Raccoon Attack: Finding and Exploiting Most-Significant-Bit-Oracles in TLS-DH(E)

Robert Merget, ¹ Marcus Brinkmann, ¹ Nimrod Aviram, ²
Juraj Somorovsky, ³ Johannes Mittmann, ⁴ Jörg Schwenk ¹

Usenix Security 2021

¹ Ruhr University Bochum
² School of Computer Science, Tel Aviv University
³ Paderborn University
⁴ Bundesamt für Sicherheit in der Informationstechnik (BSI), Germany
TLS-DH(E)

\[ g^a \mod p, g^b \mod p, \text{signature} \]

\[ a \in \mathbb{Z}_p \]

\[ b \in \mathbb{Z}_p \]

\[ g^{ab} \mod p \]

PRF\((g^{ab}, n_c, n_S, \text{“master secret”}) = \text{master}_\text{secret}\)
A conventional Diffie-Hellman computation is performed. The negotiated key (Z) is used as the pre_master_secret, and is converted into the master_secret, as specified above. **Leading bytes of Z that contain all zero bits are stripped** before it is used as the pre_master_secret.
Constant Time Execution

TLS key derivation is based on hash functions

Hash functions operate in $O(n)$ not $O(1)$

This creates various side-channels:

- Compression function invocation
- Hash function invocation
- Key padding
- Direct side-channel

Example: SHA-256 in OpenSSL
Attack Overview

\[ \text{PMS} = (g^a \cdot g^r)^b = g^{ab} \cdot g^{rb} \]
Retrieving the PMS

Constructing Instance of Hidden Number Problem:
\[ \alpha \cdot t_1 = y_1 \pmod{p}, \]
\[ \alpha \cdot t_2 = y_2 \pmod{p}, \]
\[ \alpha \cdot t_3 = y_3 \pmod{p}, \ldots \]

Constructing Instance of Hidden Number Problem:
\[ \alpha = g^{ab}, \quad t_i = g^{r_i b}, \quad 0 < y_i < 2^{n-k} \]
# Performance

<table>
<thead>
<tr>
<th>DH Group</th>
<th>Bit Length</th>
<th>k=24</th>
<th>k=20</th>
<th>k=16</th>
<th>k=12</th>
<th>k=8</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 5114</td>
<td>1024</td>
<td>d=50</td>
<td>d=60</td>
<td>d=80</td>
<td>d=100</td>
<td>d=200</td>
</tr>
<tr>
<td></td>
<td></td>
<td>T=6s</td>
<td>T=10s</td>
<td>T=26s</td>
<td>T=111</td>
<td>T~2,5h</td>
</tr>
<tr>
<td>LibTomCrypt</td>
<td>1036</td>
<td>d=50</td>
<td>d=60</td>
<td>d=80</td>
<td>d=100</td>
<td>d=180</td>
</tr>
<tr>
<td></td>
<td></td>
<td>T=6s</td>
<td>T=10s</td>
<td>T=28s</td>
<td>T=52s</td>
<td>T~1,5h</td>
</tr>
<tr>
<td>SKIP</td>
<td>2048</td>
<td>d=100</td>
<td>d=120</td>
<td>d=160</td>
<td>Unsolved</td>
<td>Unsolved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>T=112</td>
<td>T=207</td>
<td>T=977s</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

k = Leading zero bits leaked  
d = Number of equations used  
T = Time to solve HNP
Impact

Scan of Alexa Top 100k:

- 32% of the scanned servers supported DHE cipher suites
- 10.9% of those servers reused their ephemeral keys

Firefox was the last browser to drop support in September 2020

No major browser supports DHE anymore
Direct Raccoon

CVE-2020-2529: F5-Big IP leaks leading zero byte

$g^a \cdot g^r$

1x Alert | 2x Alert
Countermeasure

Generally:

- Do not leak partial information about secret values
- Make secrets constant size

For TLS:

- Clients should avoid DH(E)
- Servers should not reuse ephemeral keys
- Servers and clients should not use DH
Raccoon and ECDH(E)

Leading zero bytes of shared ECDH secrets maintained

Requires implementation-specific side-channels

Further research required, currently not exploitable
Raccoon and TLS 1.3

Leading zero bytes of ALL shared secrets maintained

Foresight by David Benjamin in Draft-13 proved useful

Ephemeral key reuse is uncommon
Why the mess?

It seems that many developers of SSL3/TLS have independently read PKCS3
and all missed the fact that it defines the derived output of "Phase II"
to have exactly the same number of octets and the number of significant
octets of prime P. Consequently, those developers independently all
implemented DH to strip leading zero octets from the result, and their
SSL3/TLS implementations of DH and DHE ciphersuites all interoperated
due to their common (arguably mistaken) interpretations of PKCS3.
This is observed in SSLeay (forerunner of OpenSSL), OpenSSL, in NSS,
and in all presently interoperable implementations of SSL3/TLS DHE
ciphersuites. The current draft of the next TLS RFC explicitly
requires leading zeros to be removed from the DH output before being
treated as the pre-master secret. See section 8.1.2 of
Raccoon & DH(E) Proofs

Security proofs exist:

- TLS DH(E) (Jager et. al) (CRYPTO 2012)
- TLS-DH (Krawczyk et. al) (CRYPTO 2013)

Zero byte stripping/timing is not modeled

Proofs rely on PRF-ODH assumption

Assumption is not in practice
Conclusion

- No need to panic, exploitation is difficult
- The Raccoon attack is not TLS specific
- First time HNP is used to attack DH

More info: [https://raccoon-attack.com](https://raccoon-attack.com)

Tool to scan your servers:
[https://github.com/tls-attacker/TLS-Scanner](https://github.com/tls-attacker/TLS-Scanner)

@ic0nz1
robert.merget@rub.de