Master of Puppetnets:
DDOS Attacks Launched On MySpace

Blase Ur
blaseur@rci.rutgers.edu

0. ABSTRACT

In a Puppetnet Attack, the owner of a popular website coerces that site's visitors to request large amounts of data from an arbitrary victim webserver, creating a distributed denial of service attack using web browsers. In this work, we introduce a novel Puppetnet Attack that is launched on the MySpace social network, eliminating the previous attack requirement that attackers be popular webmasters. We describe a proof-of-concept MySpace Puppetnet attack in which specially crafted comments are posted to a minor celebrity's MySpace page. We then quantify this attack, finding that it is often feasible to craft a 10 - 100 megabyte MySpace comment and thus direct hundreds of gigabytes or even terabytes of daily traffic towards victim web servers. We discuss the required invariants of MySpace Puppetnet attacks and evaluate countermeasures from the perspectives of both the victim and MySpace. We find that the current threat of MySpace Puppetnet attacks strongly motivates restricting HTML usage that is currently permitted in MySpace.

1. INTRODUCTION

Puppetnets, a recently proposed idea in which innocent users' web browsers are manipulated into requesting large amounts information, allows for attacks to be carried out in a new way and on a large scale. Distributed denial of service attacks, in which many nodes request a large amount of information from some victim with the intent of disrupting the victim's normal operation, are the most prominent attack possible in a Puppetnet. For instance, a malicious web site could direct all of its visitors to download hundreds of images from some victim web server, bombarding the victim with requests and preventing legitimate traffic from passing through. Unfortunately, the Puppetnets project necessarily assumes that an attacker is the owner of a large and popular webpage, limiting the applicability of a Puppetnet attack. We aim to relax this assumption and describe a Puppetnet attack that can be carried out by a powerless, arbitrary user by leveraging MySpace.com, a popular social networking website.

If an arbitrary user can leverage MySpace to launch a distributed denial of service (DDOS) attack, the barrier to carrying out an attack is significantly lowered. Previous DDOS attacks have been launched by virus writers who control massive botnets, and accumulating a botnet is a very non-trivial task. In contrast, arbitrary users having the ability to launch a DDOS attack would likely make these attacks much more common. In general, an arbitrary user cannot gather an audience to listen to her, let alone impel that audience to carry out an attack for her. The arbitrary user is not very powerful in terms of influence, knowledge, or resources. However, the MySpace era has allowed for social hierarchies to be significantly flattened online. Middle school soccer players and famous rock stars can be “virtual” friends, allowing celebrities to accumulate hundreds of thousands or even millions of friends and fans. Users can customize their MySpace pages using a subset of the HTML markup language, making the online world their own personal space. This empowerment of the arbitrary user is reflective of many sociological shifts, yet also becomes a vulnerability.

Since the arbitrary user can become friends with a celebrity on MySpace, that user is then able to leave comments on that celebrity's

MySpace page. In essence, the arbitrary user now has a forum to reach a mass audience. Furthermore, HTML can often be included in these comments. As a result, an arbitrary user can craft a comment that coerces its viewers to request large amounts of data from some victim web server. If this comment is posted on a handful of celebrities' MySpace pages, a large number of visitors from all over the world will request information from the victim web server, creating a DDOS attack.

This novel approach of using MySpace comments and HTML to craft a DDOS attack can feasibly result in hundreds of gigabytes or even terabytes of traffic per day being requested from an arbitrary victim web server, assuming that the victim server is currently hosting files of non-trivial size. The steps of this attack are discussed and the potential damage quantized in Sections 4 and 5, respectively. In Section 6, we discuss invariants and characteristics of the attack, identifying those elements which are necessary and useful for MySpace Puppetnet attacks to succeed. In Section 7, we examine defense mechanisms that can be implemented by potential victim webservers as well as MySpace, discussing how and why MySpace's policies on HTML usage should be tightened in order to prevent MySpace Puppetnet attacks.

2. BACKGROUND

2.1 Puppetnets

Our work most directly expands on Puppetnets, a project which describes and quantifies a method for coercing web browsers into performing a distributed denial of service attack [F,X]. To create a Puppetnet, the owner of a popular website inserts code into her webpage that directs users to access the resources of a victim site. Visitor's browsers are asked to perform some bandwidth-heavy task, such as downloading images from a victim website. For instance, all visitors to the malicious Foo.com website are asked to download hundreds of images from a victim, Bar.com, as part of

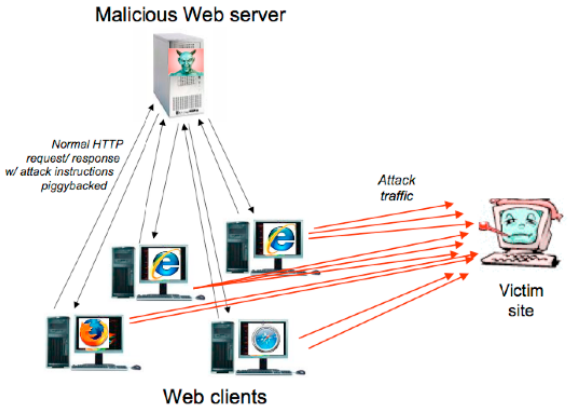

Figure A. In a Puppetnet Attack, a malicious webserver coerces the browsers that visit its site to request large amounts of traffic from some victim. (Image taken from [X])

rendering the Foo.com page. All of the users who thus request objects from Bar.com are part of the "Puppetnet" since these users' browsers are controlled like puppets by the attacker's Foo.com. This scenario is illustrated in Figure A. Although javascript code lends itself easily to short and effective PuppetNet attacks, HTML objects, such as <img> tags that load an image, can be nearly as effective.

One of the unique insights of the Puppetnets papers is that Puppetnet attacks do not violate current security principles in web browsers. The attack does not attempt to compromise the web browser, nor does it attempt to compromise the user's machine. Accessing images from another web server in the course of displaying a website is common, legal behavior. As such, Puppetnet attacks present challenges for detection**;** naively disabling the ability to load objects from other websites, a practice known as hotlinking, would hurt the functionality of many existing websites.

Although similar in intellectual concept to the Puppetnets project, our work rests on different assumptions. The Puppetnets paper assumes that an attacker has control over a popular website on which she can place attack code. The authors argue in support of this assumption by pointing out that hacker and warez websites are among some of the most frequently visited sites. In contrast, we make no assumptions about the attacker's resources. We examine the ability of an

arbitrary user with very modest knowledge and resources to launch a PuppetNet style attack.

2.2 MySpace

MySpace is a social networking website, and is one of the most popular websites in the United States. [E]
Members of the site design publicly accessible profiles describing their interests, create public connections with their friends, and search and comment on their friends' profiles. As of September, 2007, over 200 million MySpace accounts had been created. [I]

2.2.1 MySpace Profiles- HTML and layouts

The centerpiece of a user's MySpace experience is her profile. A typical profile contains photos of the user, information about the user and her hobbies, a blog, and a list of other users who have been identified as friends. A subset of commands from the CSS and HTML markup languages are permitted, allowing users to greatly customize the look of their profile.

