**How Viruses Work [4]**

* + Appended Viruses

A program virus attaches itself to a program; then, whenever the program is run, the virus is activated. This kind of attachment is usually easy to design and implement. In the simplest case, a virus inserts a copy of itself into the executable program file before the first executable instruction. Then, all the virus instructions execute first; after the last virus instruction, control flows naturally to what used to be the first program instruction. Such a situation is shown in [Figure 10.](#_bookmark6) This kind of attachment is simple and usually effective. The virus writer need not know anything about the program to which the virus will attach, and often the attached program simply serves as a carrier for the virus. The virus performs its task and then transfers to the original program. Typically, the user is unaware of the effect of the virus if the original program still does all that it used to. Most viruses attach in this manner.



**Figure 10: Virus Attachment**

* + Viruses that Surround a Program

An alternative to the attachment is a virus that runs the original program but has control before and after its execution. For example, a virus writer might want to prevent the virus from being detected. If the virus is stored on disk, its presence will be given away by its file name, or its size will affect the amount of space used on the disk. The virus writer might arrange for the virus to attach itself to the program that constructs the listing of files on the disk. If the virus regains control after the listing program has generated the listing but before the listing is displayed or printed, the virus could eliminate its entry from the listing and falsify space counts so that it appears not to exist. A surrounding virus is shown in [Figure 11.](#_bookmark7)

****

**Figure 11: Surrounding Virus**

* + **Integrated Viruses and Replacements**

A third situation occurs when the virus replaces some of its target, integrating itself into the original code of the target. Such a situation is shown in [Figure 12.](#_bookmark8) Clearly, the virus writer has to know the exact structure of the original program to know where to insert which pieces of the virus.



Figure 12:Virus Insertion

 **How Malicious Code (Virus) Gains Control**

To gain control of processing, malicious code such as a virus (V) has to be invoked instead of the target (T). Essentially, the virus either has to seem to be T, saying effectively“I am T,”or the virus has to push T out of the way and become a substitute for T, saying effectively “Call me instead of T.”A more blatant virus can simply say“invoke me [you fool].”

The virus can assume T name by replacing (or joining to) T's code in a file structure; this invocation technique is most appropriate for ordinary programs. The virus can overwrite T in storage (simply replacing the copy of T in storage, for example).

Alternatively, the virus can change the pointers in the file table so that the virus is located instead of T whenever T is accessed through the file system. These two cases are shown in [Figure 13](#_bookmark9).

The virus can supplant T by altering the sequence that would have invoked T to now invoke the virus V; this invocation can replace parts of the resident operating system by modifying pointers to those resident parts.



**Figure 13: Virus V Replacing Target T**

**Browser Attacks Types [4]**

\*Man-in-the-Browser

A man-in-the-browser attack is an example of malicious code that has infected a browser. Code inserted into the browser can read, copy, and redistribute anything the user enters in a browser. The threat here is that the attacker will intercept and reuse credentials to access financial accounts and other sensitive data.

***Man-in-browser: Trojan horse that intercepts data passing through the browser.***

In January 2008, security researchers led by Liam Omurchu of Symantec detected a new Trojan horse, *which they called SilentBanker*. This code linked to a victim’s browser as an add-on or browser helper object; in some versions it listed itself as a plug-in to display video. As a helper object, it set itself to intercept internal browser calls, including those to receive data from the keyboard, send data to a URL, generate or import a cryptographic key, read a file (including display that file on the screen), or connect to a site; this list includes pretty much everything a browser does. SilentBanker started with a list of over 400 URLs of popular banks throughout the world. Whenever it saw a user going to one of those sites, it redirected the user’s keystrokes through the Trojan horse and recorded customer details that it forwarded to remote computers (presumably controlled by the code's creators).

Banking and other financial transactions are ordinarily protected in transit by an encrypted session, using a protocol named *SSL or HTTPS*, and identified by a lock icon on the browser’s screen. This protocol means that the user’s communications are encrypted during transit. But remember that cryptography, although powerful, can protect only what it can control. Because SilentBanker was embedded within the browser, it intruded into the communication process as shown in [Figure 14.](#_bookmark10) When the user typed data, the operating system passed the characters to the browser. But before the browser could encrypt its data to transmit to the bank, SilentBanker intervened, acting as part of the browser. Notice that this timing vulnerability would not have been countered by any of the other security approaches banks use, such as an image that only the customer will recognize or two-factor authentication. Furthermore, the URL in the address bar looked and was authentic, because the browser actually did maintain a connection with the legitimate bank site.



Figure 14:SilentBanker Operates in the Middle of the Browser

**SSL encryption is applied in the browser; data are vulnerable before being encrypted.**

As if intercepting details such as name, account number, and authentication data were not enough, SilentBanker also changed the effect of customer actions. So, for example, if a customer instructed the bank to transfer money to an account at bank A, SilentBanker converted that request to make the transfer go to its own account at bank B, which the customer’s bank duly accepted as if it had come from the customer. When the bank returned its confirmation, SilentBanker changed the details before displaying them on the screen. Thus, the customer found out about the switch only after the funds failed to show up at bank A as expected.

