NONCE-DISRESPECTING ADVERSARIES

PRACTICAL FORGERY ATTACKS ON GCM IN TLS

Hanno Böck, Aaron Zauner, Sean Devlin, Juraj Somorovsky, Philipp Jovanovic

TLS ENCRYPTION

Step 1: Asymmetric key exchange (RSA, DHE, ECDHE) to generate shared keys

Step 2: Symmetric encryption and authentication

Today we're interested in Step 2.

TLS SYMMETRIC MODES

CBC/HMAC

RC4 (stream cipher)

GCM

(new: ChaCha20/Poly1305)

CBC/HMAC

Vaudenay Padding Oracle (2002), Vaudenay/Moeller (2003/2004), BEAST (2011), Lucky Thirteen (2013), POODLE (2014), POODLE-TLS (2014), more POODLEs (2015), Lucky Microseconds (2015), Padding Oracle in OpenSSL / CVE-2016-2107

CBC/HMAC PROBLEMS

CBC/HMAC in TLS used an implicit IV in TLS 1.0.

The padding content in SSLv3 was undefined.

All TLS versions use MAC-then-Pad-then-Encrypt.

Encrypt-then-MAC extension, but it's rarely used.

TLS 1.2 PREDICTS LUCKY THIRTEEN

This leaves a small timing channel, since MAC performance depends to some extent on the size of the data fragment, but it is not believed to be large enough to be exploitable, due to the large block size of existing MACs and the small size of the timing signal. (TLS 1.2, RFC 5246)

LUCKY THIRTEEN IS A BIG MESS

Amazon tried to implement countermeasures that didn't work.

Some implementations (Go, TLS Lite) are known vulnerable and don't want to fix it.

OpenSSL introduced another (worse) padding oracle while fixing Lucky Thirteen.

RC4

After Lucky Thirteen many sites switched to RC4.

Fluhrer/Shamir/Mantin (2001), attack on TLS by AlFardan/Bernstein/Patterson (2013), Bar-Mitzva-Attack/Mantin (2015), Garman/va der Merwe/Paterson (2015), Vanhoef/Piessens (2015).

RFC 7465: "Prohibiting RC4 Cipher Suites"

GCM

GCM - Galois/Counter Mode.

Usually used with AES.

Only available in TLS 1.2.

"This seems like a good moment to reiterate that everything less than TLS 1.2 with an AEAD cipher suite is cryptographically broken." (Adam Langley)

WHAT IS GCM?

GCM is an AEAD (Authenticated Encryption with Additional Data)

Rationale: If you give people an encryption mode and an authentication mechanism they will combine it in an insecure way. So give them a standard that combines both.

GCM is a combination of Counter Mode and a GHASH authentication tag.

GCM - OPINIONS

"Do not use GCM. Consider using one of the other authenticated encryption modes, such as CWC, OCB, or CCM." (Niels Ferguson)

"We conclude that common implementations of GCM are potentially vulnerable to authentication key recovery via cache timing attacks." (Emilia Käsper, Peter Schwabe, 2009)

"AES-GCM so easily leads to timing side-channels that I'd like to put it into Room 101." (Adam Langley, 2013)

"The fragility of AES-GCM authentication algorithm" (Shay Gueron, Vlad Krasnov, 2013)

"GCM is extremely fragile" (Kenny Paterson, 2015)

GCM

Everybody uses GCM, but nobody likes it.

"There's also an annoying niggle with AES-GCM in TLS because the spec says that records have an eight byte, explicit nonce. Being an AEAD, the nonce is required to be unique for a given key. Since an eight-byte value is too small to pick at random with a sufficiently low collision probability, the only safe implementation is a counter. [...] Thankfully, all the major implementations use a counter and I did a scan of the Alexa, top 200K sites to check that none are using random values - and none are." (Adam Langley)

NONCE

Number used once.

If you use the same Nonce twice (with the same key) it's no longer a nonce.

TLS: 8 Byte / 64 Bit nonce

THE SPEC (RFC 5288 / TLS 1.2)

"Each value of the nonce_explicit MUST be distinct for each distinct invocation of the GCM encrypt function for any fixed key. Failure to meet this uniqueness requirement can significantly degrade security. The nonce_explicit MAY be the 64-bit sequence number."

INTERNET-WIDE SCAN RESULTS

184 hosts with repeating nonces

72445 hosts with random looking nonces

FINDING AFFECTED VENDORS

Certificate info, website content, HTTP "Server:" header.

Often load balancers hiding their true identity.

Contacting website owners hardly works.

IT LOOKS RANDOM, BUT ISN'T

Check Point devices using LFSR - this is secure.

DUPLICATE NONCES

We could identify two vendors

Radware (Cavium chip), update from vendor

Several pages from VISA Europe (vendor not yet disclosed)

DEVICES WITH RANDOM NONCES

A10, IBM Lotus Domino (both published updates).

Sangfor (no vendor response).

MORE?

There are more devices with different behaviors that we were unable to identify.

Security test tools and pen testers should checks for this.

WHAT'S THIS? (RADWARE AND OTHERS)

```
0100000003001741
0100000003001741
f118cd0fa6ff5a15
f118cd0fa6ff5a16
f118cd0fa6ff5a74
```

OPENSSL 1.0.1J

t1_enc.c:

```
        if (EVP_CIPHER_mode(c) == EVP_CIPH_GCM_MODE)
                {
                EVP_CipherInit_ex(dd,c,NULL,key,NULL,(which & SSL3_CC_WRITE)
                EVP_CIPHER_CTX_ctrl(dd, EVP_CTRL_GCM_SET_IV_FIXED, k, iv);
                }
```

e_aes.c (EVP_CIPHER_CTX_ctrl/aes_gcm_ctrl):

```
                if (c->encrypt &&
                        RAND_bytes(gctx->iv + arg, gctx->ivlen - arg) <= 0)
                        return 0;
```

THE FORBIDDEN ATTACK

Described by Joux during NIST GCM standardization (2006).

