Dragonblood is Still Leaking: Practical Cache-based Side-Channel in the Wild

Daniel De Almeida Braga
Pierre-Alain Fouque
Mohamed Sabt
CSAW 2020 - November, 6th 2020
Context and Motivations

Client

Access Point

WEP

1999

2003

WEP

2004

WPA

2018

WPA2

WPA3

today
Context and Motivations

Client

WEP
1999

WPA
2003

WPA2
2004

WPA3
2018

Access Point
Context and Motivations

Client

Access Point

WEP 1999

WPA 2003

WPA2 2004

WPA3 2018

today 2
Context and Motivations

Client

Access Point

WEP

1999

WPA

2003

WPA2

2004

2018

Office dictionary attack

KRACK attack
Context and Motivations

- **1999**: WEP
- **2003**: WPA
- **2004**: WPA2
- **2018**: WPA3
- **today**:

  + More secure
  + Based on a PAKE (Dragonfly)
PAKE: Password Authenticated Key Exchange

- PAKE protocols aim to combine the Key Exchange and authentication parts
- Password is used to:
  - Authenticate the user
  - Derive strong cryptographic material
- No offline dictionary attack
Context and Motivations

- More secure
- Based on a PAKE (Dragonfly)

Dragonblood attacks
def processPassword(pwd):
    if "a" in pwd:
        res = long_processing(pwd)
    else:
        res = short_processing(pwd)
    return res
def processPassword(pwd):
    if "a" in pwd:
        res = long_processing(pwd)
    else:
        res = short_processing(pwd)
    return res

Gain information through timing:

- 0.5 seconds ⇒ no a
- 10 seconds ⇒ a
Side Channel Attacks

```python
def processPassword(pwd):
    if "a" in pwd:
        res = long_processing(pwd)
    else:
        res = short_processing(pwd)
    return res
def processPassword2(pwd):
    if "a" in pwd:
        res = long_processing(pwd)
    else:
        res = long_processing2(pwd)
    return res
```

Gain information through timing:
- 0.5 seconds ⇒ no a
- 10 seconds ⇒ a
def processPassword(pwd):
    if "a" in pwd:
        res = long_processing(pwd)
    else:
        res = short_processing(pwd)
    return res

def processPassword2(pwd):
    if "a" in pwd:
        res = long_processing(pwd)
    else:
        res = long_processing2(pwd)
    return res

Gain information through timing:
- 0.5 seconds $\Rightarrow$ no a
- 10 seconds $\Rightarrow$ a

Gain information execution flow:
- Execute long_processing $\Rightarrow$ a
- Else, no a in pwd
Contributions

1. Show that current countermeasures are not sufficient for cache-based side-channel
Contributions

1. Show that current countermeasures are not sufficient for cache-based side-channel
2. Mount an offline dictionary attack to recover the password
Contributions

1. Show that current countermeasures are not sufficient for cache-based side-channel
2. Mount an offline dictionary attack to recover the password
3. Provide a PoC on Real-World-like scenarios (IWD and FreeRadius)
1. Show that current countermeasures are not sufficient for cache-based side-channel
2. Mount an offline dictionary attack to recover the password
3. Provide a PoC on Real-World-like scenarios (IWD and FreeRadius)
4. Raise awareness on how practical these attacks are
Attacker Model

Client

c0:85:9b
Attacker Model

Client

d8:a3:21

c0:85:9b
Attacker Model

Client

e9:5d:bf

c0:85:9b
Core Idea

AP

WPA3 auth

Victim

Spy process

Trace parsing

Leaked information

Offline dictionary

Remaining passwords
Core Idea

Rogue AP

WPA3 auth

Victim

Spy process

Leaked information

Offline dictionary

Remaining passwords
Core Idea

Rogue AP

Victim

WPA3 auth

Spy process

Leaked information

Offline dictionary attack

Trace parsing

Remaining passwords
Core Idea

Rogue AP → WPA3 auth → Victim

Spy process

Leaked information → Trace parsing

Offline dictionary attack → Remaining passwords

Remaining passwords
Practical Results

Achieve very reliable results with only 10 measurements per MAC address
### Practical Results

