Web Spoofing
An Internet Con Game

Edward W. Felten, Dirk Balfanz, Drew Dean, and Dan S. Wallach

Technical Report 540-96

Department of Computer Science, Princeton University

Graphics by Markus Hübner

Introduction

This paper describes an Internet security attack that could endanger the privacy of World Wide Web users and the integrity of their data. The attack can be carried out on today's systems, endangering users of the most common Web browsers, including Netscape Navigator and Microsoft Internet Explorer.

Web spoofing allows an attacker to create a "shadow copy" of the entire World Wide Web. Accesses to the shadow Web are funneled through the attacker's machine, allowing the attacker to monitor the all of the victim's activities including any passwords or account numbers the victim enters. The attacker can also cause false or misleading data to be sent to Web servers in the victim's name, or to the victim in the name of any Web server. In short, the attacker observes and controls everything the victim does on the Web.

We have implemented a demonstration version of this attack.

Spoofing Attacks

In a *spoofing attack*, the attacker creates misleading context in order to trick the victim into making an inappropriate security-relevant decision. A spoofing attack is like a con game: the attacker sets up a false but convincing world around the victim. The victim does something that would be appropriate if the false world were real. Unfortunately, activities that seem reasonable in the false world may have disastrous effects in the real world.

Spoofing attacks are possible in the physical world as well as the electronic one. For example, there have been several incidents in which criminals set up bogus automated-teller machines, typically in the public areas of shopping malls [1]. The machines would accept ATM cards and ask the person to enter their PIN code. Once the machine had the victim's PIN, it could either eat the card or "malfunction" and return the card. In either case, the criminals had enough information to copy the victim's card and use the duplicate. In these attacks, people were fooled by the context they saw: the location of the machines, their size and weight, the way they were decorated, and the appearance of their electronic displays.

People using computer systems often make security-relevant decisions based on contextual cues they see. For example, you might decide to type in your bank account number because you believe you are visiting your bank's Web page. This belief might arise because the page has a familiar look, because the bank's URL appears in the browser's location line, or for some other reason.

To appreciate the range and severity of possible spoofing attacks, we must look more deeply into two parts of the definition of spoofing: security-relevant decisions and context.

Security-relevant Decisions

By "security-relevant decision," we mean any decision a person makes that might lead to undesirable results such as a breach of privacy or unauthorized tampering with data. Deciding to divulge sensitive information, for example by typing in a password or account number, is one example of a security-relevant decision. Choosing to accept a downloaded document is a security-relevant decision, since in many cases a downloaded document is capable of containing malicious elements that harm the person receiving the document [2].

Even the decision to accept the accuracy of information displayed by your computer can be security-relevant. For example, if you decide to buy a stock based on information you get from an online stock ticker, you are trusting that the information provided by the ticker is correct. If somebody could present you with incorrect stock prices, they might cause you to engage in a transaction that you would not have otherwise made, and this could cost you money.

Context

A browser presents many types of context that users might rely on to make decisions. The text and pictures on a Web page might give some impression about where the page came from; for example, the presence of a corporate logo implies that the page originated at a certain corporation.

The appearance of an object might convey a certain impression; for example, neon green text on a purple background probably came from *Wired* magazine. You might think you're dealing with a popup window when what you are seeing is really just a rectangle with a border and a color different from the surrounding parts of the screen. Particular graphical items like file-open dialog boxes are immediately recognized as having a certain purpose. Experienced Web users react to such cues in the same way that experienced drivers react to stop signs without reading them.

The names of objects can convey context. People often deduce what is in a file by its name. Is `manual.doc` the text of a user manual? (It might be another kind of document, or it might not be a document at all.) URLs are another example. Is `MICR0S0FT.COM` the address of a large software company? (For a while that address pointed to someone else entirely. By the way, the round symbols in `MICR0S0FT` here are the number zero, not the letter O.) Was `dole96.org` Bob Dole's 1996 presidential campaign? (It was not; it pointed to a parody site.)

People often get context from the timing of events. If two things happen at the same time, you naturally think they are related. If you click over to your bank's page and a username/password dialog box appears, you naturally assume that you should type the name and password that you use for the bank. If you click on a link and a document immediately starts downloading, you assume that the document came from the site whose link you clicked on. Either assumption could be wrong.

If you only see one browser window when an event occurs, you might not realize that the event was caused by another window hiding behind the visible one.

Modern user-interface designers spend their time trying to devise contextual cues that will guide people to behave appropriately, even if they do not explicitly notice the cues. While this is usually beneficial, it can become dangerous when people are accustomed to relying on context that is not always correct.