In their profile, users can post personal interests and favorite things, such as their favorite music, movies, television shows, and books. Demographic information, including the user's town of residence, date of birth, and ethnicity, can be included. Some users also choose to include information about their relationship status and lifestyle, as well as schools they've attended and companies they've worked for.

2.2.2 MySpace Friends

A key feature of MySpace is the ability to create connections with friends, creating the social aspect of social networking. When viewing the profile of another MySpace member, a user can choose to “add” that member as a friend. After the other member confirms the friendship, MySpace considers those two users to be friends.

Friendship brings with it benefits both of access control and public display. MySpace users can set privacy controls so that their blog postings or their entire profile are visible only to their friends, adding a layer of access control. In terms of public display, users' profiles contain a listing of their “top friends” on the profile itself, along with the option to view a complete list of friends. Although this benefit is primary sociological, it plays multiple important roles on social networking websites. [M] Public display of friends is a major form of social currency based on both the quantity and quality of friends that a user keeps and displays on their profile. As a result, many users are very willing to add friends and gain the associated social currency even if they are increasing access to their personal information. There is a low barrier for friendship in social networks. [G]

2.2.3 Comments

As part of the access control benefits of friendship, a user's friends are permitted to leave publicly visible comments on her MySpace profile. These comments can be used as messages to a user or as attempts to advertise a social relationship or upgrade a user's own social status. A user can choose to moderate comments posted to their profile before they are seen by others, although by default comments are not moderated. By default, some subset of HTML tags are permitted in comments, although this feature can also be disabled by the user.

2.2.4 Famous Friends On MySpace

On MySpace, a user's circle of friends can include more than her real life friends**;** she can be friends with celebrities. Famous people ranging from teen pop idols to metal bands to self-made MySpace celebrities have their own MySpace profiles and regularly accept their fans' friendship requests. Celebrities can have hundreds of thousands, or even millions, of MySpace friends. [P] MySpace Music, a section of the MySpace site that tracks bands on MySpace and provides access to information about the most popular groups, is a simple way for users to locate their musical idols on the site.

3. EVALUATING HTML FILTERING ON MYSPACE

In order to identify potential avenues for attack on MySpace, we first needed to clarify how HTML can be used on the site. Since we aimed to relax the assumption that Puppetnet Attackers already control a popular website or MySpace profile, we concentrated on the HTML that can be inserted by relatively powerless users in high traffic areas. In particular, we focused on the use of HTML in comments posted on other users' profiles.

HTML posted to MySpace in comments (and also to profiles as a whole) is filtered before it is posted. To examine which HTML tags are filtered, we created a comment containing the bare tags permitted in HTML 4.01 according to [K,PP]. Many tags were posted intact on the resulting comment. In particular, tags for text modification such as
, <address> and often passed unscathed through the filtering process. However, potentially more dangerous tags such as <a>, <base>, <bgsound>, <head>, and <img> were also permitted.

3.1 Detailed Evaluation of Potentially Dangerous Permitted Tags

In turn, we investigate and evaluate the repercussions of each permitted tag that is not primarily used to format text.

<a>, the anchor tag, allows users to link to other web sites. Although the bare tag was not visibly filtered, anchor tags are filtered behind the scenes. Outgoing links are redirected through an intermediary, msplinks.com. As a result, MySpace retains the ability to turn off links to sites containing spam, phishing attacks, or malware. [HH] However, in an attack that relies on a large amount of bandwidth, anchors are not immediately useful.

The <base> tag allows users to change the default path when using relative URLs. For instance, <a href=”page2.htm”> would normally point to the file page2.htm in the same directory as the current page. However, setting <base href=”www.mysite.com”> would instead cause the aforementioned anchor to point to www.mysite.com/page2.htm. [OO] As with anchor tags, the <base> tag is not immediately useful in an attack based on bandwith, although it could be used to improve the efficiency of constructed attacks.

The <bgsound> tag allows a user to automatically begin playback of a wav, au, or midi file in the background of a page. [S] The Mozilla Firefox web browser does not support the bgsound tag, but the more ubiquitous Internet Explorer browser does support the tag. The bgsound tag supports an autostart property that begins playback of the sound automatically, as well as a volume property that specifies the playback volume of the sound relative to full wave output. [FF] MySpace does not seem to restrict the source URL of bgsound tags, allowing this tag to point to arbitrary files at arbitrary domains.

The <head> tag demarcates a header area in which users specify a document's meta information, information about the document itself. [QQ] For instance, the title of the webpage, style tags, and scripts are contained in the head tag. Certain tags may only be placed with the header, making this tag useful.

<img> tags allow users to display an image. Relevant attributes include height and width parameters, allowing a user to specify the size of the image in terms of pixels. MySpace does not seem to restrict the source URL of images, permitting users to link to images on any domain. This process of culling images from one domain to be displayed on another domain's website is called hotlinking. Interestingly, MySpace seems to check for the existence of a source image as part of the filtering process. If an image does not exist, the entire <img> tag is replaced with two periods.

3.2 Disallowed Tags

A large number of HTML tags are not permitted in MySpace. Many of the disallowed tags are filtered by rewriting the tag and replacing < with < . For example, <script> would become <script>. On the page, the "less than" symbol would still display correctly as text, but <script> would not be interpreted as an HTML tag.

A few tags are eliminated entirely and instead replaced by two periods. These tags are: <applet>, <body>, <button>, <embed>, <input>, <isindex>, <object>, and <plaintext>. The <param> tag is replaced by <msprm>, which seems to be a custom parameter tag for MySpace. The <iframe> tag is replaced by <[iframe]>.

4. ATTACK IMPLEMENTATION

Our primary design goal for creating a MySpace Puppetnet attack is to allow an average user to direct a large amount of traffic towards an arbitrary website. We first describe the specific code that would coerce visitors to a MySpace profile to request a large amount of data from an arbitrary third party website. We then identify target locations for the attacker to post this malicious code. Finally, we provide improvements that increase the efficiency of this method.

4.1 Terminology and Assumptions

*Attacker- A malicious user who wishes to cause a distributed denial of service attack. No expertise is assumed about this user beyond good familiarity with the internet and very basic knowledge of HTML tags.

*Victim- A web server that will suffer a distributed denial of service attack. We assume that the web server hosts a relatively small number of files whose combined size is non-trivial, does not block hotlinked files, and does not extensively filter by the HTTP referrer tag.

*Target Profile- A popular MySpace profile. We assume that all target profiles that have been selected will allow HTML tags in comments.

*Puppetnet Members- Visitors to an infected target profile. Puppetnet members are unknowingly coerced to request data from the victim.

*Victim Files- Up to 30 files on the victim web server that will be requested by members of the Puppetnet during the attack.