<table>
<thead>
<tr>
<th></th>
<th>Dict. size</th>
<th>Cost on AWS</th>
<th>Avg traces for full reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rockyou</td>
<td>$1.4 \cdot 10^7$</td>
<td>0,00037 €</td>
<td>16</td>
</tr>
<tr>
<td>CrackStation</td>
<td>$3.5 \cdot 10^7$</td>
<td>0,0011 €</td>
<td>17</td>
</tr>
<tr>
<td>HavelBeenPwned</td>
<td>$5.5 \cdot 10^8$</td>
<td>0,014 €</td>
<td>20</td>
</tr>
<tr>
<td>8 characters</td>
<td>$4.6 \cdot 10^{14}$</td>
<td>11848,2 €</td>
<td>32</td>
</tr>
</tbody>
</table>

Number of the Required Traces / Cost to Prune all Wrong Passwords
### Practical Results

<table>
<thead>
<tr>
<th></th>
<th>Dict. size</th>
<th>Cost on AWS</th>
<th>Avg traces for full reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rockyou</td>
<td>$1.4 \cdot 10^7$</td>
<td>0,00037 €</td>
<td>16</td>
</tr>
<tr>
<td>CrackStation</td>
<td>$3.5 \cdot 10^7$</td>
<td>0,0011 €</td>
<td>17</td>
</tr>
<tr>
<td>HaveIBeenPwned</td>
<td>$5.5 \cdot 10^8$</td>
<td>0,014 €</td>
<td>20</td>
</tr>
<tr>
<td>8 characters</td>
<td>$4.6 \cdot 10^{14}$</td>
<td>11848,2 €</td>
<td>32</td>
</tr>
</tbody>
</table>

Number of the Required Traces / Cost to Prune all Wrong Passwords
Practical Results

<table>
<thead>
<tr>
<th></th>
<th>Dict. size</th>
<th>Cost on AWS</th>
<th>Avg traces for full reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rockyou</td>
<td>$1.4 \cdot 10^7$</td>
<td>0,00037 €</td>
<td>16</td>
</tr>
<tr>
<td>CrackStation</td>
<td>$3.5 \cdot 10^7$</td>
<td>0,0011 €</td>
<td>17</td>
</tr>
<tr>
<td>HaveIBeenPwned</td>
<td>$5.5 \cdot 10^8$</td>
<td>0,014 €</td>
<td>20</td>
</tr>
<tr>
<td>8 characters</td>
<td>$4.6 \cdot 10^{14}$</td>
<td>11848,2 €</td>
<td>32</td>
</tr>
</tbody>
</table>

Number of the Required Traces / Cost to Prune all Wrong Passwords
FreeRadius to be fixed in 3.0.22

The code is now largely the same between master and v3.0.x, which makes it easier to see that it's correct.
Thank you for your attention!

https://gitlab.inria.fr/ddealmei/poc-iwd-acsac2020
@daniel.de-almeida-braga@irisa.fr
A and B agree on a prime order group $E(\mathbb{F}_p)$, of order $q$

**Dragonfly**

**Alice (A)**

$P \leftarrow p2g(pwd, A, B)$

**Bob (B)**

$P \leftarrow p2g(pwd, A, B)$

Commit

Key derivation

Confirmation

Key derivation
A and B agree on a prime order group $E(\mathbb{F}_p)$, of order $q$

**Dragonfly**

**Alice (A)**

$P \leftarrow p2g(pwd, A, B)$

**Bob (B)**

$P \leftarrow p2g(pwd, A, B)$

**Commit**

**Key derivation**

**Confirmation**

**Key derivation**
Dragonfly - Secret derivation (EC)

HuntingAndPecking(pwd, A, B)