TCP and DNS Spoofing

Another class of spoofing attack, which we will not discuss here, tricks the user's software into an inappropriate action by presenting misleading information to that software [3]. Examples of such attacks include TCP spoofing [4], in which Internet packets are sent with forged return addresses, and DNS spoofing [5], in which the attacker forges information about which machine names correspond to which network addresses. These other spoofing attacks are well known, so we will not discuss them further.

Web Spoofing

Web spoofing is a kind of electronic con game in which the attacker creates a convincing but false copy of the entire World Wide Web. The false Web looks just like the real one: it has all the same pages and links. However, the attacker controls the false Web, so that all network traffic between the victim's browser and the Web goes through the attacker.

Consequences

Since the attacker can observe or modify any data going from the victim to Web servers, as well as controlling all return traffic from Web servers to the victim, the attacker has many possibilities. These include surveillance and tampering.

Surveillance The attacker can passively watch the traffic, recording which pages the victim visits and the contents of those pages. When the victim fills out a form, the entered data is transmitted to a Web server, so the attacker can record that too, along with the response sent back by the server. Since most on-line commerce is done via forms, this means the attacker can observe *any account numbers or passwords the victim enters*.

As we will see below, the attacker can carry out surveillance even if the victim has a "secure" connection (usually via Secure Sockets Layer) to the server, that is, even if the victim's browser shows the secure-connection icon (usually an image of a lock or a key).

Tampering The attacker is also free to modify any of the data traveling in either direction between the victim and the Web. The attacker can modify form data submitted by the victim. For example, if the victim is ordering a product on-line, the attacker can change the product number, the quantity, or the ship-to address.

The attacker can also modify the data returned by a Web server, for example by inserting misleading or offensive material in order to trick the victim or to cause antagonism between the victim and the server.

Spoofing the Whole Web

You may think it is difficult for the attacker to spoof the entire World Wide Web, but it is not. The attacker need not store the entire contents of the Web. The whole Web is available on-line; the attacker's server can just fetch a page from the real Web when it needs to provide a copy of the page on the false Web.

How the Attack Works

The key to this attack is for the attacker's Web server to sit between the victim and the rest of the Web. This kind of arrangement is called a "man in the middle attack" in the security literature.

URL Rewriting

The attacker's first trick is to rewrite all of the URLs on some Web page so that they point to the attacker's server rather than to some real server. Assuming the attacker's server is on the machine `www.attacker.org`, the attacker rewrites a URL by adding http://www.attacker.org to the front of the URL. For example, `http://home.netscape.com` becomes `http://www.attacker.org/http://home.netscape.com`. (The URL rewriting technique has been used for other reasons by two other Web sites, the Anonymizer and the Zippy filter. See page 9 for details.)

Figure 1 shows what happens when the victim requests a page through one of the rewritten URLs. The victim's browser requests the page from `www.attacker.org`, since the URL starts with `http://www.attacker.org`. The remainder of the URL tells the attacker's server where on the Web to go to get the real document.

