After Taking a Bite Out of SSL 3.0, This 'POODLE' Needs Some Time in Obedience Class

15 OCT

After Taking a Bite Out of SSL 3.0, This 'POODLE' Needs Some Time in Obedience Class

October 15, 2014 / Ryan Nolette / Advanced Threat Protection, Detection and Response, Endpoint and Server Security, Tech Toolbox

POODLE (Padding Oracle On Downgraded Legacy Encryption) is an attack on the SSLv3 protocol that allows the plaintext of secure connections to be calculated by a network attacker.

This POODLE bites and has been all the rage today. Let's take a deeper look.

TLS (Transport Layer Security) and SSL (Secure Sockets Layer) are protocols that provide data encryption and authentication between clients and servers when that data is being sent across a network. An example is checking your email or logging into your social media account. The terms "SSL" and "TLS" are regularly used interchangeably or collectively as "TLS/SSL," but one, "SSL," is, in effect, the forerunner of the other, "TLS." "SSL 3.0" served as the basis for "TLS 1.0," which, consequently, is occasionally referred to as "SSL 3.1."

This vulnerability enables a network attacker to recover the plaintext communications of a victim. This means that an attacker, by exploiting this vulnerability, can gain access to a user's SSL session and recover things such as passwords and cookies. Possessing such information allows an attacker to access a user's private account data on a website (webmail or other online accounts that are signed into over https).

POODLE is similar to previously known attacks against SSL/TLS. The best publicized were BEAST (Browser Exploit Against SSL/TLS) and CRIME (Compression Ratio Info-leak Made Easy).

Why is this such an Issue?

The vulnerable protocol, SSL 3.0, is nearly 15 years old and nearly all browsers support it. If a newer, more secure protocol (e.g., TLSv1.2) fails to make the connection, the browser will attempt to connect on the older protocols – including SSL 3.0.

This vulnerability demonstrates that legacy support is the real threat here.

The problem occurs when old versions of protocols are left running to support people who choose not to upgrade. (I'm not talking about those people who decide to apply a patch a month or two after release. Rather, I'm referring to those who don't patch/update for years, if ever.) To support these people, vendors keep outdated and vulnerable code running.

Currently, a non-zero number of servers supporting SSLv3 still exist. This means an attacker can trick a browser to use SSLv3 even when both the browser of the victim and the server they are trying to access support a more recent version of SSL.

This makes the attack widely applicable, which is why this vulnerability is such an issue.

How do I Remediate this Issue?

Currently, there is no officially released client-side fix for this vulnerability.

Don’t panic. Sip your coffee and keep reading.

Some browsers offer the ability to specify which versions of SSL their browser can use. If your browser supports this feature, you can disable the ability to use SSLv3 and mitigate the threat until a fix is released. You can also use TLS_FALLBACK_SCSV, the Transport Layer Security Signaling Cipher Suite Value that blocks protocol downgrades.

On the server side, however, the current suggested mitigation (Posted by Bodo Möller, Google Security Team) is:

“Disabling SSL 3.0 support, or CBC-mode ciphers with SSL 3.0, is sufficient to mitigate this issue, but presents significant compatibility problems, even today. Therefore our recommended response is to support TLS_FALLBACK_SCSV. This is a mechanism that solves the problems caused by retrying failed connections and thus prevents attackers from inducing browsers to use SSL 3.0. It also prevents downgrades from TLS 1.2 to 1.1 or 1.0 and so may help prevent future attacks.”

Both Google and Mozilla have published statements confirming the pending removal of support for SSLv3 from the next release of their products.

Bit9+CarbonBlack’s product offerings, Bit9 Platform Server and Carbon Black Server, have configuration remediations available.

Now, on to the cool stuff.

How Does this Attack Work?

(I will not be getting into the encryption details in this blog post, but for a great explanation of what is happening down to the nuts and bolts, check out the blog post by Daniel Fox Franke

And, for further details, please see the write-up by Bodo Möller, Thai Duong, and Krzysztof Kotowicz and POODLE attacks on SSLv3 by Adam Langley)

To exploit the vulnerability, the victim must be running JavaScript and the attacker has to be on the same network as the victim (e.g., the public Wi-Fi at the local coffee shop or airport Wi-Fi). Short version: any Wi-Fi network without AP isolation.

These restrictions make POODLE less actionable than an attack that can be conducted remotely against any computer on the Internet, but it is very effective once these requirements are met.

Since this exploit is triggered by a Man-in-the Middle (MITM) attacker to decrypt HTTPS cookies, techniques similar to those used against BEAST also are leveraged.

The latest implementations of SSL/TLS have more efficient version negotiation mechanisms that should prevent the usage of an older version. However, some servers don't implement version negotiation correctly. It happens. It's something we have to accept and work with.

Browsers can break the negotiation mechanism by retrying connections with older versions when initial TLS handshaking fails. Since this is the case, an attacker can cause a browser to speak SSLv3 to any server and then run the above attack by just injecting a few errors on the network.

The step-by-step at the thousand-foot view looks like this:

To launch the POODLE attack:

1. Run a JavaScript agent on http://example.com to get the victim's browser to send cookiebearing HTTPS requests to https://example.com.
   1. First requirement met, javascript.
2. Intercept and modify the SSL records sent by the browser in such a way that there's a nonnegligible chance that example.com will accept the modified record.
   1. Second requirement met, access to the same network as the victim. Also referred to as MITM.
3. If the modified record is accepted, the attacker can decrypt one byte of the cookies.
4. Repeat steps 2-3 until you have decrypted as many of the cookies as desired.

How Can I Tell If My Server is Vulnerable?

If your server is accessible from the Internet, you can use Qualys SSL Lab's excellent SSL test tool. In the Configuration / Protocols section, you should see SSL 3 listed as "No" (and SSL 2 also should be a "No" or you are vulnerable to other more severe vulnerabilities.)

If your server is not externally accessible, you can still test it easily using openssl:

```
openssl s_client -connect [YourServer]:443 -ssl3
```

If you successfully connect, then your server is still vulnerable. If you get a handshake failure, then you are not vulnerable.

In conclusion, don't panic. Because this attack requires JavaScript and MITM access, unless you are using public Wi-Fi or your network has already been popped (in which case you have bigger problems), this attack isn't a huge threat for the average user.

As technology keeps changing and new protocols emerge, we need to think about decommissioning the use of antiquated versions more proactively. A very small percentage of Internet traffic uses SSLv3 legitimately, and most of this traffic is via crawlers. We should use this vulnerability as a reminder to audit traffic to our servers and remove support for outdated protocols that are not in meaningful use.

It's time to take POODLE to obedience class.

Tags: bit9 | Carbon Black | Legacy Applications | Padding Oracle On Downgraded Legacy Encryption | POODLE | SSL 3.0 | web encryption

More Posts