\n\begin{align*}
\text{n, found, } & i \leftarrow \text{len}(p) + 64, \ \text{false, 1} \\
\text{while not found :} & \\
\text{base} & \leftarrow H(\max(A, B) \mid \min(A, B) \mid pwd \mid i) \\
\text{s} & \leftarrow (KDF_n(base, label) \mod (p - 1)) + 1 \\
\text{if } s^3 + as + b \text{ is a quadratic residute mod } p & \\
\text{found, } & x \leftarrow \text{true, s} \\
\text{i} & = i + 1 \\
\text{y} & \leftarrow \sqrt{x^3 + ax + b} \\
\text{if lsb(y) == lsb(base) :} & \\
\text{return (x, y) } & \\
\text{return (x, p - y) } &
\end{align*}
HuntingAndPecking(\(pwd, A, B, k\))

\(n, \text{found, } i \leftarrow \text{len}(p) + 64, \text{false, } 1\)

while not found and \(i < k\) :

\(base \leftarrow H(\text{max}(A, B) \mid \text{min}(A, B) \mid pwd \mid i)\)

\(s \leftarrow (KDF_n(base, \text{label}) \mod (p - 1)) + 1\)

if \(s^3 + as + b\) is a quadratic residure mod \(p\)

if not found :

\(\text{found, } x, \text{base}_{\text{saved}} \leftarrow \text{true, } s, \text{base}\)

\(i = i + 1\)

\(y \leftarrow \sqrt{x^3 + ax + b}\)

if \(\text{lsb}(y) == \text{lsb}(\text{base}_{\text{saved}})\) :

return \((x, y)\)

return \((x, p - y)\)
HuntingAndPecking($pwd$, $A$, $B$, $k$)

$n$, $found$, $i \leftarrow \text{len}(p) + 64$, $\text{false}$, 1

while not $found$ and $i < k$ :
    $base \leftarrow H(\max(A, B) \mid \min(A, B) \mid pwd \mid i)$
    $s \leftarrow (KDF_n(base, label) \mod (p - 1)) + 1$
    if $s^3 + as + b$ is a quadratic residute mod $p$
        if not $found$ :
            $found$, $x$, $basesaved \leftarrow \text{true}$, $s$, $base$
        $i = i + 1$
    $y \leftarrow \sqrt{x^3 + ax + b}$
    if $\text{lsb}(y) == \text{lsb}(basesaved)$ :
        return $(x, y)$
    return $(x, p - y)$
HuntingAndPecking(pwd, A, B, k)

\[ n, \text{found}, i \leftarrow \text{len}(p) + 64, \text{false}, 1 \]

while not found and \( i < k \):

\[ \text{base} \leftarrow H(\text{max}(A, B) | \text{min}(A, B) | \text{pwd} | i) \]
\[ s \leftarrow (KDF_n(\text{base}, \text{label}) \mod (p - 1)) + 1 \]

if \( s^3 + as + b \) is a quadratic residuete mod \( p \)

if not found:

\[ \text{found, x, base}_{\text{saved}} \leftarrow \text{true, s, base} \]
\[ i = i + 1 \]

\[ y \leftarrow \sqrt{x^3 + ax + b} \]

if \( \text{lsb}(y) == \text{lsb}(\text{base}_{\text{saved}}) \):

return \((x, y)\)

return \((x, p - y)\)
HuntingAndPecking(pwd, A, B, k)

\[ n, \text{found}, i \leftarrow \text{len}(p) + 64, \text{false}, 1 \]

while not found and \( i < k \):

\[ \text{base} \leftarrow H(\max(A, B) \mid \min(A, B) \mid pwd \mid i) \]
\[ s \leftarrow (KDF_n(\text{base}, \text{label}) \mod (p - 1)) + 1 \]

if \( s^3 + as + b \) is a quadratic residue modulo \( p \)

\[ \text{if not found} : \]
\[ \text{found, x, base}_{\text{saved}} \leftarrow \text{true, s, base} \]
\[ i = i + 1 \]

\[ y \leftarrow \sqrt{x^3 + ax + b} \]

\[ \text{if lsb}(y) == \text{lsb}(\text{base}_{\text{saved}}) : \]
\[ \text{return (x, y)} \]
\[ \text{return (x, p - y)} \]

\[ \text{: new iteration} \]
\[ \text{: successful conversion} \]