4.2 Attack Code Contents

Of the HTML tags that pass through the MySpace filters without major modification, the <bgsound> and <img> tags most directly lend themselves to generating large amounts of bandwidth since they request multimedia files. As a result, we focus our initial construction on these two tags. We initially focus on <img> tags since they are supported in all major browsers, while <bgsound> tags are supported only in Internet Explorer.

Since <img> tags leave a visible trail, the pictures themselves, on a webpage, it is necessary to minimize this visible attack remnant. Fortunately, the <img> specification includes parameters for the height and width of images, so we choose to set both of these dimensions to 1 pixel.

We then consider how many <img> tags of the form <img src="#URL#" height=1 width=1> can fit in one comment. In tests using a 30 character URL, we were able to fit 32 <img> tags of that form into a comment. Although we cannot assume an arbitrary URL will fit in 30 characters, URL redirectors and shorteners (as described in Section 4.4) make this a realistic assumption. Since the number of <img> tags possible in a comment is inversely proportional to the size of each <img> tag, a more efficient way of crafting equivalent <img> tags would increase the attack bandwidth.

The efficacy of this constructed comment assumes the existence of approximately 30 unique image files on the victim's webserver. The bandwidth drawn by this attack is directly proportional to the total size of those 30 image files located on the victim's server. The mechanism for discovering potential target images is outside the scope of this work, although a Google Image search for “extra large files” or running a web spider on the victim are both logical first steps in the discovery process.

4.3 Attack Code Location

While the average user's own MySpace profile does not receive enough traffic to create a Puppetnet attack, a celebrity's MySpace profile certainly receives a large amount of traffic. In particular, the comments section of a celebrity's profile seems like a logical place to post malicious code since it will be seen by all of the visitors to that celebrity's MySpace page.

There exist three potential roadblocks to posting a comment on a celebrity's MySpace page. The first and most trivial barrier is that an attacker needs to become MySpace friends with the celebrity. Fortunately, there is a low barrier to friendship. Since most bands and celebrities want to expand their fan base and receive free advertising, the vast majority of MySpace celebrities indiscriminately accept friend requests and thus accumulate hundreds of thousands or millions of friends. We experimentally verified this assumption, and are now intimately connected with everyone from the rock group Weezer to MySpace star Tila Tequila.

The remaining two roadblocks seem more serious, yet both can be easily overcome. Most of MySpace's biggest celebrities do not permit HTML comments on their profile, choosing to change the default setting. We visited the MySpace profiles of 25 popular major label bands, including the 10 most popular major label bands that day on MySpace music, and did not find any profiles that permitted HTML comments.

However, while major MySpace celebrities generally do not permit HTML comments, minor MySpace celebrities and independent bands often do permit HTML comments. In addition to major label artists, MySpace Music also tracks the current most popular artists in both the “independent record label” and “unsigned band” categories, providing a potential set of fairly popular MySpace profiles that nonetheless would likely permit HTML.

The final roadblock in posting attack code regards MySpace profiles whose comments are moderated. Understandably, some bands and celebrities would like to maintain control over the posts on their MySpace page and thus choose to moderate the comments posted. Moderated comments become troublesome when an attack becomes obvious to the moderator. An attacker could assume that moderators would not notice that a comment is taking a long time to load. However, even this assumption is unnecessary. To disguise comments when they are in the moderation phase, we turn to URL redirection.

4.4 URL Redirection

There exist a number of URL redirection services, including TinyURL and SnipURL, that allow users to shorten long URLs. However, many of these services also allow for dynamic redirection of URLs, removing the third and final roadblock. When submitting a comment on a moderated MySpace profile, an attacker can redirect to images that load quickly. For instance, awaiting moderation, the malicious comment might point to thirty one pixel images, whose combined load time is negligible. Following approval by the moderator, the URL redirection services could be instructed to redirect the shortened URL to large images on the victim's webserver.

Two assumptions are at work in this example. We first assume the existence of one pixel images with the same file names as large images on the victim's web server. This assumption is trivial, though, since an attacker can post the desired one pixel images herself on

any web host that allows its images to be hot linked. We also assume that the URL redirection service allows the redirection target to be changed after some period of time. SnipURL, which we use in our proof of concept example, does indeed allow this behavior.

URL redirection brings with it two ancillary benefits. First of all, an attacker can now easily change the target of the attack even after posting the malicious comments on MySpace. For instance, if one victim successfully implements countermeasures against a Puppetnet Attack in progress, the attacker can quickly and easily redirect the attack towards another victim. The attacker is also able to shorten the URLs of target images by using URL redirection services and thus create more efficient attacks within the limited space allotted to MySpace comments.

4.5 Generalizing The Attack To Non-Media Files

In order to generalize the MySpace Puppetnet attack to work with all file types, rather than just images and .wav files, it's important to evaluate how web browsers handle non-image filetypes inside the <img> tag and non-audio filetypes inside the <bgsound> tag. Since Internet Explorer handles both the <img> and <bgsound> tags, our testbed consists of a Windows XP computer running Internet Explorer 6.0.2900. We post three sets of files, with each set containing both a 4.00mb executable (.exe) and a 3.98mb zip file of that executable (.zip), to our personal web host. Exe and Zip files are chosen because of their ubiquity on the web and large file size. We clear the local Internet Explorer cache and then open a MySpace page containing the following HTML in a comment:

```
a) <img src="http://snipurl.com/1v88i" width=10 height=10>
[which points to /wtwo.zip]
b) <img src="http://snipurl.com/1v8hb" width=10
height=10>   [which points to /wtwo.exe]
c) <bgsound src="http://www.blaseur.com/wone.zip?a.wav">
d) <bgsound
src="http://www.blaseur.com/wone.exe?a.wav">
e) <bgsound src="http://www.blaseur.com/wwone.zip">
```

Lines a and b are crafted to test how Internet Explorer handles zip and exe files when they are referenced in an <img> tag. Lines c and d are crafted to test how Internet Explorer handles zip and exe files when they are referenced in a <bgsound> tag. Line e tests whether adding "?a.wav," passing a non-existent wave file as a parameter, is even necessary to fool the web browser into downloading an inappropriate file type. If the file referenced in line e is downloaded completely, then adding these extra parameters is unnecessary.

We then downloaded the raw access logs from our web host and evaluated how much data had been downloaded. The following series of GET requests correspond to the above HTML:

```
a') "GET /wtwo.zip HTTP/1.1" 200 259068
b') "GET /wtwo.exe HTTP/1.1" 200 247743
c') "GET /wone.zip?a.wav HTTP/1.1" 200 3977792
d') "GET /wone.exe?a.wav HTTP/1.1" 200 4000588
e') "GET /wwone.zip HTTP/1.1" 200 3977792
```

When a zip or exe file is included in an <img> tag, about 250kb of a 4mb file is downloaded. Our results suggest that the web browser begins downloading these files and then aborts the download when it is apparent that the requested file is not an image. However, when a zip or exe file is included in a <bgsound> tag, the entire file is downloaded. Furthermore, no kludges or fake parameters seem necessary to fool the browser into thinking that a wav file is being downloaded, as seen by the equality between lines c' and e'.

