
Mathy Vanhoef

Keynote, IFIP WG 11.4 “Network & Distributed Systems Security” Workshop. May 16, 2024.
Introduction

Goal of this talk:
› Explain some interesting network attacks + demos 😊
› Common theme: attacks are enabled by novel threat model

I will use the word “threat model” rather informally:
› In some attacks, the adversary is given extra capabilities
› In other attacks, the focus is more on new attack techniques
Agenda

› Attacks that introduced new threat models:
  » The BEAST and HEIST attack (TLS/HTTPS)
  » The Multi-Channel MitM (KRACK)
  » Outbound Connections (FragAttacks)
  » DNS Spoofing & VPNs (TunnelCrack)

› Conclusion
The BEAST attack against SSL/TLS

› Phillip Rogaway (‘95): CBC encryption can be attacked when the Initialization Vectors (IVs) are predictable

› Fixed in TLS1.1, but TLS1.0 was still very common
  » “It’s hard to abuse, so not important to fix”

› Duong & Rizzo (‘11): attacked CBC in practice by assuming malicious JavaScript in the browser + network MitM
  » And extended attack to achieve full plaintext recovery
  » Sudden scramble to update implementations

Reference: One Bad Apple: Backwards Compatibility Attacks on State-of-the-Art Cryptography
The BEAST Threat Model

› Arguably most influential contribution was the threat model:
  › Attack can execute JavaScript in the victim’s browser
  › And attacker can intercept (encrypted) network traffic

› This completely broke an established protocol in practice

› The “BEAST threat model” was (and is) used in many works
  › In many attacks against RC4, including our RC4 NOMORE attack
  › Many TLS attacks (Lucky13, Bleichenbacher attacks, DROWN)
  › In the CRIME and BREACH attack to abuse compression
Abusing compression

CRIME and BREACH attack
› Abused compression at the TLS and HTTP level to leak information in response, e.g., *leak CRSF tokens*
› Assumed execution of malicious JavaScript + network MitM
   » Network MitM was used to measure length of response

TIME and HEIST attack
› Like BREACH abuses compression to recover CRSF token
› But uses *timing side-channels instead of needing MitM*
DEMO: HEIST Attack

Welcome, Mr. Sniffles

Failed to send carrots to xtahsicqcoy: insufficient funds available

Balance

Your balance is -6 carrots. Need more? Apply for a loan now!

Recent transactions
Reflection

› The new “BEAST threat model” enabled various follow-up works to construct more practical attacks

› Some attacks were further improved to reduce the required capabilities of the attacker
“Attacks only get better, they never get worse.”

— Bruce Schneier
Agenda

› Attacks that introduced new threat models:
   » The BEAST attack (TLS)
   » The Multi-Channel MitM (KRACK)
   » Outbound Connections (FragAttacks)

› Conclusion
Reinstallation Attack

 effet. Called a “Multi-Channel MitM” (MC-MitM)
Reinstallation Attack

optional 802.1x authentication
### Reinstallation Attack

<table>
<thead>
<tr>
<th>Phone</th>
<th>802.1x Authentication</th>
<th>Router</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \text{optional 802.1x authentication} )</td>
<td></td>
</tr>
<tr>
<td>( \text{Msg1}(r, \text{ANonce}) )</td>
<td>( \text{Msg1}(r, \text{ANonce}) )</td>
<td></td>
</tr>
<tr>
<td>( \text{Msg2}(r, \text{SNonce}) )</td>
<td>( \text{Msg2}(r, \text{SNonce}) )</td>
<td></td>
</tr>
<tr>
<td>( \text{Msg3}(r+1; \text{GTK}) )</td>
<td>( \text{Msg3}(r+1; \text{GTK}) )</td>
<td></td>
</tr>
</tbody>
</table>
# Reinstallation Attack

<table>
<thead>
<tr>
<th>Optional 802.1x authentication</th>
<th>Reinstallation Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Msg1(r, ANonce)</strong></td>
<td><strong>Msg1(r, ANonce)</strong></td>
</tr>
<tr>
<td><strong>Msg2(r, SNonce)</strong></td>
<td><strong>Msg2(r, SNonce)</strong></td>
</tr>
<tr>
<td><strong>Msg3(r+1; GTK)</strong></td>
<td><strong>Msg3(r+1; GTK)</strong></td>
</tr>
<tr>
<td><strong>Msg4(r+1)</strong></td>
<td><strong>Block Msg4</strong></td>
</tr>
</tbody>
</table>

**Install PTK & GTK**

**Note:** The diagram illustrates the flow of messages in a Reinstallation Attack scenario, with the final step being the block of the 4th message to prevent reinstalling the PTK and GTK.
Reinstallation Attack

Install PTK & GTK

Msg4(r+1)

Block Msg4
Reinstallation Attack

In practice, Msg4 is sent encrypted.
Reinstallation Attack

Key reinstallation!
Packet number is reset
Reinstallation Attack

Msg4(r+1)

Install PTK & GTK

Msg3(r+2; GTK)

Enc^1_{ptk}\{Msg4(r+2)\}

Reinstall PTK & GTK

Enc^1_{ptk}\{Data(…)}

Same packet number is used!
Reinstallation Attack

Keystream

Install PTK & GTK

Msg4(r+1)

Msg3(r+2; GTK)

Enc_{ptk}^{1} \{ Msg4(r+2) \}

Enc_{ptk}^{1} \{ Data(...) \}

Decrypted!
Reinstallation Attack

wpa_supplicant 2.4+
installed all-zero key
Reinstallation Attack

Msg4 may be lost due to noise: attack can occur “naturally”!!
Installation of all-zero key was detected (!!)

