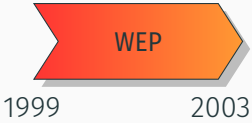
Dragonblood is Still Leaking: Practical Cache-based Side-Channel in the Wild

Daniel De Almeida Braga
Pierre-Alain Fouque
Mohamed Sabt

ACSAC 2020 - December, 10th 2020

Context and Motivations

Context and Motivations



Context and Motivations

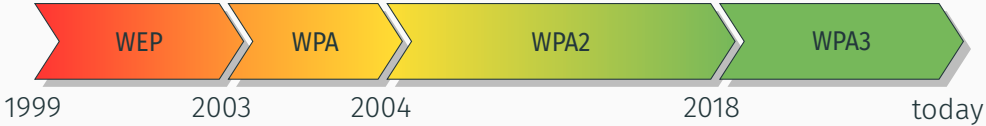
Context and Motivations

Context and Motivations

Offline dictionary attack

KRACK attack

Context and Motivations



+ More secure
+ Based on a PAKE (Dragonfly[1])

[1] D. Harkins, 2015, *Dragonfly Key Exchange*, RFC 7664

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[1] attacks

[1] M. Vanhoef et al. *Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd*. In IEEE S&P. 2020

Side Channel Attacks

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

Side Channel Attacks

```
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 $\Rightarrow$ no a


10 seconds $\Rightarrow$ a

Side Channel Attacks