In the Mozilla Firefox web browser, preliminary results suggest that non-image files are partially downloaded and then aborted when pointed to by <img> tags, as in Internet Explorer. However, the <bgsound> tag is not supported in Firefox. We leave the further exploration of alternative browsers, differences among browser versions, and the exploration of other large internet files (including video files and applets) to future work.

Overall, MySpace visitors using Internet Explorer can be coerced to download common non-image files in full using only the <bgsound>

tag. Because Firefox does not support the <bgsound> tag, the bandwidth consumed by non-media file coercion is sharply limited although non-trivial. However, since Internet Explorer enjoys a market share of over 80% in the web browser market, most members of the Puppetnet will likely be using Internet Explorer. [H]

4.6 Detailed Attack Outline

Based on the aforementioned improvements and subtleties, we recap the necessary steps for a potential MySpace Puppetnet attack:

Step 1) The attacker locates up to 30 static files publicly available on a victim web server and notes the URLs of each of these files. These files should be chosen exclusively for their large size. For this step, a specialized tool can be designed to search for appropriate targets or appropriate files on a given target.

Step 2) The attacker creates a MySpace profile, searches for celebrities as target profiles, and adds all of these target profiles as friends. If a target profile moderates its comments or if the URLs of the victim files are long, the attacker also creates an account on SnipURL or a similar URL redirection service. The attacker waits until all of her friend requests have been accepted.

Step 3) If the target profile moderates its comments, all URLs on the victim's website should be temporarily replaced by SnipURLs using the account created in Step 2. The attacker creates SnipURLs that all point towards a single 1 pixel image on any server.

Step 4) The attacker creates a series of HTML tags that point to the files identified in Step 1. For victim files that are images, the attacker uses a series of <img> tags whose height and width are 1 pixel. For victim files that are not images, the attacker uses <bgsound> tags. The attacker then posts this code as a comment in each of the target profiles.

Step 5) (Only if SnipURL has been used to bypass moderation). After the posted comments have passed moderation, the attacker redirects the relevant SnipURLs to point to the victim files.

Step 6) As comments are pushed out of the first page of a profile by more recent comments, the attacker reposts her original comment.

5. QUANTIFYING A MYSPACE PUPPETNET

Although there are many variables that affect the efficacy of a MySpace Puppetnet Attack, we estimate the potential of an average attack. From a construction perspective, the number, types, and size of files available on a victim web server are the variables that most directly influence the damage possible. Defense methods which also significantly reduce the damage of a MySpace Puppetnet attack are discussed in Section 7.

In quantifying a MySpace Puppetnet attack, we make conservative estimates about the files available on a victim web server and conservative estimates about the number of target profiles on which the malicious comment will be posted. However, we assume that the victim web server is not doing any sort of filtering. This assumption is reasonable primarily since HTTP filtering for this sort of attack is far from universally implemented.

We first identify the number of daily hits to target profiles, those profiles on which an attacker can post malicious code. We then estimate the bandwidth of potential comments and calculate an overall estimate.

5.1 Daily Profile Hits

We earlier hypothesized that “b-list” celebrities and independent musical artists would likely permit profile comments containing HTML even though many top celebrities block HTML comments. In December, 2007, we visited the MySpace profiles of the current 20 most popular

unsigned bands and 20 most popular independent label artists, as identified by MySpace Music, to verify this hypothesis. On each of these profiles, we visually inspected whether comments contained images. If comments on that profile did contain images, we concluded that the profile permitted HTML.

Of the 40 profiles we examined, 11 contained comments with HTML. We then proceeded to estimate the daily traffic at each of these profiles by dividing the total number of profile visitors (over the life of the profile) by the number of days since the profile had been created. Estimates of the number of profile views per day for these 11 artists is contained in Figure B.

Artist	Profile Views Per Day
I Set My Friends on Fire	25,400
Millionaires	15,100
Hurricane Chris	13,100
Flo Rida	11,800
Jon Young	9,300
Pretty Willie	7,600
Sugarcult	7,200
Mindless Self Indulgence	7,000
Jordyn Taylor	4,000
Yo Gotti	3,300
3rd Flo	1,600

Figure B. The average number of profile views per day for the musicians among the current top 20 unsigned or independent label bands was computed by dividing the total number of profile views by the number of days the profile had been active. These 11 profiles combined for 105,400 views per day.

These 11 profiles receive a total of 105,400 views per day. While less popular artists would receive a smaller number of views, we have found also located a large number of additional groups who receive 500-1,500 daily profile views. To provide a conservative estimate on a MySpace Puppetnet, we use just these 11 profiles as the target profiles in quantifying the attack. It is, of course, important to note there exist a finite number of vulnerable profiles and any attack scales linearly, so a very large increase in the number of profile views requires a large increase in the attacker's effort.

5.2 Size of a Single Comment

The size of a single comment is greatly variable with the characteristics of the victim webserver. However, we provide rough estimates on the different file sizes possible in different scenarios.

Pictures from a fairly recent digital camera with high JPEG quality are generally 2 - 4 megabytes each. Since we can fit approximately 30 photographs in a comment, a single comment of JPEG photos would draw roughly 100 megabytes each time it was loaded. This scenario would be commonly encountered if attacking a professional or even amateur photographer's website, or the website of users who uploaded digital photos straight from their camera. On commercial image hosts, most image files are small and heavily compressed. However, photographers often try to maintain high quality images of their work.

Musicians or music enthusiasts who host their own website often have mp3 files, which tend to range from 3-10mb for an average song. Assuming an average of 5mb per mp3 file, an attacker can craft 150mb comments of mp3 files by leveraging the <bgsound> tag. Similarly, videographers' websites are vulnerable because of the large media files they host. Software developers and distributors are also vulnerable with any .exe or .zip files present on their website. A typical software package may vary from 100 kilobytes to 100 megabytes.

On the other end of the spectrum, a small website with only a few images may total just a few hundred kilobytes of unique files. As such, we consider comments of 4 orders of magnitude: 100 kilobytes, 1 megabyte, 10 megabytes, and 100 megabytes. Since the attack scales linearly, the bandwidth of each of these attacks increases by an order of magnitude.

5.3 Total Number of Coerced Requests

We assume that the attacker crafts a comment that requests approximately 30 unique objects (of any size) from the victim. Since in our example, posting a comment to 11 judiciously selected target profiles resulted in approximately 105,400 daily pageviews, 3,162,000 requests (hits) are submitted to the victim server each day. This is equivalent to an average of over 2,000 hits per minute. In contrast, it has been estimated that the "Slashdot Effect" similarly results in just a few thousands hits per minute. [A,II]

5.4 Bandwidth Analysis

The total daily bandwidth for an attack is calculated by multiplying the estimated number of daily profile hits (105,400) by the size of each comment. This product assumes that each visitor waits for the page to load rather than leaving a slow loading page. Since many internet users have broadband connections, it is not unreasonable to assume that a large percentage of Puppetnet members do indeed allow the full page to load. However, the total bandwidth might scale down by a small linear factor based on users leaving the page before it is completely loaded.

