As with marriage, SSL security success is in the details

ATTACKS AGAINST SSL

Something old, something new, something borrowed, and something blue. **BY KURT SEIFRIED**

The year 2009 has been very interesting for SSL security. Several new and practical attacks were publicized, and fortunately, most were fixed within a relatively short period of time. The year began with an effective attack against MD5-based SSL certificates [1], technologically a very sophisticated attack and not one we're likely to see in the wild now that MD5 certificates are being phased out. Then, at BlackHat 2009 in Las Vegas, Moxie Marlinspike talked about several attacks against SSL, including a very old issue that has recently become a problem.

Something Old: SSL Strip

This attack is fairly simple and imaginative. Because the majority of users go to non-SSL-secured websites before being redirected to SSL-encrypted websites (e.g., a payment processor, account login server, etc.), attackers can easily prevent them from using encryption and thus snoop all their traffic. In the past, an attacker might have tried to execute a man-in-the-middle attack by sniffing unencrypted traffic or using a self-signed certificate to pretend to be a legitimate site, but these are exactly the things that SSL was designed to prevent.

But what if the victim never makes it to the secure website? Many sites (my bank, probably your bank, most major online retailers, etc.) have non-SSL-protected front ends where you do your shopping or access the login page to get to your account. By rewriting all links in the web pages that point to SSL-encrypted websites to point to non-SSL-encrypted sites, an attacker can view and rewrite content without alerting the user, unless of course they are paranoid enough to notice that the lock icon and so forth are missing. The SSL strip program automates all of this, including an ARP spoofer (so anyone on your local network or sharing the same wireless point as you) to redirect traffic to your system [2].

Installing SSL Strip

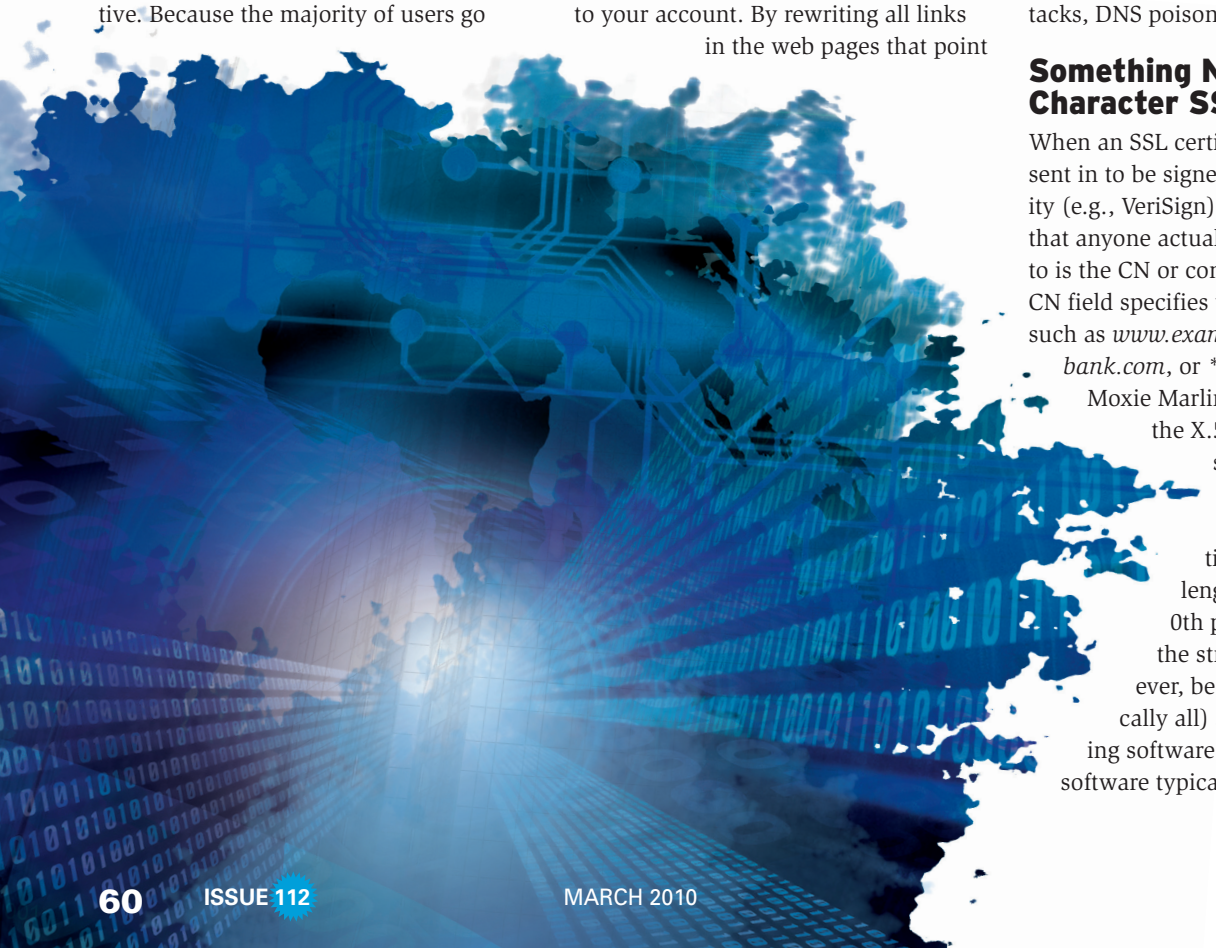
SSL strip is a Python application that uses the Twisted [3] framework. So, all you need to do is install twisted, download and unpack the SSL strip tarball and optionally install it (if you don't install it, you can simply run it from the local directory you unpacked it to):

```
yum install twisted-web

apt-get install twisted-web

python setup.py install
```