```
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

Side Channel Attacks

```
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)

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)

4. Raise awareness on how practical these attacks are

Our main result

A cache based side channel attack

let us extract information during

the password conversion with

an offline dictionary attack

Our main result

A cache based side channel attack

let us extract information during

the password conversion with

an offline dictionary attack

Our main result

FLUSH+RELOAD[1] and PDA[2]

Unintended information leakage

A cache based side channel attack

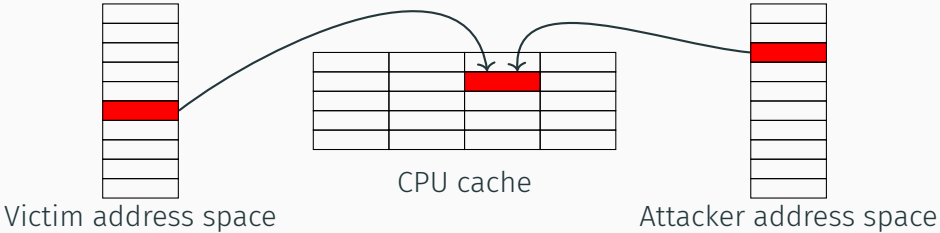
let us extract information during

the password conversion with

an offline dictionary attack

[1] Y. Yarom et al. *Flush+Reload: a High Resolution, Low Noise, L3 Cache Side-Channel Attack.* In USENIX Security Symposium. 2014.
[2] T. Allan et al. *Amplifying side channels through performance degradation.* In ACSAC. 2016

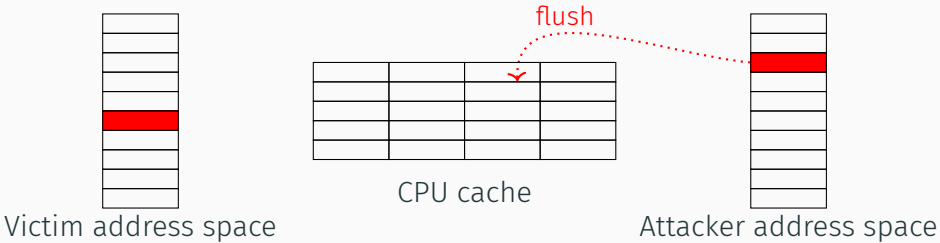
Flush+Reload[1]



1. Maps the victim's address space

[1] Y. Yarom et al. *Flush+Reload: a High Resolution, Low Noise, L3 Cache Side-Channel Attack.* In USENIX Security Symposium. 2014.

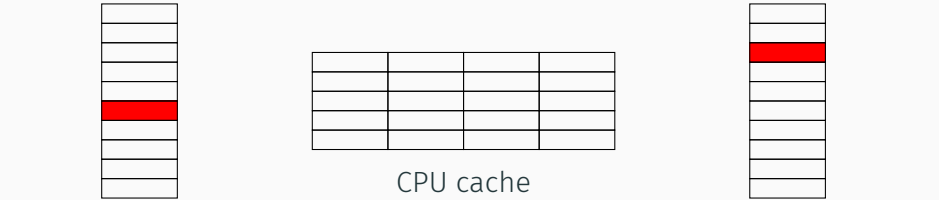
Flush+Reload[1]



1. Maps the victim's address space
2. Flush the instruction we monitor