At the low end, a 100kb total comment seen by 105,400 visitors in a day would result in 10 gigabytes of daily traffic. While 10 gigabytes of daily traffic is not trivial, it would generally not cause a huge disturbance for many modern servers. A 1 megabyte comment would generate 100 gigabytes of daily traffic, which would cause a moderate problem for some web hosts. Similarly, a 10 megabyte comment would generate 1 terabyte of daily traffic, causing significant problems for many smaller web hosts.

In what might be considered a worst case, albeit absolutely feasible, scenario, a 100 megabyte comment attacking media-heavy sites such as those of photographers or musicians would generate 10 terabytes of daily traffic. This is a significant amount of traffic and would likely cripple many web servers.

6. ATTACK INVARIANTS AND CHARACTERISTICS

Invariants are those elements which must be present for an attack to work. Characteristics are those elements which are not essential for an attack but often indicate that an attack is in progress when they are present. In order to evaluate potential defenses from MySpace Puppetnet attacks, we first describe the invariants present in each MySpace Puppetnet Attack. In Section 7, we propose and critically evaluate defenses based on these invariants.

Attack Invariant 1: HTML tags point to files on a third party server

In order for a Puppetnet Attack to exist, some HTML on a MySpace page must point to files on a non-MySpace server. Furthermore, this HTML will likely not point to a popular third party media server that is well equipped to handle a very large number of bandwidth intensive requests, but rather to a significantly less capable webserver that will fall victim to the distributed denial of service attack.

Attack Invariant 2: A large number of HTML tags are present in a MySpace comment.

In general, there is no need for users to place a large amount of HTML in comments, particularly if the tags used are not associated with text formatting. MySpace users may post an image urging others to visit their MySpace profile, but they would generally not include dozens of images. In contrast, an effective MySpace Puppetnet attack makes efficient use of the space provided in a comment and fills it with HTML tags requesting objects from the victim.

Attack Invariant 3: The HTTP Referrer of Most Puppetnet Members will be the same

The log files on a victim server will show a very large number of requests from the same set of referrers, namely the targeted MySpace pages which contain the attack code. While the display of referrer information is not entirely accurate and

can even be disabled by some web browsers (i.e. Opera), proxies, and firewalls, a large percentage of users will still betray the referrer.

Attack Invariant 4: The same set of files will be requested by each visitor.

Barring a large number of aborted requests, erratic page loading, or rapid switching of a URL redirection service, Puppetnet members statically request the exact same objects from the victim. Since scripting is not permitted by the MySpace filters, the objects are hardcoded into the HTML and trying to invalidate this invariant by rapidly switching the target of a URL redirection service is cumbersome and suspicious at best.

Attack Invariant 5: When present, images are artificially made very small e.g. 1 pixel by 1 pixel

Since a 1 pixel by 1 pixel or similar image is not easily visible, images that small are likely not intended to be seen. These images can be used to stealthily draw a large amount of bandwidth. Alternatively, nearly invisible images can be used to track visitors. Regardless of their intent, 1 pixel by 1 pixel images often have some nefarious purpose and very little legitimate purpose.

Attack Characteristic 1: <img> and <bgsound> tags point to files that are not images or audio files.

If a file is referenced by an <img> or <bgsound> tag yet is not compatible with that tag, there is likely no legitimate reason to download that file. However, this sort of behavior permits attackers to generalize the MySpace Puppetnet attack to all available files.

Attack Characteristic 2: Extensive URL shortening is used.

If the HTML in a comment references URL shortening or URL redirection websites, this layer of indirection may be used to maximize the efficiency of a Puppetnet attack or redirect traffic to bypass moderators or dynamically switch targets. The primary use or URL shortening services, to shorten the link to some outside file or website, is irrelevant since MySpace automatically redirects links through their own msplinks.com domain.

7. ATTACK DEFENSES

It is possible to describe and evaluate potential defenses against MySpace Puppetnet attacks based on the attack's invariants and characteristics. Defenses can be implemented at two places: Either at all possible victim webservers, or at MySpace (the attack vector).

Victim's Defense 1: Block Referrers from the MySpace Domain

All webservers could choose to block access to any visitor whose HTTP referrer indicates that they are coming from MySpace.com. Although it is possible to prevent referrer information from being sent in certain browsers and situations, this defense would significantly lessen the impact of a MySpace Puppetnet attack. However, all legitimate traffic generated from posts on MySpace would be needlessly blocked, eliminating the use of word of mouth on MySpace to generate legitimate traffic.

Victim's Defense 2: Filter by Referrer

Rather than indiscriminately blocking all traffic from MySpace.com, a victim webserver could actively analyze the referrers of traffic to its page. If a particular MySpace profile refers a large number of users to a webserver to request many large files, then all subsequent requests from that referrer could be dropped. While the server would still receive a large number of hits, the bandwidth would be greatly reduced by dropping the requests that come from the suspicious referrer. While filtering by referrer is a well-studied problem, it is quite far from universally implemented.

Victim's Defense 3: Filter a Set of Files

While filtering by referrer is effective for those web browsers that submit referrer information, filtering requests based on a certain set of files also deserves consideration. Since a MySpace Puppetnet attack statically points to some easily identifiable subset of files, requests for this subset of files can be dropped. Access to these files can be blocked entirely, or requests for files in this set can be dropped after some k-subset of these files has been requested by one IP address. This technique leverages the static nature of a MySpace Puppetnet attack. However, this approach conflates popularity with an attack. A file that is popular will seem suspicious and be blocked, denying service to legitimate users who request it.

Victim's Defense 4: Prevent Hotlinking of Images

While preventing hotlinked images is a well-studied problem and easily solved on a web server, it is not universal or even very widespread since it is permitted by default. If all servers were to disallow hotlinked images and files, MySpace Puppetnets would be powerless. However, there exist legitimate uses for hotlinked images and files and thus it seems hasty to prevent hotlinking.

MySpace's Defense 1: Filtering out Third Party URLs

Since MySpace currently filters HTML posted to the site, adding a rule filtering out third party URLs in HTML tags (as is currently done for links) would entirely prevent MySpace Puppetnet attacks without much effort. However, this approach would also prevent MySpace from taking advantage of other media hosts' bandwidth. If MySpace disallowed third party URLs in images, MySpace would then need to host these images itself in order to provide the same experience to users. While this approach would potentially simplify MySpace users' experience in posting images, it would require increased server capabilities.

