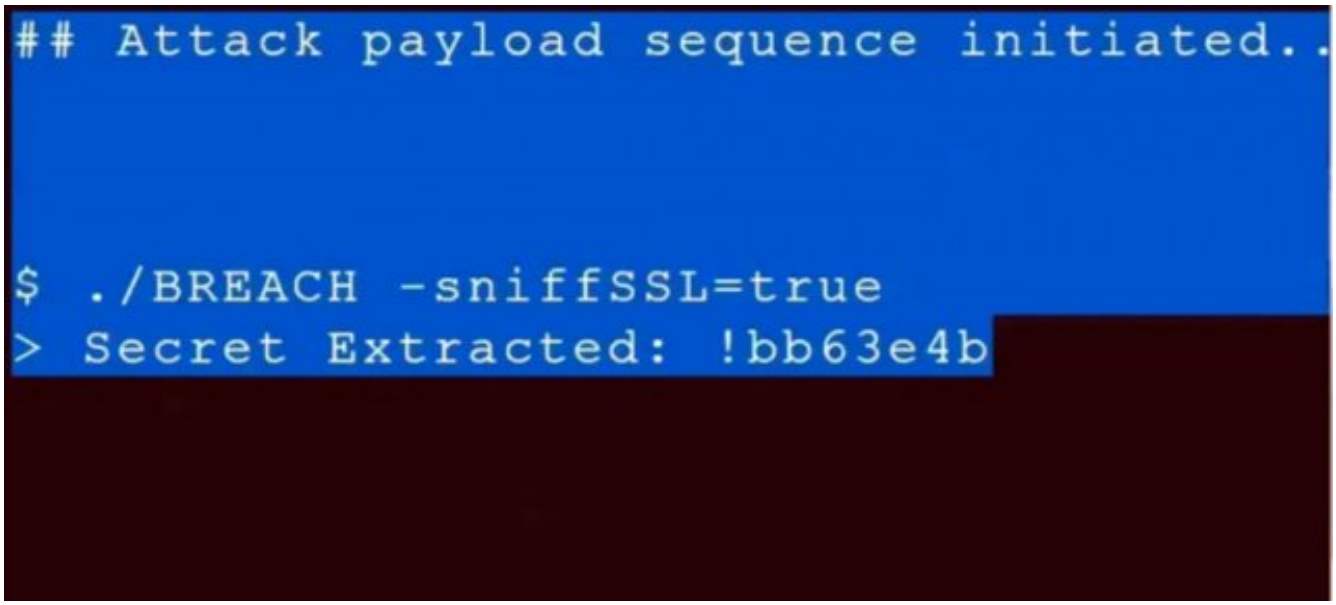
BIZ & IT —

Gone in 30 seconds: New attack plucks secrets from HTTPS-protected pages

Exploit called BREACH bypasses the SSL crypto scheme protecting millions of sites.

DAN GOODIN - 8/1/2013, 11:30 PM



Prado, Harris, and Gluck

A frame from a video demonstration showing BREACH in the process of extracting a 32-character security token in an HTTPS-encrypted Web page.

The HTTPS cryptographic scheme, which protects millions of websites, is susceptible to a new attack that allows hackers to pluck e-mail addresses and certain types of security credentials out of encrypted pages, often in as little as 30 seconds.

The technique, scheduled to be demonstrated Thursday at the Black Hat security conference in Las Vegas, decodes encrypted data that online banks and e-commerce sites send in responses that are protected by the widely used transport layer security (TLS) and secure sockets layer (SSL) protocols. The attack can extract specific pieces of data, such as social security numbers, e-mail addresses, certain types of security tokens, and password-reset links. It works against all versions of TLS and SSL regardless of the encryption algorithm or cipher that's used.

It requires that the attacker have the ability to passively monitor the traffic traveling between the end user and website. The attack also requires the attacker to force the victim to visit a malicious link. This can be done by injecting an iframe tag in a website the victim normally visits or, alternatively, by tricking the victim into viewing an e-mail with hidden images that automatically download and generate HTTP requests. The malicious link causes the victim's computer to make multiple requests to the HTTPS server that's being targeted. These requests are used to make "probing guesses" that will be explained shortly.

"We're not decrypting the entire channel, but only extracting the secrets we care about," Yoel Gluck, one of three researchers who developed the attack, told Ars. "It's a very targeted attack. We just need to find one corner [of a website response] that has the token or password change and go after that page to extract the secret. In general, any secret that's relevant [and] located in the body, whether it be on a webpage or an Ajax response, we have the ability to extract that secret in under 30 seconds, typically."