[1] Y. Yarom et al. *Flush+Reload: a High Resolution, Low Noise, L3 Cache Side-Channel Attack.* In USENIX Security Symposium. 2014.

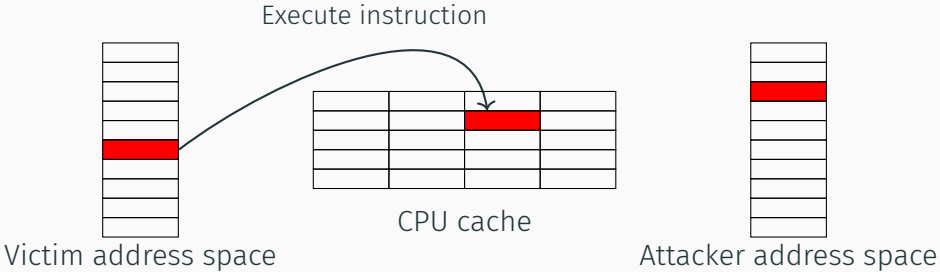
Flush+Reload[1]



1. Maps the victim's address space
2. Flush the instruction we monitor
3. See how much time it takes to reload

[1] Y. Yarom et al. *Flush+Reload: a High Resolution, Low Noise, L3 Cache Side-Channel Attack.* In USENIX Security Symposium. 2014.

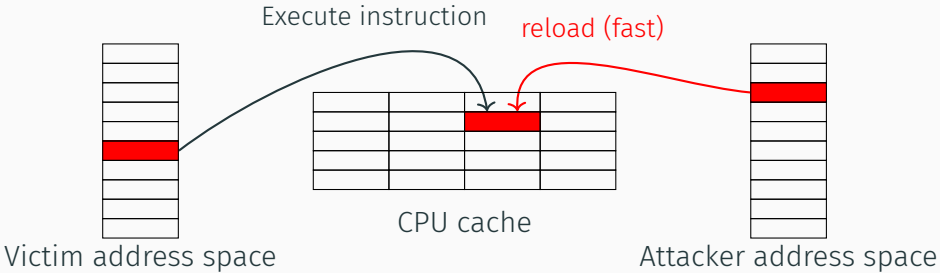
Flush+Reload[1]



1. Maps the victim's address space
2. Flush the instruction we monitor
3. See how much time it takes to reload
   - Fast $\Rightarrow$ the victim already executed

[1] Y. Yarom et al. *Flush+Reload: a High Resolution, Low Noise, L3 Cache Side-Channel Attack.* In USENIX Security Symposium. 2014.

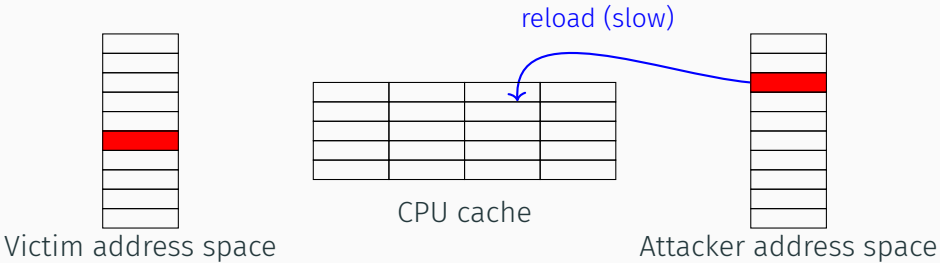
Flush+Reload[1]



1. Maps the victim's address space
2. Flush the instruction we monitor
3. See how much time it takes to reload
   - Fast $\Rightarrow$ the victim already executed

[1] Y. Yarom et al. *Flush+Reload: a High Resolution, Low Noise, L3 Cache Side-Channel Attack.* In USENIX Security Symposium. 2014.

Flush+Reload[1]



1. Maps the victim's address space
2. Flush the instruction we monitor
3. See how much time it takes to reload
   - Fast ⇒ the victim already executed
   - Slow ⇒ the victim did not

[1] Y. Yarom et al. *Flush+Reload: a High Resolution, Low Noise, L3 Cache Side-Channel Attack.* In USENIX Security Symposium. 2014.

Our main result