Nonce reuse allows an attacker to recover the authentication key.

Attacker can modify messages with high precision.

GCM BACKGROUND

K: the encryption key

$H = E_K(0)$: the authentication key derived by encrypting the all-zero block under K

N: the per-encryption nonce

GCM AUTHENTICATION

high level view:

- format the message as a polynomial
- mask with $E_K(N)$
- plug in H

GCM AUTHENTICATION

high level attacker's view:

- find a polynomial with H a root
- factor the polynomial (this is easy!)
- each root is a candidate for H (usually only a few!)

GCM AUTHENTICATION

For concreteness, consider a message with no AAD and one block of ciphertext.

$$f(x) = C_1 x^2 + Lx + E_K(N)$$

$$f(H) = T$$

- L: 128-bit block encoding the message length
- T: the output authentication tag

THE ATTACKER KNOWS ALGEBRA!

Subtract T:

$$f'(x) = C_1 x^2 + Lx + E_K(N) - T$$

$$f'(H) = 0$$

H is a root of f', and we have efficient algorithms for finding roots of a polynomial!

Problem: we don't know f'.

NONCE REUSE

Suppose we have two messages encrypted under the same nonce:

$$f_1'(x) = C_{1,1}x^2 + L_1x + E_K(N) - T_1$$

$$f_2'(x) = C_{2,1}x^2 + L_2x + E_K(N) - T_2$$

$$g(x) = f_1'(x) - f_2'(x)$$

$$g(x) = (C_{1,1} - C_{2,1})x^2 + (L_1 - L_2)x - (T_1 - T_2)$$

NONCE REUSE

$$g(x) = (C_{1,1} - C_{2,1})x^2 + (L_1 - L_2)x - (T_1 - T_2)$$

$$g(H) = 0$$

g is fully known to the attacker: we can factor it to recover H.

A MITM ATTACK ON TLS

1. User visits
   http://attacker.com.

A MITM ATTACK ON TLS

1. User visits http://attacker.com.
2. Attacker serves JavaScript to poll https://nonce-repeater.com.

A MITM ATTACK ON TLS

1. User visits http://attacker.com.
2. Attacker serves JavaScript to poll https://nonce-repeater.com.
3. Attacker collects responses indexed by nonce.

A MITM ATTACK ON TLS

1. User visits http://attacker.com.
2. Attacker serves JavaScript to poll https://nonce-repeater.com.
3. Attacker collects responses indexed by nonce.
4. When the server repeats a nonce, attacker executes Joux's attack to recover the authentication key.

A MITM ATTACK ON TLS

1. User visits http://attacker.com.
2. Attacker serves JavaScript to poll https://nonce-repeater.com.
3. Attacker collects responses indexed by nonce.
4. When the server repeats a nonce, attacker executes Joux's attack to recover the authentication key.
5. Attacker redirects user to a known resource, e.g. https://nonce-repeated.com/index.html.

A MITM ATTACK ON TLS

1. User visits http://attacker.com.
2. Attacker serves JavaScript to poll https://nonce-repeater.com.
3. Attacker collects responses indexed by nonce.
4. When the server repeats a nonce, attacker executes Joux's attack to recover the authentication key.
5. Attacker redirects user to a known resource, e.g. https://nonce-repeated.com/index.html.
6. Attacker replaces the ciphertext C in the server response with (C XOR index.html XOR malicious.html) and updates the tag.

A MITM ATTACK ON TLS

```
HTTP/1.1 301 Moved Permanently
Strict-Transport-Security: max-age=31536000
Date: Tue, 02 Aug 2016 20:47:06 GMT
Server: Apache
X-Frame-Options: SAMEORIGIN, SAMEORIGIN
Location: https://www.mi5.gov.uk/careers?146718903ac4b72b
Cache-Control: max-age=1209600
Expires: Tue, 16 Aug 2016 20:47:06 GMT
Content-Length: 255
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="https://www.mi5.gov.uk/careers?146718903ac4b72b">here</a>.</p>
</body></html>
```

```
HTTP/1.1 200 OK
GCM: lol
Ignore: rict-Transport-Security: max-age=31536000
Date: Tue, 02 Aug 2016 20:47:06 GMT
Server: Apache
X-Frame-Options: SAMEORIGIN, SAMEORIGIN
Location: https://www.mi5.gov.uk/careers?146718903ac4b72b
Cache-Control: max-age=1209600
Expires: Tue, 16 Aug 2016 20:47:06 GMT
Content-Length: 255
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

<html><body style="margin:0"><script>document.body.style.height = window.innerHeight+'px';</script><iframe src="https://attacker.org/blackhat/" style="width:100%;height:100%" frameBorder="0"></iframe></body></html>
```

FUTURE

Draft for Chacha20/Poly1305 and TLS 1.3 uses fully implicit nonce based on record number.

Synthetic IVs and nonce misuse resistant schemes.

CONCLUSION

TLS 1.2 tells implementors to use a nonce, but gives no guidance how to do that properly.

Some people get it wrong.

We need better test tools for TLS implementation flaws (TLS-Fuzzer looks promising).

THANKS FOR LISTENING

https://github.com/nonce-disrespect/nonce-disrespect

Test your hosts:

https://gcm.tlsfun.de/