\*Keystroke Logger

A *keystroke logger (or key logger)* is either hardware or software that records all keystrokes entered. The logger either retains these keystrokes for future use by the attacker or sends them to the attacker across a network connection.

As a hardware device, a keystroke logger is a small object that plugs into a USB port, resembling a plug-in wireless adapter or flash memory stick. Of course, to compromise a computer you have to have physical access to install (and later retrieve) the device. You also need to conceal the device so the user will not notice the logger (for example, installing it on the back of a desktop machine). In software, the logger is just a program installed like any malicious code. Such devices can capture passwords, login identities, and all other data typed on the keyboard. Although not limited to browser interactions, a keystroke logger could certainly record all keyboard input to the browser.

\*Page-in-the-Middle

A *page-in-the-middle attack is another type of browser attack in which a user is redirected to another page.* Similar to the man-in-the-browser attack, a page attack might wait until a user has gone to a particular web site and present a fictitious page for the user. As an example, when the user clicks “login” to go to the login page of any site, the attack might redirect the user to the attackers page, where the attacker can also capture the user's credentials.

\*Program Download Substitution

Coupled with a page-in-the-middle attack is a download substitution. In a **download substitution**, the attacker presents a page with a desirable and seemingly innocuous program for the user to download, for example, a browser toolbar or a photo organizer utility. What the user does not know is that instead of or in addition to the intended program, the attacker downloads and installs malicious code. *A user agreeing to install a program has no way to know what that program will actually do*. The advantage for the attacker of a program download substitution is that users have been conditioned to be wary of program downloads, precisely for fear of downloading malicious code. In this attack, the user knows of and agrees to a download, not realizing what code is actually being installed. (Then again, users seldom know what really installs after they click [Yes].) This attack also defeats users’ access controls that would normally block software downloads and installations, because the user intentionally accepts this software.

Web Attacks Targeting Users [4]

We next consider two classes of situations involving web content. The first kind involves false content, most likely because the content was modified by someone unauthorized; with these the intent is to mislead the viewer. The second, more dangerous, kind seeks to harm the viewer.

* 1. False or Misleading Content

The case of computer artifacts is similar. An incoherent message, a web page riddled with grammatical errors, or a peculiar political position can all alert you that something is suspicious, but a well-crafted forgery may pass without question. The falsehoods that follow include both obvious and subtle forgeries.

* + 1. ***Defaced Web Site****:*The simplest attack, a **website defacement**, occurs when an attacker replaces or modifies the content of a legitimate web site. For example, in January 2010, BBC reported that the web site of the incoming president of the European Union was defaced to present a picture of British comic actor Rowan Atkinson (Mr. Bean) instead of the president. The objectives of website defacements also vary. Sometimes the goal is just to prove a point or embarrass the victim. Some attackers seek to make a political or ideological statement, whereas others seek only attention or respect. In some cases the attacker is showing a point, proving that it was possible to defeat integrity.
		2. ***Fake Web Site:*** A similar attack involves a fake web site. In [Figure 15](#_bookmark11) we show a fake version of the web site of Barclays Bank (England) at [http://www.gb-bclayuk.com/.](http://www.gb-bclayuk.com/) The real Barclays site is at [http://group.barclays.com/Home.](http://group.barclays.com/Home) As you can see, the forger had some trouble with the top image, but if that were fixed, the remainder of the site would look convincing. Web sites are easy to fake because the attacker can obtain copies of the images the real site uses to generate its web site. All the attacker has to do is change the values of links to redirect the unsuspecting victim to points of the attacker’s choosing.



**Figure 15: Fake Web Site for Barclays Bank**

* + 1. ***Fake Code***: its sources, effects, and countermeasures. We described how opening a document or clicking a link can lead to a surreptitious download of code that does nothing obvious but installs a hidden infection. One transmission route we did not note was an explicit download: programs intentionally installed that may advertise one purpose but do something entirely different. [Figure 16](#_bookmark12) shows a seemingly authentic ad for a replacement or update to the popular Adobe Reader. The link from which it came (www.pdf-new-2010-download.com) was redirected from www.adobe-download-center.com; both addresses seem like the kinds of URLs Adobe might use to distribute legitimate software.

**Figure 16: Advertisement of Fake Software**

Whether this attack is meant just to deceive or to harm depends on what code is actually delivered. This example shows how malicious software can masquerade as legitimate. The charade can continue unnoticed for some time if the malware at least seems to implement its ostensible function, in this case, displaying and creating PDF documents. Perhaps the

easiest way for a malicious code writer to install code on a target machine is to create an application that a user willingly downloads and installs.