FLUSH+RELOAD and PDA

Unintended information leakage

A cache based side channel attack let us extract information during the password conversion with an offline dictionary attack

Our main result

FLUSH+RELOAD and PDA

Unintended information leakage

A cache based side channel attack

let us extract information during

the password conversion with

an offline dictionary attack

Password to point on an Elliptic Curve

Our main result

FLUSH+RELOAD and PDA

Unintended information leakage

A cache based side channel attack

let us extract information during

the password conversion with

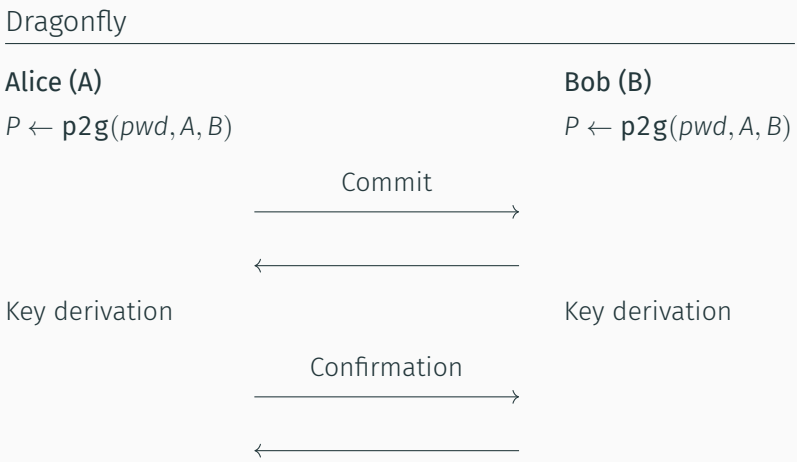
an offline dictionary attack

Password to point on an Elliptic Curve

Passive attacker can eliminate wrong passwords from a list

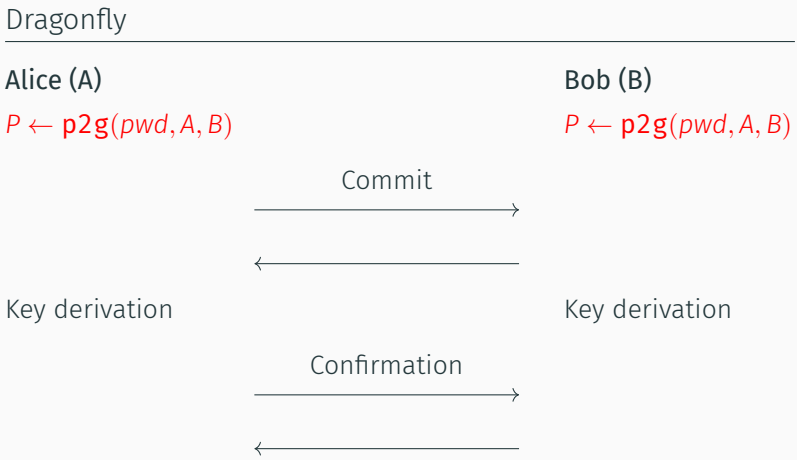
Dragonfly Protocol Overview

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



Dragonfly Protocol Overview

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



Dragonfly - Password Conversion (EC)

HuntingAndPecking(pwd, A, B, k)

1 : $found,\ i = \textbf{false},\ 1$
2 : **while not** $found$ **or** $i < k$:
3 : $\quad x_{cand} = F(A, B, pwd, i)$
4 : $\quad$ **if** x_{cand} is a point's coordinate :
5 : $\quad\quad$ **if not** $found$:
6 : $\quad\quad\quad found,\ x = \textbf{true},\ x_{cand}$
7 : $\quad i\ = i + 1$
8 : $y = \sqrt{x^3 + ax + b}$
9 : **return** (x, y)

Dragonfly - Password Conversion (EC)