It's the latest attack to chip away at the HTTPS encryption scheme, which forms the cornerstone of virtually all security involving the Web, e-mail, and other Internet services. It joins a pantheon of other hacks introduced over the past few years that bear names such as CRIME, BEAST, Lucky 13, and SSLStrip. While none of the attacks have completely undermined the security afforded by HTTPS, they highlight the fragility of the two-decade-old SSL and TLS protocols. The latest attack has been dubbed BREACH, short for Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext.

As its name suggests, BREACH works by targeting the data compression that just about every website uses to conserve bandwidth. Based on the standard Deflate algorithm, HTTP compression works by eliminating repetitions in strings of text. Rather than iterating "abcd" four times in a chunk of data, for instance, compression will store the string "abcd" only once and then use space-saving "pointers" that indicate where the remaining three instances of the identical pattern are found. By reducing the number of bytes sent over a connection, compression can significantly speed up the time required for a message to be received. In general, the more repetitions of identical strings found in a data stream, the more potential there will be for compression to reduce the overall size.

Using what's known as an oracle technique, attackers can use compression to gain crucial clues about the contents of an encrypted message. That's because many forms of encryption—including those found in HTTPS—do little or nothing to stop attackers from seeing the size of the encrypted payload. Compression oracle techniques are particularly effective at ferreting out small chunks of text in the encrypted data stream.

BREACH plucks out targeted text strings from an encrypted response by guessing specific characters and including them in probe requests sent back to the targeted Web service. The attack then compares the byte length of the guess to the original response. When the guess contains the precise combination of characters found in the original response, it will generally result in a payload that's smaller than those produced by incorrect guesses. Because deflate compression stores the repetitive strings without significantly increasing the size of the payload, correct guesses will result in encrypted messages that are smaller than those produced by incorrect guesses.

The first thing an attacker using BREACH might do to retrieve an encrypted e-mail address is guess the @ sign and Internet domain immediately to its right. If guesses such as "@arstechnica.com" and "@dangoodin.com" result in encrypted messages that are larger than the request/response pair without this payload, the attacker knows those addresses aren't included in the targeted response body. Conversely, if compressing "@example.com" against the encrypted address results in no length increase, the attacker will have a high degree of confidence that the string is part of the address he or she is trying to extract. From there, attackers can guess the string to the left of the @ sign character by character.

Assuming the encrypted address was johndoe@example.com, guesses of a@example.com, b@example.com, c@example.com, and d@example.com would cause the encrypted message to

grow. But when the attacker guesses e@example.com, it would result in no appreciable increase, since that string is included in the targeted message. The attacker would then repeat the same process to recover the remainder of the e-mail address, character by character, moving right to left.

The technique can be used to extract other types of encrypted text included in Web responses. If the site being targeted sends special tokens designed to prevent so-called cross-site request forgery attacks, the credential will almost always contain the same format—such as "request_token=" followed by a long text string such as"bb63e4ba67e24d6b81ed425c5a95b7a2"—each time it's sent. The compression oracle attack can be used to guess this secret string.

An attacker would begin by adding the text "request_token=a" to the text of the encrypted page being targeted and send it in a probe request to the Web server. Since the size of the encrypted payload grows, it would be obvious this guess is wrong. By contrast, adding "request_token=b" to the page wouldn't result in any appreciable increase in length, giving the attacker a strong clue that the first character following the equal sign is b. The attacker would use the same technique to guess each remaining character, one at a time, moving left to right.

Most attacks that use the BREACH technique can be completed by making only a "few thousand" requests to the targeted Web service, in about 30 seconds with optimal network conditions and small secrets, and in minutes to an hour for more advanced secrets.

BREACH, which was devised by Gluck along with researchers Neal Harris and Angelo Prado, builds off the breakthrough CRIME attack researchers Juliano Rizzo and Thai Duong demonstrated last September. Short for Compression Ratio Info-leak Made Easy, CRIME also exploited the compression in encrypted Web requests to ferret out the plaintext of authentication cookies used to access private user accounts. The research resulted in the suspension of TLS compression and the temporary disabling of an open networking compression protocol known as SPDY until it could be modified. BREACH, by contrast, targets the much more widely used HTTP compression that virtually all websites use when sending responses to end users. It works only against data sent in responses by the website.

"If you go to the Wikipedia page or any of the specialized security pages, they will tell you that CRIME is mitigated as of today and is no longer an interesting attack and nobody cares about it," Prado said. "So we are bringing it back and making it work better, faster in a different context."

The good news concerning BREACH is that it works only against certain types of data included in Web responses and then only when an attacker has succeeded in forcing the victim to visit a malicious link. Still, anytime an attacker can extract sensitive data shielded by one of the world's most widely used encryption schemes it's a big deal, particularly as concerns rise about NSA surveillance programs. Making matters more unsettling, there are no easy ways to mitigate the damage BREACH can do. Unlike TLS compression and SPDY, HTTP compression is an essential technology that can't be replaced or discarded without inflicting considerable pain on both website operators and end users.