* 1. Malicious Web Content
* **Web Bug**: You probably know that a web page is made up of many files: some text, graphics, executable code, and scripts. When the web page is loaded, files are downloaded from a destination and processed; during the processing they may invoke other files (perhaps from other sites) which are in turn downloaded and processed, until all invocations have been satisfied. When a remote file is fetched for inclusion, the request also sends the IP address of the requester, the type of browser, and the content of any cookies stored for the requested site. These cookies permit the page to display a notice such as “Welcome back, Elaine,” bring up content from your last visit, or redirect you to a particular web page. A web bug, also called a clear GIF, 1x1 GIF, or tracking bug, is a tiny image, as small as 1 pixel by 1 pixel (depending on resolution, screens display at least 100 to 200 pixels per inch), an image so small it will not normally be seen. Nevertheless, it is loaded and processed the same as a larger picture. Part of the processing is to notify the bug’s owner, the advertiser, who thus learns that another user has loaded the advertising image.
* **Clickjacking:** Suppose you are at a gasoline filling station with three buttons to press to select the grade of fuel you want. The station owner, noticing that most people buy the lowest-priced fuel but that his greatest profit comes from the highest-priced product, decides to pull a trick. He pastes stickers over the buttons for the lowest and highest prices saying, respectively, ―high performance‖ (on the lowest-priced button) and ―economy‖ (on the expensive, high-profit button). Thus, some people will inadvertently push the economy/high-priced button and unwittingly generate a higher profit. Unfair and deceptive, yes, but if the owner is unscrupulous, the technique would work; however, most businesses would not try that, because it is unethical and might lose customers. But computer attackers do not care about ethics or loss of customers, so a version of this technique becomes a computer attack. As shown in Figure 4-13, the computer attack uses an image pasted over, that is, displayed on top of, another image. We are all familiar with the click box “ Do you want to delete this file? [Yes] [No].” *Clickjacking* is a technique that essentially causes that prompt box to slide around so that [Yes] is always under the mouse. The attacker also makes this box transparent, so the victim is unaware of clicking anything. Furthermore, a second, visible image is pasted underneath, so the victim thinks the box being clicked is something like “For a free prize, click [Here].” The victim clicks where [Here] is on the screen, but [Here] is not a button at all; it is just a picture directly under [Yes] (which is invisible). The mouse click selects the [Yes] button.



**Figure 17: Clickjacking**

***Example:*** *Sidebar 4-7 Facebook Clickjack Attack*

*In Summer 2010, thousands of Facebook users were tricked into posting that they “liked” a particular site. According to BBC news (3 June 2010), victims were presented with sites that many of their friends had “liked,” such as a video of the World Cup tennis match. When the users clicked to see the site, they were presented with another message asking them to click to confirm they were over age 18. What the victims did not see was that the confirmation box was a sham underneath an invisible box asking them to confirm they “liked” the target web site. When the victims clicked that they were over 18, they were really confirming their “like” of the video. This attack seems to have had no malicious impact, other than driving up the “like” figures on certain benign web sites. You can readily imagine serious harm from this kind of attack, however.*

* **Drive-By Download:** Similar to the clickjacking attack, a drive-by download is an attack in which code is downloaded, installed, and executed on a computer without the user’s permission and usually without the user’s knowledge. In one example of a drive-by download, in April 2011, a web page from the U.S. Postal Service was compromised with the Blackhole commercial malicious-exploit kit. Clicking a link on the postal service web site redirected the user to a web site in Russia, which presented what looked like a familiar ―Error 404—Page Not Found‖ message, but instead the Russian site installed malicious code carefully matched to the user’s browser and operating system type (*eWeek*, 10 April 2011).

14 Email Attacks

1. **Fake Email:** Given the huge amount of email sent and received daily, it is not surprising that much of it is not legitimate. Some frauds are easy to spot, as our first example shows, but some illegitimate email can fool professionals, as in our second example. A recent email message advised me that my Facebook account had been deactivated, shown in Figure 4-15. The only problem is, I have no Facebook account. In the figure I have shown where some of the links and buttons actually lead, instead of the addresses shown; the underlying addresses certainly do not look like places Facebook would host code.



**Figure 18: Fake Email**

1. **Fake Email Messages as Spam:** Similarly, an attacker can attempt to fool people with fake email messages. Probably everyone is familiar with **spam**, fictitious or misleading email, offers to buy designer watches, anatomical enhancers, or hot stocks, as well as get-rich schemes involving money in overseas bank accounts. Similar false messages try to get people to click to download a browser enhancement or even just click for more detail. Spammers now use more realistic topics for false messages to entice recipients to follow a malicious link.
2. **Fake (Inaccurate) Email Header Data:** As we just described, one reason email attacks succeed is that the headers on email are easy to spoof, and thus recipients believe the email has come from a safe source. Here we consider precisely how the spoofing occurs and what could be done. Control of email headers is up to the sending mail agent. The header form is standardized, but within the Internet email network as a message is forwarded to its destination, each receiving node trusts the sending node to deliver accurate content. However, a malicious, or even faulty, email transfer agent may send messages with inaccurate headers, specifically in the “from” fields.
3. **Phishing:** One type of fake email that has become prevalent enough to warrant its own name is phishing (pronounced like “fishing”). In a phishing attack, the email message tries to trick the recipient into disclosing private data or taking another unsafe action. Phishing email messages purport to be from reliable companies such as banks or other financial institutions, popular web site companies (such as Facebook, Hotmail, or Yahoo), or consumer products companies.