MySpace's Defense 2: Whitelisting HTML

MySpace currently blacklists HTML, filtering out tags it deems inappropriate or malicious. Instead, MySpace could whitelist some subset of HTML and provide a graphical interface for users to take advantage of this HTML. Users would no longer be burdened with learning or copying HTML. Instead of posting bare HTML, the user would indicate that they want to insert an image, specify its location, height, and width in the interface, and click a button. This whitelisted HTML could enforce very fine grained policies. Perhaps only 1 image would be permitted in a comment, but an arbitrary amount of formatting tags would be permitted. Unfortunately, this technique removes freedom from users and stifles creativity and innovation, sentiments which have caused previous ill will among users. [JJ] However, given the popularity of third party "MySpace Profile Editors," this defense may actually prove popular among less technically inclined users. [J]

7.1 Choosing A Defense

In comparing the different possible defenses, it is important to note that actions by potential Puppetnet victims require that every web host in the world update their servers and implement some sort of filtering or else remain vulnerable. In contrast, defending against MySpace Puppetnets from MySpace's end requires changes in just one centralized location. As such, the most effective and most efficient defenses would be implemented by MySpace rather than the victims.

In deciding whether to filter out third party URLs or migrate to a system where HTML is whitelisted based on what is considered reasonable usage, it is important to consider the potential effects on the MySpace user experience. Puppetnet defenses should cause as few disruptions as possible to end users. At a high level, filtering third party URLs in plain HTML tags would primarily prevent hotlinking images and .wav files. If there existed a mechanism through which potential third party collaborators

could integrate their content into MySpace through some sort of OpenSocial or Facebook Application specification, there would be no need to give individual users unrestricted freedom to hotlink content through HTML. Instead, trust would be placed in a small number of developers rather than a large number of anonymous users. As an added bonus, changes in filtering by MySpace would not break third party applications. Indeed, OpenSocial and similar platforms will be supported by MySpace in the near future, making the value of bare HTML questionable.

Furthermore, MySpace hosting all images used in profiles allows the company to better control what is visible to its users. A third party image host could potentially change a popular image to display something lewd, and this change would be out of MySpace's control. Similarly, if an image were deleted from a third party host, it would no longer be visible on MySpace. While there is a non-trivial infrastructural cost associated with image hosting, MySpace already has a large infrastructure built for hosting music and images. Hosting additional images would likely only minimally increase this hosting burden.

While further discussion of specific defense choices is left to future work, proactively implementing defenses on MySpace rather than potential victim webservers is seen to be more expedient.

8. RELATED WORK

8.1 Other Puppetnets

Attacks in the spirit of the Puppetnet concept have been explored in the literature, though under different names. These attacks all trick a large number of correctly operating hosts into collectively launching an attack. In [GG], a method for launching distributed denial of service attacks using “reflectors” is described. An attacker sends requests to a large number of correctly operating web servers, termed reflectors. However, these requests have been spoofed so that they appear to come from a victim website. The reflectors then perform their correct behavior and send the requested data to the victim, who did not actually request that information. If the size of the data sent to the victim is significantly amplified compared to the size of the requests, a reflector attack can be dangerous. Gnutella, a peer-to-peer file sharing network, is among the attack vessels investigated as a potential source for reflectors. However, reflector attacks generally rely on spoofing, which in many cases can be detected. [F] Our work does not require any spoofing, so it is not vulnerable to this particular defense.

A peer-to-peer Puppetnet that does not require spoofing is observed in [Z]. In this example, DC++ clients connected to DirectConnect hubs are redirected to request information from arbitrary victim servers, generating around one gigabit of data per second from the victim servers. Our work diverges from this attack since we do not require our “puppets” to be using any particular software other than a standard web browser. Our work draws from a much larger pool of potential puppets.

A Puppetnet style attack is also seen in [R]. In this attack, web browsers are used to create “referrer spam,” in which an attacker's website appears to direct a large amount of traffic to some victim website. As in our work, <img> tags are used to generate this traffic. However, the purpose of “referrer spam” is not to attack the bandwidth of the victim site, but rather to request non-existent objects. As a result, the attacker will appear to be magnanimously referring valuable web traffic to the victim and thus appear on a blogger's list of top referring websites. In essence, the attacker feigns goodwill in order to fraudulently gain a link from the victim's website.

8.2 Web 2.0 Security

MySpace, our proposed attack vessel, is considered a Web 2.0 site. Web 2.0 is broadly defined as those websites that enable significant user-generated content, often leveraging

technology such as AJAX (Asynchronous Javascript and XML). However, with this user-generated content comes security risks and unintended consequences. [DD]

Well-studied yet still prevalent attacks in Web 2.0 applications include Cross Site Request Forgery (CSRF) attacks and Cross Site Scripting Attacks (CSS). [O] CSRF attacks impel a victim's web browser to unknowingly carry out some action while they are legitimately logged in to one of their accounts. For instance, if a user is currently logged into their bank's website, a CSRF attack might direct that bank to transfer money. In contrast, an XSS attack injects code into a web page, code which is often visible to (and thus executed by) other users' browsers.

The Samy Worm on MySpace and Yamanner Worm on Yahoo are two widespread examples of Web 2.0 attacks. In the Samy worm, any MySpace user who viewed Samy's profile automatically added Samy as their friend and their "hero." The worm was then inserted into that user's page, and thus the worm spread exponentially. Within 24 hours, over 1 million MySpace users had been infected. [V] In the Yamanner Worm, specially crafted emails that were viewed inside Yahoo's webmail service automatically sent copies of themselves to addresses on the Yahoo user's contact list. [L] This worm is notable for operating based on HTML and Javascript tags in a web browser rather than in a standalone email client on a user's local machine.

For Web 2.0 attacks to work, malicious code must be inserted, often relying on trickery to bypass security mechanisms. To combat these attacks, it is widely recommended that user input to Web 2.0 sites be properly sanitized at the web server level before being displayed. [BB,KK] However, both the Samy Worm and Yamanner Worm exploited loopholes in input sanitization. The Samy Worm took advantage of the newline (\n) character in the middle of the Javascript tag and the ability to evaluate concatenations of strings in order to bypass MySpace filters. The Yamanner worm used a malformed HTML tag, tricking the filtering that was designed to catch it. When Yahoo automatically filtered out the malformed target="" part of a tag that included *target=""onload=*, the onload tag was included in the webpage even though the filters were designed to remove it. Many subtle tricks exist to pass by filters designed to prevent scripts from being passed into Web 2.0 applications, and the attempt to filter input is essentially an arms race between attackers and defenders. [LL]

Work in preventing XSS and similar attacks focuses on eliminating or controlling the use of Javascript in webpages. Approaches include blacklisting or whitelisting individual scripts so that a webmaster can control which scripts execute on her page [U]**;** automatically bundling <noexecute> tags with popular library functions used for creating interactive content [CC]**;** instrumenting Javascript code which users can choose to run or disallow [SS]**;** using context free grammars and parsing to stop command injection [MM]**;** allowing high level policies to dictate Javascript code execution [Q]**;** and giving web browsers a greater role in enforcing security policies [N].