HuntingAndPecking(pwd, A, B, k)

1 : $found,\ i = \textbf{false},\ 1$
2 : **while not** $found$ **or** $i < k$:
3 : $\quad x_{cand} = F(A, B, pwd, i)$ ← : new iteration
4 : $\quad$ **if** x_{cand} is a point's coordinate :
5 : $\quad\quad$ **if not** $found$:
6 : $\quad\quad\quad found,\ x = \textbf{true},\ x_{cand}$
7 : $\quad i = i + 1$
8 : $y = \sqrt{x^3 + ax + b}$
9 : **return** (x, y)

Dragonfly - Password Conversion (EC)

HuntingAndPecking(pwd, A, B, k)

1 : $found,\ i = \textbf{false},\ 1$
2 : **while not** *found* **or** $i < k$:
3 : $\quad x_{cand} = F(A, B, pwd, i)$ ← 🕵 : new iteration
4 : $\quad$ **if** x_{cand} is a point's coordinate :
5 : $\quad\quad$ **if not** *found* :
6 : $\quad\quad\quad found,\ x = \textbf{true},\ x_{cand}$ ← 🕵 : successful conversion
7 : $\quad i = i + 1$
8 : $y = \sqrt{x^3 + ax + b}$
9 : **return** (x, y)

Dragonfly - Password Conversion (EC)

HuntingAndPecking(pwd, A, B, k)

1 : $found,\ i = \textbf{false},\ 1$
2 : **while not** $found$ **or** $i < k$:
3 : $\quad x_{cand} = F(A, B, pwd, i)$ ← 🕵 : new iteration
4 : $\quad$ **if** x_{cand} is a point's coordinate : ← PDA: increase leakage
5 : $\quad\quad$ **if not** $found$:
6 : $\quad\quad\quad found,\ x = \textbf{true},\ x_{cand}$ ← 🕵 : successful conversion
7 : $\quad i = i + 1$
8 : $y = \sqrt{x^3 + ax + b}$
9 : **return** (x, y)

Dictionary Reduction

	Iter. required for A, B	Iter. required for A, B’
Leakage	3	
password1		
password2		