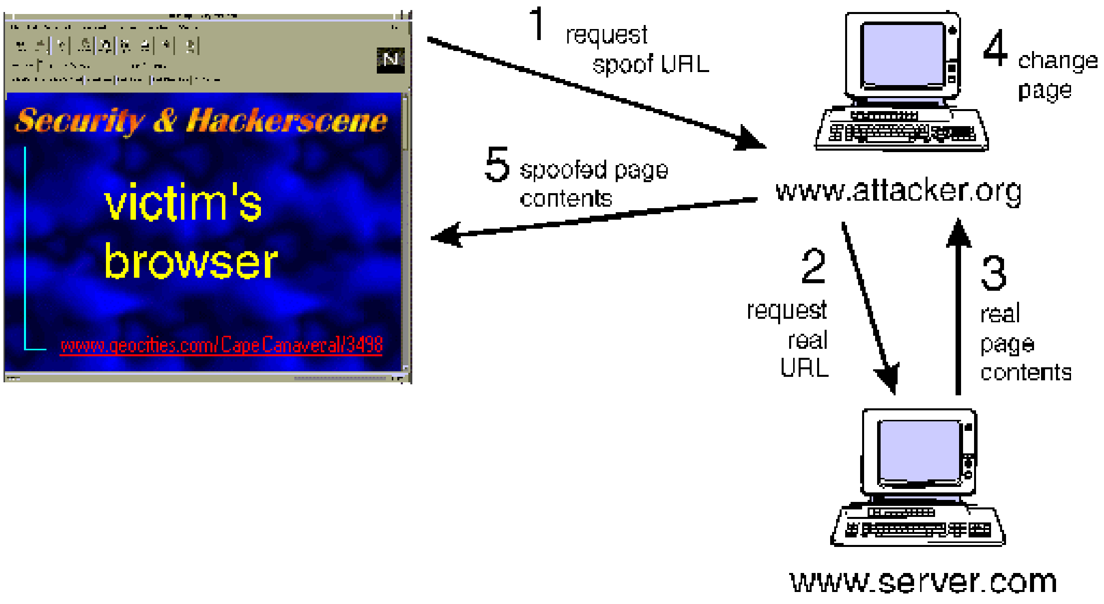

Figure 1: An example Web transaction during a Web spoofing attack. The victim requests a Web page. The following steps occur: (1) the victim's browser requests the page from the attacker's server; (2) the attacker's server requests the page from the real server; (3) the real server provides the page to the attacker's server; (4) the attacker's server rewrites the page; (5) the attacker's server provides the rewritten version to the victim.

Once the attacker's server has fetched the real document needed to satisfy the request, the attacker rewrites all of the URLs in the document into the same special form by splicing `http://www.attacker.org/` onto the front. Then the attacker's server provides the rewritten page to the victim's browser.

Since all of the URLs in the rewritten page now point to `www.attacker.org`, if the victim follows a link on the new page, the page will again be fetched through the attacker's server. The victim remains trapped in the attacker's false Web, and can follow links forever without leaving it.

Forms

If the victim fills out a form on a page in a false Web, the result appears to be handled properly. Spoofing of forms works naturally because forms are integrated closely into the basic Web protocols: form submissions are encoded in URLs and the replies are ordinary HTML Since any URL can be spoofed, forms can also be spoofed.

When the victim submits a form, the submitted data goes to the attacker's server. The attacker's server can observe and even modify the submitted data, doing whatever malicious editing desired, before passing it on to the real server. The attacker's server can also modify the data returned in response to the form submission.

"Secure" connections don't help

One distressing property of this attack is that it works even when the victim requests a page via a "secure" connection. If the victim does a "secure" Web access (a Web access using the Secure Sockets Layer) in a false Web, everything will appear normal: the page will be delivered, and the secure connection indicator (usually an image of a lock or key) will be turned on.

The victim's browser says it has a secure connection because it *does* have one. Unfortunately the secure connection is to `www.attacker.org` and not to the place the victim thinks it is. The victim's browser thinks everything is fine: it was told to access a URL at `www.attacker.org` so it made a secure connection to `www.attacker.org`. The secure-connection indicator only gives the victim a false sense of security.

Starting the Attack

To start an attack, the attacker must somehow lure the victim into the attacker's false Web. There are several ways to do this. An attacker could put a link to a false Web onto a popular Web page. If the victim is using Web-enabled email, the attacker could email the victim a pointer to a false Web, or even the contents of a page in a false Web. Finally, the attacker could trick a Web search engine into indexing part of a false Web.

Completing the Illusion

The attack as described thus far is fairly effective, but it is not perfect. There is still some remaining context that can give the victim clues that the attack is going on. However, it is possible for the attacker to eliminate virtually all of the remaining clues of the attack's existence.

Such evidence is not too hard to eliminate because browsers are very customizable. The ability of a Web page to control browser behavior is often desirable, but when the page is hostile it can be dangerous.

The Status Line

The status line is a single line of text at the bottom of the browser window that displays various messages, typically about the status of pending Web transfers.

The attack as described so far leaves two kinds of evidence on the status line. First, when the mouse is held over a Web link, the status line displays the URL the link points to. Thus, the victim might notice that a URL has been rewritten. Second, when a page is being fetched, the status line briefly displays the name of the server being contacted. Thus, the victim might notice that `www.attacker.org` is displayed when some other name was expected.

The attacker can cover up both of these cues by adding a JavaScript program to every rewritten page. Since JavaScript programs can write to the status line, and since it is possible to bind JavaScript actions to the relevant events, the attacker can arrange things so that the status line participates in the con game, always showing the victim what would have been on the status line in the real Web. Thus the spoofed context becomes even more convincing.

The Location Line

The browser's location line displays the URL of the page currently being shown. The victim can also type a URL into the location line, sending the browser to that URL. The attack as described so far causes a rewritten URL to appear in the location line, giving the victim a possible indication that an attack is in progress.

This clue can be hidden using JavaScript. A JavaScript program can hide the real location line and replace it by a fake location line which looks right and is in the expected place. The fake location line can show the URL the victim expects to see. The fake location line can also accept keyboard input, allowing the victim to type in URLs normally. Typed-in URLs can be rewritten by the JavaScript program before being accessed.

Viewing the Document Source

There is one clue that the attacker cannot eliminate, but it is very unlikely to be noticed.