However, each of these approaches to prevent XSS attacks does not apply to stopping Puppetnet attacks launched in a web browser. These defense mechanisms all aim to detect and prevent unauthorized behavior and attempts to bypass filters. In contrast, a Puppetnet attack exploits perfectly legal behavior in a web browser, but uses this legal behavior in an unintended fashion. No loopholes are necessary.

8.2 Social Networking Sites as Attack Vessels

The MySpace Puppetnet attack uses MySpace, particularly MySpace comments, as the attack vessel. While the use of celebrity's comment section for an attack is a novel contribution, social networking sites are already used as vessels for other attacks. As described in [Y], social networking websites are attractive for attacks since they are popular, let users upload content, and often require some scripting capabilities. Many existing social networking

exploits take advantage of software bugs. In the wild, malware ads leveraging the Windows WMF vulnerability and Quicktime vulnerabilities have been observed. [W,AA] One exploit in particular even used celebrities on MySpace as the attack vessel. This attack focused on the MySpace pages of famous musicians, including Alicia Keys, in order to attract a large number of page views. [RR] This attack redirected users to another webpage which tried to download a fake codec, a suspicious behavior. It is believed that a phishing attack allowed the attackers to post malicious code (a very large background image used as a link) on certain celebrities' pages. [NN] In contrast, our attack does not require the celebrity's login information; instead, it can be performed by anyone.

It is important to note that the MySpace Puppetnet attack is specific to sites that permit users to place HTML code on other user's profiles. For instance, a comparable attack is not immediately obvious on Facebook or many popular blogging websites where HTML insertion is not permitted. In this sense, the MySpace Puppetnet attack can be seen as an argument against unrestricted HTML use. Although scripting languages and certain HTML tags are viewed to be dangerous, this work expands the reach of what tags and behaviors should be viewed with suspicion.

8.3 Using Small Images Maliciously

While, to our knowledge, we are the first to suggest using images on social networking sites to launch a distributed denial of service attack, there currently exist other questionable uses of images on MySpace. A handful of third party sites have been developed to allow users to track who sees their profile. [EE] These third party profile trackers ask MySpace users to create an account on their site. The user is then presented with a small snippet of HTML code containing a 1 pixel image whose URL contains a unique string. The user places this code in their MySpace profile. Visitors to that profile now request the 1 pixel image from the third party tracker, revealing their IP address as part of the request. All IP addresses that request a given image can be assumed to have visited the associated MySpace profile. If both the profile owner and profile visitor have signed up with the third party tracker, the tracker can use some additional trickery and log the MySpace username of the visitor and present this information to the owner. However, in the general case, only an IP address is tracked.

The Puppetnet attack we have described also permits tracking. However, our approach suggests that an arbitrary user can track visitors' IP addresses on any profile that permits HTML comments, rather than just their own. An attacker need only post a comment with a 1 pixel image hosted on her own web host. By logging the requests to her own web server and examining the referrer information, the attacker can identify the IP addresses of visitors to the profile.

8.4 XSS PuppetNets

A series of short papers released around the time of the Puppetnets project explores cross-site scripting vulnerabilities and suggests that they might be used to launch, among other malicious uses, distributed denial of service attacks. [B,C,D] These papers explore the necessary conditions and implementations of self-propagating XSS viruses that eventually can be assembled into a controllable zombie framework of browsers. Among other contributions, these papers concentrate on technical details related to virus propagation and communication in the zombie network. In contrast, we explore a specific attack vector in current Web 2.0 architecture and quantify the possibility of these attacks. We also focus equally on defense mechanisms rather than maximizing the efficacy of zombie computers.

9. CONCLUSIONS AND FUTURE WORK

We have made three primary contributions in this work. First of all, we have demonstrated a novel method for a powerless and arbitrary user to launch a distributed denial of service attack against an arbitrary web server. We only assume that the victim web server makes available some set of reasonably large, static files and that the victim does not actively block connections whose HTTP referrer is MySpace. Secondly, we have demonstrated an attack launched on a social networking website that does not violate the spirit of current security models, even though it may its resource usage may be unreasonable. In contrast, previous attacks on MySpace and other sites have taken advantage of loopholes in filtering (i.e. the Samy Worm) or known file handling vulnerabilities (such as those in WMF and Quicktime files). However, our attack merely requests files, which is a normal behavior in a web browser and on a website. Finally, we have identified the MySpace comments of moderate celebrities as a potential attack vector for attacks using a small subset of HTML. Had the Samy worm been launched using comments on popular pages rather than in his individual profile, the worm would have spread much more rapidly.

In essence, we have demonstrated why it might not be prudent to allow arbitrary users to post any sort of code, even in a scripting language like HTML, on a popular public space. We could even consider this oversight to be a violation of the principle of least privilege. Is it really necessary for users to have the ability to post HTML on someone else's MySpace page, or should they be restricted to using HTML to customize just their own profile? If there does not seem to be a need or even tangible advantage to giving users this permission, perhaps it should be disallowed. This argument also ties into the constant arms race between attackers and security vendors and services. By giving users relatively generous permissions and then blacklisting HTML that is known to cause problems, are the architects of MySpace escalating this arms race? Would it be preferable to instead whitelist only certain HTML privileges, such as adding a single image hosted on certain pre-approved hosts. If this were the case, a GUI could be provided to users, perhaps fitting in better with the fairly young and technologically inexperienced MySpace demographic.

An interesting observation that came to light in this work is the disparity between web brosers in handling certain HTML tags. Although the existence of differences between browsers and the potentially overly generalized HTML specifications is a well known problem in web design, we were able to take advantage of the disparities surrounding the <bgsound> tag to generalize our attack. It does not seem reasonable that asking a web browser to play a file with the .exe extension as audio would be permitted, suggesting that either stricter implementation of stricter HTML standards are in order.

One conclusion that makes more sense in the context of social networking history is the reliance of MySpace on third party media content. Based on our work, it seems reasonable for MySpace to disallow hotlinking of content from other sites as long as it provides a robust mechanism for hosting its own content, such as images. However, the historical perspective elucidates this design decision. MySpace initially had difficulty keeping up with very quick initial growth in user demand, so allowing third parties to host media content was initially beneficial. However, at this point in time, MySpace should consider revisiting this policy and potentially ban hotlinking. Particularly if the OpenSocial initiative is well adopted and MySpace adds many widgets and applications, HTML may not be necessary for users to customize their MySpace profiles. At the least, it seems that HTML should be turned off automatically for very popular MySpace users, particularly in their comments section.