At their Black Hat demo, the researchers will release a collection of tools that will help developers assess how vulnerable their applications and online services are to BREACH attacks. Most mitigations will be application-specific. In other cases, the attacks may give rise to new "best practices" advice on how to avoid including certain types of sensitive data in encrypted Web responses. Most websites already list only the last four digits of a customer's credit card number; BREACH may force websites to truncate other sensitive strings as well.

"We expect that it could be leveraged in particular situations, maybe with an intelligence agency, or maybe an individual actor or a malicious crime organization might use this in a targeted scenario," Prado said. "Any malware writer today has the ability to do something like this if they have not been doing it already."

Article updated to correct statement that SPDY was suspended. It was temporarily disabled until it could be modified.

Promoted Comments

Jacob640 / Smack-Fu Master, in training / *et Subscriptor* JUMP TO POST

> dtrumpet wrote:
> > Jacob640 wrote:
> > > tkeer wrote:
> > > I could be under thinking, but seems like a simple fix - a few randomly generated characters in an HTML comment at the bottom of the page would randomize the size, making it much more difficult to infer any useful information.
> >
> > Using padding would only increase the number of messages required to extract the secret because you can use statistics to compensate for the random variation. By injecting the same string multiple times the attacker can find out if the average length after injection deviates from the expected average and thus determine if the message length has changed. Increasing the proportion of padding to message only increase the size of sample required to determine if a change has occurred. However the average number of messages required to perform such an attack should be fairly easy to calculate and so you could set a dynamic padding:message ratio which is calculated to increase the attack time hopefully unfeasible levels.
> >
> > However it might be possible that doing this would increase the page size so much that you might as well disable the HTTP compression altogether.
>
> This is another form of the "increasing key size/password length just increases the average number of guesses required to recover the desired secret" argument. Making things computationally infeasible is a fundamental tool of security. And, since we're talking about an online attack here, increasing the number of required guesses also increases the odds of detection.

Exactly, that's why I suggested a padding ratio that requires an unfeasible number of messages. But my main point is that if you have to add so much data to the page that you increase the page size significantly it might be just eaisier to disable compression on dynamic assets since the random padding bytes that you add will be almost totally incompressible.

Additionally whilst such online attacks should be easy to detect in theory the practical problems and technical challenges will mean that some sites, esp. smaller ones will not have such capabilities in the near future.

26 posts | registered Jan 22, 2013

chairwheel / Smack-Fu Master, in training JUMP TO POST

> nibb wrote:
> Most web server operators and companies are just putting everything under SSL which is ridiculous. An article like this in Ars does not need to be protected. Sensitive data should go under SSL connections, public data should not for the simple reason that you will able to server 100 times more traffic without it. I just hate it when I see SSL locks on blogs and other public pages which make no sense to protect in the first page and there is no user interaction either, not to mention when they display mixed insecure elements in the same page....

An argument for encrypting non-sensitive pages (such as the pages of a bank's website listing their branch offices and business hours) is that it makes it harder to do simple man-in-the-middle replacement of page content. The bad guy then would not need to trick the user into visiting a malicious URL to accomplish the attack on the secure part of the site, they just need to inject that invisibly into the non-secure pages.

Even aside from security concerns, there are unreputable ISPs that redirect all their customers' web traffic through proxy servers which replace the ads in the sites visited with their own ads (or inserting additional ads), thus diverting the ad revenue stream from the site owner to the ISP. Using SSL also makes this harder for the ISP to do and is a form of ensuring your site is delivered to the end user exactly like you intend it.

If the site operator has the compute capacity for running everything through SSL, what concern is it of yours.

1 post | registered Aug 2, 2013

READER COMMENTS 66

SHARE THIS STORY

DAN GOODIN

Dan is the Security Editor at Ars Technica, which he joined in 2012 after working for The Register, the Associated Press, Bloomberg News, and other publications.

EMAIL dan.goodin@arstechnica.com // **TWITTER** @dangoodin001

Myst: The challenges of CD-ROM | War Stories

Cyan Worlds co-founder Rand Miller goes behind the scenes of the development of one of the best selling PC games of all time, Myst. The HyperCard-developed title ran into some snags when trying to run on the CD-ROM format. Rand and his brother, Robyn, compressed the image and audio data as much as they could so the game could run smoothy on 1x CD-ROM drives.

High Speeds On A Small Scale

How Mind Control Saved Oddworld: Abe's Oddysee

More videos

← PREVIOUS STORY

NEXT STORY →

Related Stories

Today on Ars

NEWSLETTER SIGNUP

Join the Ars Orbital Transmission mailing list to get weekly updates delivered to your inbox.

SIGN ME UP →