead of redirection. Perhaps the website was simply a red herring or an attempt at implicating someone else.

After some investigation, we were able to determine that av3.net was owned by the author of the worm and thus, email addresses were harvested. The unnecessary GET parameters were likely added so that the harvested email addresses weren't visible in the URL box. The actual harvesting of the email addresses does not appear to be done by any back-end scripting on the site, but perhaps just parsing of the standard web access logs. Another reason AJAX or a dummy site wasn't used is perhaps because the author desired page views for generating income via hosted advertisements.

Rise and Downfall
Because the worm redirected automatically to another site that hosted a hit counter, we were able to track infection rates and also get a glimpse of who was infected.

JS.Yamanner@m infected close to 200,000 users between June 11–12, 2006, before dying out due to
the vulnerability being fixed. The number of hits is likely to be higher than the number of infected users since a common reaction of a user, when they saw their browser redirected to the av3.net website, would be to hit the back button. But doing so would just cause the infection to execute again and then redisplay the av3.net site. According to the web stats, infected users included a variety of governments and large corporations including financial institutions.

While the worm spread quickly due to the use of a vulnerability in Yahoo! Mail, the worm was eradicated immediately as soon as Yahoo! patched the vulnerability. At that moment the worm could no longer spread and the hits on av3.net stopped (see Figure 1).

Web applications such as Yahoo! Mail have a distinct advantage over client applications since as soon as a vulnerability is patched, all users are protected immediately. In contrast, email applications such as Outlook suffer from the fact that not all users upgrade or patch their installations straight away and therefore remain vulnerable to similar attacks.

The country most affected by JS.Yamanner@m was the United States, which is no surprise considering the popularity of Yahoo! Mail within the US market. The second-most affected country was Iran. The reason for this became clear when we tracked the infection back to ‘ground zero’ and managed to obtain some information about the author himself.

**Ground Zero**

The worm simply forwards an existing message. However, forwarding the message means that the original headers can be found on previous messages.

After obtaining a few samples, we were able not only to determine the original infection, but also to create interesting relationship trees demonstrating how the worm spread from one user to the next and how users were interconnected.

The diagram shown in Figure 2 is a partial branch. The blue ellipses represent Yahoo! Mail groups and the squares represent individual Yahoo! email addresses. The red circle on the right is the first infection.
After some investigation of the originating addresses, it seems that the author may have been Iranian, or at least shared interest with and had connections with Persian speakers. Thus, it was no surprise to see that the second most affected country was Iran as the first person to be infected likely had contacts with people living in Iran.

Further confirmation of the worm’s country of origin appeared when the author sent an email to anti-virus companies in search of employment.

“… Finally I should mention that I don’t like to disturb no one. Since I live in iran and taking a Job in good computer companies is very hard (becaue getting Visa is very hard from US) I just want to prove that I have some abilities in web programming. And I like to work with professional team like you if there is any way to do that…”

The final piece of data came when attempting to get the site shutdown and at the same time determine if it was an innocent bystander. At first, no contact information was available on the site, so as part of normal procedure we sent a message to the upstream provider.

Later, however, we found that the page had been modified and that it included contact information. A message sent to the address given on the web page was met with a reply that confirmed that the author of the worm owned the site and it even came with a full name. Interestingly, the data indicates that the author has spent time living in Canada. Of course, this could just be someone attempting to implicate someone else. We won’t know unless authorities actually arrest the author.
Conclusion

JS.Yamanner@m was not only the first webmail worm, but it demonstrated how web applications are equally susceptible to vulnerabilities. More so, JS.Yamanner@m demonstrates the impact of AJAX and how AJAX can be harnessed to develop more efficient web application worms.

While similar worms will also generally require a vulnerability in the web application, web application vulnerabilities are not rare. Fortunately, as soon as the vulnerability is patched, the worm stops spreading.
About Symantec
Symantec is the global leader in information security, providing a broad range of software, appliances, and services designed to help individuals, small and mid-sized businesses, and large enterprises secure and manage their IT infrastructure. Symantec’s Norton™ brand of products is the worldwide leader in consumer security and problem-solving solutions. Headquartered in Cupertino, California, Symantec has operations in 35 countries. More information is available at www.symantec.com.