As future work, one useful direction would be to identify the differences between browsers and browser versions in how <img> and <bgsound> tags are handled. On some browsers, do <img> tags that point to executable files nonetheless download the entire executable file,

or do all browsers abort? Furthermore, identifying browser differences becomes important in filtering by referrers. For instance, which browsers allow users to prevent the submission of referrer information? A detailed study of multiple versions of popular browsers would allow for a more informed attack analysis. Additionally, Internet Explorer 6.0's seemingly liberal handling of all file types with the <bgsound> tag seems like a possible avenue for attack. Is it possible to cause damage by leveraging the fact that an arbitrary file can now be downloaded to the local cache on a user's machine just from visiting a webpage?

The extent to which a MySpace Puppetnet attack can be made less effective by large scale caching is also an interesting problem, particularly since a static set of files is always downloaded. To what extent do the large web caches, as well as local caches at the borders of universities and ISPs, cache the files used in this attack? A potential experiment would be for a researcher to post a very small image on her web server and then insert that image into a MySpace comment on a popular profile. By comparing the number of hits on the MySpace page to the number of HTTP GET requests logged for that image file over a certain time frame, the effect of caching can be roughly estimated.

Another area of future work exists in providing a deep analysis of how to construct malicious Puppetnet comments automatically. It seems possible, although not entirely trivial, to create a script that automates Google searches or spiders a web page in order to identify large files that can be hotlinked. Once a tool like this is created, running this tool on a small subset of existing webpages would significantly tighten the bounds on the amount of damage victims would endure during a MySpace Puppetnet attack and better allow researchers to put the attack in context.

A final area of future work involves investigating the application of existing flash crowd detection techniques to reducing the flash crowds that are observed in the MySpace Puppetnet attack, particularly since all members of this flash crowd are requesting the same static set of resources. What do previous work on flash crowds and previous investigations of the Slashdot effect suggest about additional defense mechanisms?

ACKNOWLEDGEMENTS

I'd like to thank Vinod Ganapathy for teaching this course, giving me an introduction to software security, and very much supporting my project ideas. I'd also like to thank Rutgers, Dean Don Brown, Dean Ilene Rosen, and Professor Allender for giving me the chance to take this course. I'd finally like to thank Rachna Dhamija, Professor Michael Smith, and Simson Garfinkel for advising and supporting me during my undergraduate thesis process and helping me tremendously in learning to do research.

REFERENCES

[A] Stephen Adler. The Slashdot Effect, An Analysis of Three Internet Publications.

[B] W. Alcorn. The Cross-Site Scripting Virus. 2005.

[C] W. Alcorn. The Advanced Cross-Site Scripting Virus. 2006.

[D] W. Alcorn. The Browser Exploitation Framework. http://www.bindshell.net/tools/beef

[E] Alexa.com. MySpace History Graph. http://www.alexa.com/data/details/traffic_details/myspace.com

[F] K. Anagnostakis. PuppetNets: Misusing Web Browsers as a Distributed Attack Infrastructure. (Presentation).

[G] danah boyd. Friends, Friendsters, and MySpace Top 8: Writing Community Into Being on Social Network Sites. First Monday 11(12).

[H] BusinessWire. Tuf-of-War between Firefox and Internet Explorer Continues.

[I] Pete Cashmore. 200 Million MySpace Accounts. Mashable. September 9, 2007.

[J] Pete Cashmore. MySpace Layouts Top 10. Mashable. July 28, 2006.

[K] CDI's Comprehensive HTML Cross Reference http://www.willcam.com/cmat/html/crossref.html

[L] E. Chien. Malicious Yahooligans. Virus Bulletin. 2006.

[M] J Donath and danah boyd. Public Displays of Connection.

[N] U. Erlingsson, B. Livshits, and Y. Xie. End-to-end Web Application Security.

[O] J. Grossman. Javascript Malware. Presentation. 2007.

[P] Lev Grossman. Tila Tequila. Time Magazine. December 16, 2006.

[Q] O. Hallaraker and G. Vigna. Detecting Malicious JavaScript Code in Mozilla. 2004.

[R] M. Healan. Referer Spam. Spywareinfo.com. 2003.

[S] HTML Reference Guide. <bgsound> http://www.html-reference.com/BGSOUND.htm

[T] C. Jackson, D. Boneh, A. Bortz, and J. Mitchell. Protecting Browser State from Web Privacy Attacks. WWW 2006.

[U] T. Jim, N. Swamy, and M. Hicks. Defeating Script Injection Attacks with Browser-Enforced Embedded Policies. WWW 2007.

[V] S. Kamkar. Technical Explanation of The MySpace Worm. 2005.

[W] B. Krebs. Hacked Ad Seen on MySpace Served Spyware to a Million. Washington Post Security Fix Blog. 2006.

[X] V. Lam, S. Antonatos, P. Akritidis, and K. Anagnostakis. PuppetNets: Misusing Web Browsers as a Distributed Attack Infrastructure. CCS 2006.

[Y] G. Lawton. Web 2.0 Creates Security Challenges. IEEE. 2007.

[Z] R. Lemos. Peer-to-Peer Networks Co-opted for DOS Attacks. SecurityFocus.com. 2007.

[AA] R. Lemos. QuickTime Worm Uses MySpace to Spread. SecurityFocus.com. 2006.

[BB] E. Levy and I. Arce. New Threats and Attacks on the World Wide Web. IEEE Security & Privacy. 2006.

[CC] B. Livshits and U. Erlingsson. Towards Security by Construction for Web 2.0 Applications. W2SP 2007.

[DD] R. McMillan. Researchers: Web 2.0 Security Seriously Flawed. PCWorld. 2007.

[EE] Mixmap. Mixmap.com

[FF] MSDN. Volume Property <bgsound>. http://msdn2.microsoft.com/en-us/library/ms535140.aspx

[GG] V. Paxson. An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks. Computer Communications Review. 2001.

[HH] Sean Percival. MySpace Begins Filtering Outgoing Links. April 23, 2007.

[II] Slashdotting. http://www.astro.princeton.edu/~mjuric/universe/slashdotting/

[JJ] Brad Stone. MySpace Restrictions Upset Some Users. New York Times. March 20, 2007.

[KK] P. Ritchie. The Security Risks of AJAX/Web 2.0 Applications. Network Security. 2007.

[LL] RSnake. XSS (Cross Site Scripting) Cheat Sheet. ha.ckers.org.

[MM] Z. Su and G. Wassermann. The Essence of Command Injection Attacks in Web Applications. POPL 2006.

[NN] Roger Thompson. Alicia Keys MySpace page is hacked. Blog Post.

[OO] W3.org. Links in HTML documents. http://www.w3.org/TR/html401/struct/links.html

[PP] W3Schools. HTML 4.01 / XHTML 1.0 Reference

[QQ] W3Schools. HTML Head.

[RR] F. Washkush. MySpace fixes Alicia Keys' page after Cyberattack. SC Magazine. 2007.

[SS] D. Yu, A. Chander, N. Islam, I. Serikov. JavaScript Instrumentation for Browser Security. POPL 2007.