Then, you simply turn on IP forwarding, add an iptables route to redirect HTTP traffic, and run the *sslstrip* Python program. To redirect local machines to your system, you can use the *arpspoof* program that is included with the dsniff program [4] or other tricks (e.g., DHCP attacks, DNS poisoning, etc.).

Something New: NULL Character SSL Certificates

When an SSL certificate is created and sent in to be signed by a signing authority (e.g., VeriSign), about the only field that anyone actually pays any attention to is the CN or common name field. The CN field specifies the name of the server, such as *www.example.org*, *www.bigbank.com*, or **.somecompany.com*. Moxie Marlinspike discovered that the X.509 and SSL certificate standards specify the CN string as a PASCAL string; so, essentially, you declare the length of the string at the 0th position and then add the string data after it. However, because most (well, basically all) SSL certificate processing software is written in C, the software typically handles the string



as a C string (which means it is NULL terminated (\0) at the end to indicate where it stops). Most programmers did not realize the implication of this and simply read the CN string into a C string structure and merrily went on their way.

The problem arises, however, when someone (who legitimately owns *example.org*, for example) gets a certificate for *www.bigbank.com\0www.example.org*. Back in the early days of SSL certificates, when they were expensive, humans actually looked at these requests, which not only included certificate requests but also often included business records and other forms of proof to show you had ownership of the domain and the right to use it. (Indeed, when I first got a certificate for seifried.org, because it was non-commercial, I had to send in scans of my passport to prove I had a claim to "seifried.") Thus, any malformed or strange-looking request was likely to get caught and not processed.

Times have changed, and now you can get SSL certificates in a few minutes using a completely automated process (which most likely does a WHOIS lookup on your domain and then emails the admin or technical contact). Thus, you could essentially apply for a certificate that looks like it belongs within your domain (i.e., *example.org*). However, when the application is processed by a browser, because it handles the CN as a C string, it will read the first part of www.bigbank.com, encounter the NULL terminator, and then drop the example.org part (allowing you to spoof www.bigbank.com with ease).

The good news is that fixing this is relatively easy; the certificate authorities simply revoked any certificates with a NULL character in them (which should never occur) and implemented filters to prevent it from happening again. On the client side, most browsers and SSL-capable clients and servers (e.g., mutt, PostgreSQL, fetchmail, Opera, etc.) have been updated to fix this as well. For more information on this attack and extensions of it, please see Moxie Marlinspike's BlackHat presentation (in PDF format) [5].