Bug report on Linux’s hostap mailing list:

“While testing with supplicant 2.4 we observed [..]:

4. We send M4 and install PTK
5. We received M3 again
6. We send M4 and install PTK

… we install it as 0 again in step (6)”
This bug was then fixed

- “[..] possibility of the authenticator having to retry EAPOL-Key message 3/4 in case the first EAPOL-Key message 4/4 response is lost. That case ended up trying to reinstall the same TK to the driver, but the key was not available”
- They didn’t realize an adversary can force this situation
- The MC-MitM threat model that allows us to do this reliably!

[1] An issue with supplicant receiving retransmitted M3 (Atul Joshi)
[2] An issue with supplicant receiving retransmitted M3 (Jouni Malinen)
[3] Fix TK configuration to the driver in EAPOL-Key 3/4 retry case
The MC-MitM is used in several works now

› The MC-MitM was originally used by us to break WPA-TKIP
› Was used to infer resource sizes in combination with malicious JavaScript, i.e., in a BEAST-like attack
› To exploit an implementation flaw in Broadcom code
› In our “framing frames” attack
› Also used in the FragAttacks research

References:
• Advanced WiFi Attacks Using Commodity Hardware (ACSAC’14)
• Request and Conquer: Exposing Cross-Origin Resource Size (USENIX Sec ’16)
• Discovering Logical Vulnerabilities in the Wi-Fi Handshake Using Model-Based Testing (Asia CCS ’17)
• Framing Frames: Bypassing Wi-Fi Encryption by Manipulating Transmit Queues (USENIX Sec ’23)
Agenda

› Attacks that introduced new threat models:
  › The BEAST and HEIST attack (TLS/HTTPS)
  › The Multi-Channel MitM (KRACK)
  › **Outbound Connections (FragAttacks)**

› Conclusion
Background

Sending small frames causes high overhead:

```
header  packet1  ACK  header  packet2  ACK  ...
```

This can be avoided by **aggregating frames**:

```
header’ packet1 packet2 ... ACK
```
Background

Sending small frames causes high overhead:

This can be avoided by **aggregating frames**:

**Problem: how to recognize aggregated frames?**
Aggregation design flaw

- Not authenticated
- False
- True
- header
- aggregated?
- encrypted
- packet
- packet1
- metadata
- len
- len
- packet2
Aggregation design flaw

Not authenticated

header | aggregated? | encrypted

False

True | metadata | len | packet1 | metadata | len | packet2

Flip flag → decrypted payload is parsed in wrong manner
Flaw was noticed while 802.11n was being standardized, but implementations based on the draft already existed (2007)

“QoS bit 7 should be protected to guard against attack that at minimum leads to a flood of traffic”

“While it is hard to see how this can be exploited, it is clearly a flaw that is capable of being fixed.”

Exploit by using new threat model 😊 (2021)
Exploit steps

Example:

- Send **e-mail** with embedded image
- Send **WhatsApp** message to cause link/image preview
Exploit steps

Get image from attacker’s server

Send special IPv4 packet
Exploit steps

1. Get image from attacker’s server
2. Encrypt as normal frame
3. Send special IPv4 packet
   - Encrypt as normal frame
Exploit steps

1. Get image from attacker’s server
2. Set aggregated flag
3. Encrypt as normal frame
4. Send special IPv4 packet
5. Can’t modify encrypted content

Set aggregated flag
Exploit steps

Get image from attacker’s server

Set aggregated flag

Inject any packet → **Inject** ICMPv6 RA with malicious DNS server
Exploit steps

Get image from attacker’s server

→ Easier than BEAST & HEIST attack against TLS!

Inject any packet → Inject ICMPv6 RA with malicious DNS server

Send special IPv4 packet

Encrypt as normal frame

Set aggregated flag
Easier version

Inject any packet $\rightarrow$ Inject ICMPv6 RA with malicious DNS server

Set aggregated flag

Encrypt as normal frame

Inject special \textit{handshake} frame

Bug in AP $\rightarrow$ do attack \textit{w/o user interaction}

(affected $\frac{2}{4}$ of home APs)
DEMO: FragAttacks A-MSDU Flaw
Conclusion

› Established protocols, when used in new situations and under new thread models, may become vulnerable to new attacks → Keep studying old protocols!

› When reading about attacks, learn about their threat model. That may be the most useful thing to know in the long term.

› Attacks only get better → threat models only get better?