By using the browser's "view source" feature, the victim can look at the HTML source for the currently displayed page. By looking for rewritten URLs in the HTML source, the victim can spot the attack. Unfortunately, HTML source is hard for novice users to read, and very few Web surfers bother to look at the HTML source for documents they are visiting, so this provides very little protection.

A related clue is available if the victim chooses the browser's "view document information" menu item. This will display information including the document's real URL, possibly allowing the victim to notice the attack. As above, this option is almost never used so it is very unlikely that it will provide much protection.

Bookmarks

There are several ways the victim might accidentally leave the attacker's false Web during the attack. Accessing a bookmark or jumping to a URL by using the browser's "Open location" menu item might lead the victim back into the real Web. The victim might then reenter the false Web by clicking the "Back" button. We can imagine that the victim might wander in and out of one or more false Webs. Of course, bookmarks can also work against the victim, since it is possible to bookmark a page in a false Web. Jumping to such a bookmark would lead the victim into a false Web again.

Tracing the Attacker

Some people have suggested that this attack can be deterred by finding and punishing the attacker. It is true that the attacker's server must reveal its location in order to carry out the attack, and that evidence of that location will almost certainly be available after an attack is detected.

Unfortunately, this will not help much in practice because attackers will break into the machine of some innocent person and launch the attack there. Stolen machines will be used in these attacks for the same reason most bank robbers make their getaways in stolen cars.

Remedies

Web spoofing is a dangerous and nearly undetectable security attack that can be carried out on today's Internet. Fortunately there are some protective measures you can take.

Short-term Solution

In the short run, the best defense is to follow a three-part strategy:

1. disable JavaScript in your browser so the attacker will be unable to hide the evidence of the attack;
2. make sure your browser's location line is always visible;
3. pay attention to the URLs displayed on your browser's location line, making sure they always point to the server you think you're connected to.

This strategy will significantly lower the risk of attack, though you could still be victimized if you are not conscientious about watching the location line.

At present, JavaScript, ActiveX, and Java all tend to facilitate spoofing and other security attacks, so we recommend that you disable them. Doing so will cause you to lose some useful functionality, but you can recoup much of this loss by selectively turning on these features when you visit a trusted site that requires them.

Long-term Solution

We do not know of a fully satisfactory long-term solution to this problem.

Changing browsers so they always display the location line would help, although users would still have to be vigilant and know how to recognize rewritten URLs.

For pages that are not fetched via a secure connection, there is not much more that can be done.

For pages fetched via a secure connection, an improved secure-connection indicator could help. Rather than simply indicating a secure connection, browsers should clearly say who is at the other end of the connection. This information should be displayed in plain language, in a manner intelligible to novice users; it should say something like "Microsoft Inc." rather than "www.microsoft.com."

Every approach to this problem seems to rely on the vigilance of Web users. Whether we can realistically expect everyone to be vigilant all of the time is debatable.

Related Work

We did not invent the URL rewriting technique. Previously, URL rewriting has been used as a technique for providing useful services to people who have asked for them.

We know of two existing services that use URL rewriting. The Anonymizer, written by Justin Boyan at Carnegie Mellon University, is a service that allows users to surf the Web without revealing their identities to the sites they visit. The Zippy filter, written by Henry Minsky, presents an amusing vision of the Web with Zippy-the-Pinhead sayings inserted at random.

Though we did not invent URL rewriting, we believe we are the first to realize its full potential as one component of a security attack.

Acknowledgments

The URL-rewriting part of our demonstration program is based on Henry Minsky's code for the Zippy filter. We are grateful to David Hopwood for useful discussions about spoofing attacks, and to Gary McGraw and Laura Felten for comments on drafts of this paper. The figure was designed by Gary McGraw.

For More Information

More information is available from our Web page at http://www.cs.princeton.edu/sip, or from Prof. Edward Felten at felten@cs.princeton.edu or (609) 258-5906.

References

[1] Peter G. Neumann. Computer-Related Risks. ACM Press, New York, 1995.

[2] Gary McGraw and Edward W. Felten. Java Security: Hostile Applets, Holes and Antidotes. John Wiley and Sons, New York, 1996.

[3] Robert T. Morris. A Weakness in the 4.2BSD UNIX TCP/IP Software. Computing Science Technical Report 117, AT&T Bell Laboratories, February 1985.

[4] Steven M. Bellovin. Security Problems in the TCP/IP Protocol Suite. Computer Communications Review 19(2):32-48, April 1989.

[5] Steven M. Bellovin. Using the Domain Name System for System Break-ins. Proceedings of Fifth Usenix UNIX Security Symposium, June 1995.

[6] Web site at http://www.anonymizer.com

[7] Web site at http://www.metahtml.com/apps/zippy/welcome.html

Back to the Security-Page