Something Borrowed: Your Name

We've all accidentally gone to a site like *reddit.org* (*reddit.com* is the real one), *icanhasacheezburger.com* (*icanhascheezburger.com*), or some other domain squatter. In the past, things were relatively simple: If an attacker wanted to register a domain that imitated a legitimate one, he or she would either drop or add letters or swap the number one and a lowercase L, for example. Defending yourself against it was kind of a pain, but not impossible; you simply had to register a lot of extra domain names with common typos, the lowercase L (l) and the one (1) swapped, and so on. Fortunately, the character set that allowed valid domain names was limited to several dozen characters.

However this has changed with the advent of International Domain Names. Now several dozen characters look virtually identical to Roman letters, such as the Cyrillic letters Es (с), Shha (h), Ye (е), Je (j), On (о), Er (р), Dze (ѕ), Kha (х), and U (у) or the Greek omicron (ο) or nu (ν). Because you have no easy way to verify the domain names you are looking at, it is hoped that browsers will start giving visual queues about the makeup of domain names, perhaps making the text a different color if it is outside your country, for example [6].

Something Blue: SSL Renegotiation

I'm running out of space, so this topic will be short and sweet. Basically, back in 1990 when the SSL and TLS specifications were written, they were slightly over-engineered to allow behaviors (like renegotiation) that turned out not to be needed or wanted by most people. By exploiting this renegotiation behavior, attackers can insert content that allows them to execute a new class of CSRF (Cross-Site Request Forgery) attacks. But not to worry, right? I mean most modern applications have strong CSRF protections, such as one-time tokens that change for each transaction preventing the insertion of a false transaction [7].

Unfortunately, some websites allow certain behaviors that can result in problems. The first real-world attack example using this issue was against Twitter. The twitter API (since fixed) allowed an attacker to insert new HTTP request headers into the request. With this, an attacker could move the content of the original request sent by the victim, such as a cookie, to be sent as an HTTP POST. This in turn resulted in Twitter taking the HTTP POST data (the user's cookie) and posting it as a public tweet [8].

Fortunately, the solution to this problem is really easy: Many software vendors are simply disabling SSL renegotiation within their software. No renegotiation, no attack.

Conclusion

It's getting messy out there. Fortunately, security researchers are getting pretty good at reading the specification documents, comparing the systems that have actually been built, and finding the resulting security flaws. Perhaps someday the people writing the specifications and software will get better at playing nicely with each other. ■

INFO

[1] "Broken Chain of Trust" by Kurt Seifried, *Linux Pro Magazine*, March 2009, p. 64, *http://www.linuxpromagazine.com/Issues/2009/100/BROKEN-CHAIN-OF-TRUST*

[2] SSL Strip: *http://www.thoughtcrime.org/software/sslstrip/*

[3] Twisted: *http://twistedmatrix.com/trac/*

[4] dsniff: *http://www.monkey.org/~dugsong/dsniff/*

[5] More Tricks For Defeating SSL In Practice: *http://www.blackhat.com/presentations/bh-usa-09/MARLINSPIKE/BHUSA09-Marlinspike-DefeatSSL-SLIDES.pdf*

[6] IDN Homograph Attack: *http://en.wikipedia.org/wiki/IDN_homograph_attack*

[7] "Attack of the CSRF" by Kurt Seifried, *Linux Pro Magazine*, February 2009, p. 66, *http://www.linuxpromagazine.com/Issues/2009/99/ATTACK-OF-THE-CSRF*

[8] TLS Renegotiation Vulnerability (CVE-2009-3555): *http://www.securegoose.org/2009/11/tls-renegotiation-vulnerability-cve.html*

THE AUTHOR

Kurt Seifried is an Information Security Consultant specializing in Linux and networks since 1996. He often wonders how it is that technology works on a large scale but often fails on a small scale.