password3		
password4		
…		
passwordn		

Dictionary Reduction

	Iter. required for A, B	Iter. required for A, B’
Leakage	3	
password1	1	
password2	3	
password3	3	
password4	4	
…	…	
passwordn	3	

Dictionary Reduction

	Iter. required for A, B	Iter. required for A, B’
Leakage	3	
password1	1	
password2	3	
password3	3	
password4	4	
...	...	
passwordn	3	

Dictionary Reduction

	Iter. required for A, B	Iter. required for A, B’
Leakage	3	2
password1	1	
password2	3	
password3	3	
password4	4	
...	...	
passwordn	3	

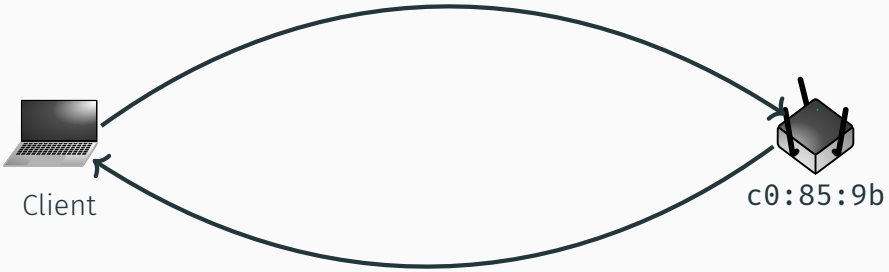
Dictionary Reduction

	Iter. required for A, B	Iter. required for A, B’
Leakage	3	2
password1	1	X
password2	3	8
password3	3	2
password4	4	X
...	...	...
passwordn	3	1

Dictionary Reduction

	Iter. required for A, B	Iter. required for A, B’
Leakage	3	2
password1	1	X
password2	3	8
password3	3	2
password4	4	X
...	...	...
passwordn	3	1

Attacker Model

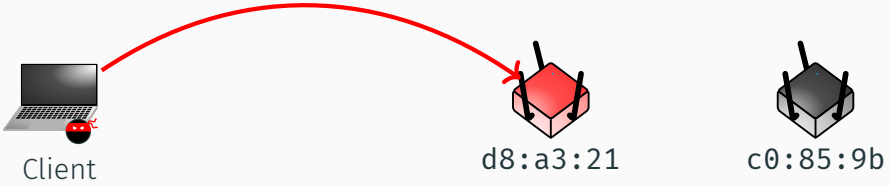


Attacker Model

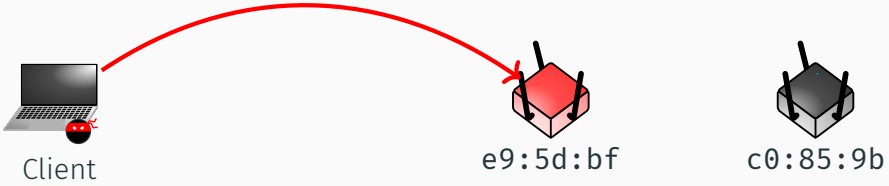
Client

c0:85:9b

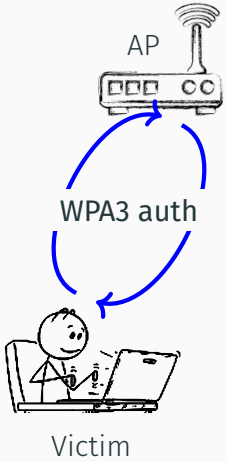
Attacker Model



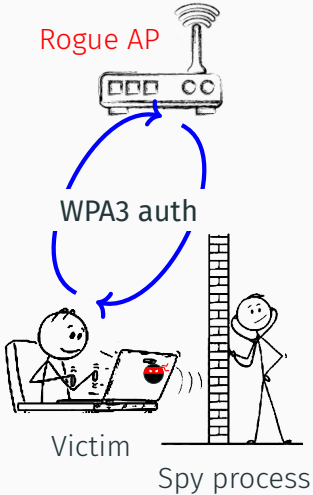
Attacker Model



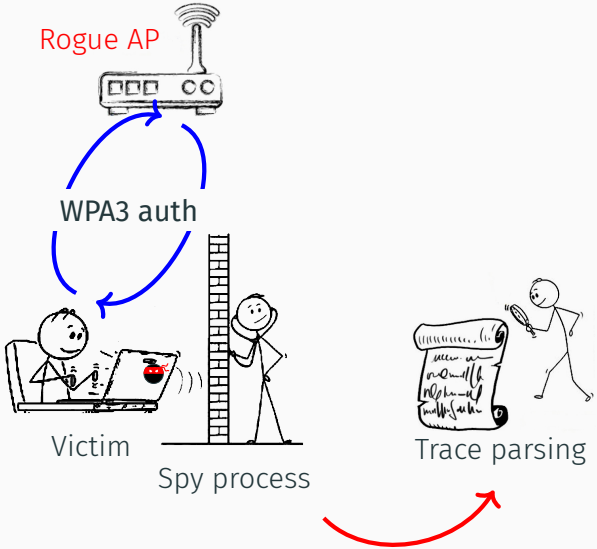
Core Idea



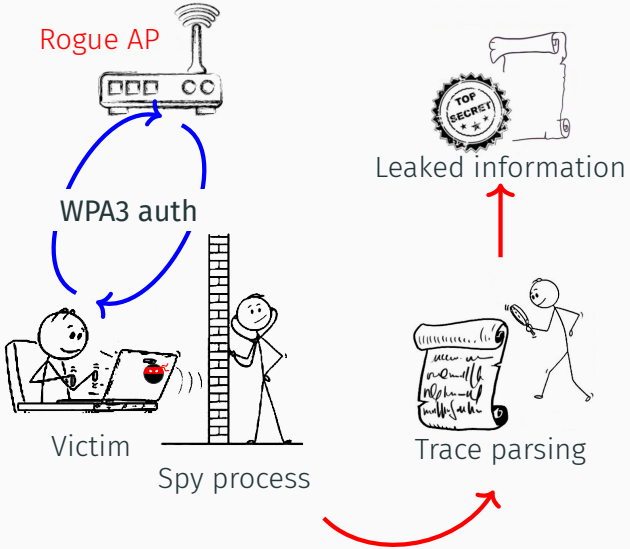
Core Idea



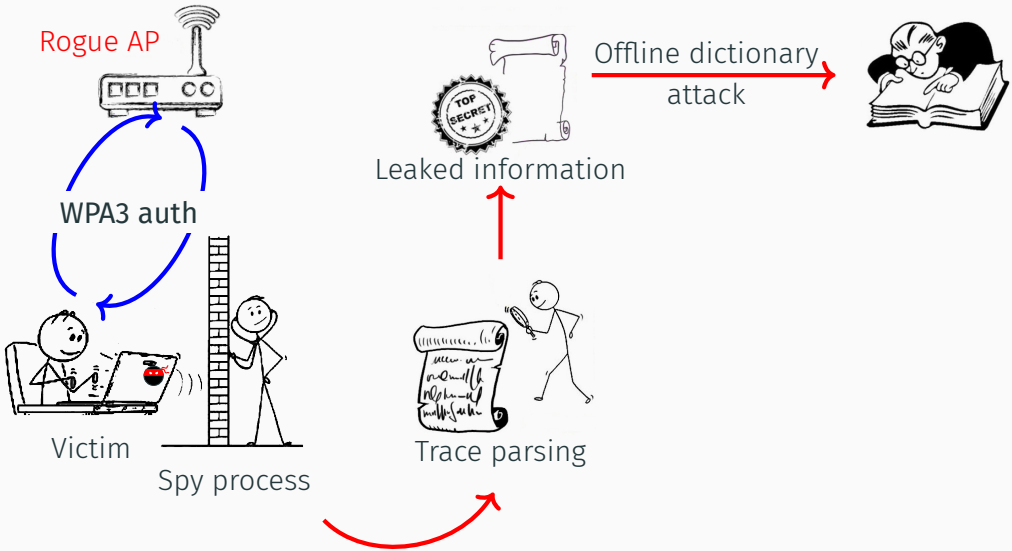
Core Idea



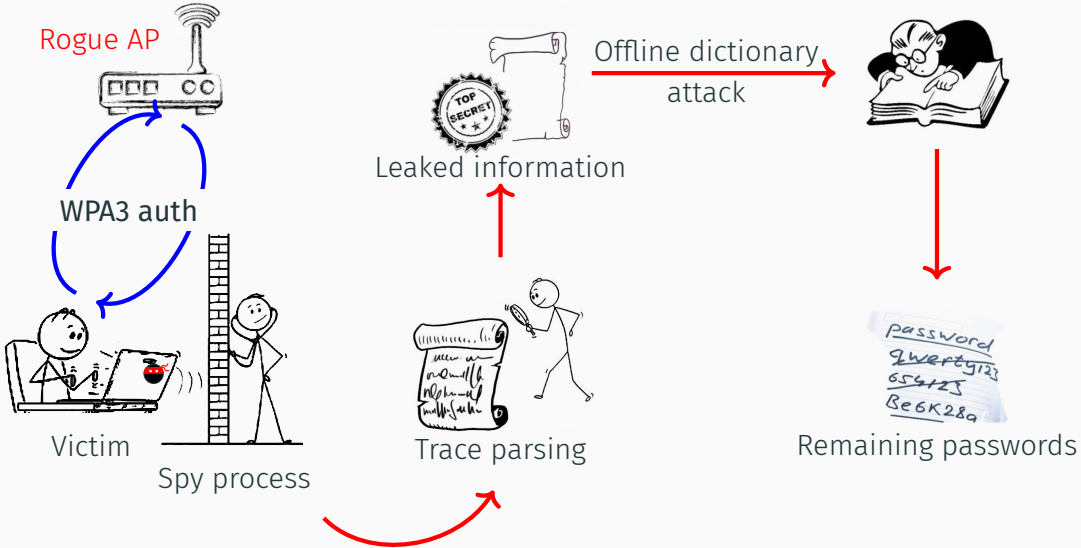
Core Idea


Rogue AP
WPA3 auth
Victim
Spy process
Leaked information
TOP SECRET
Trace parsing

Core Idea

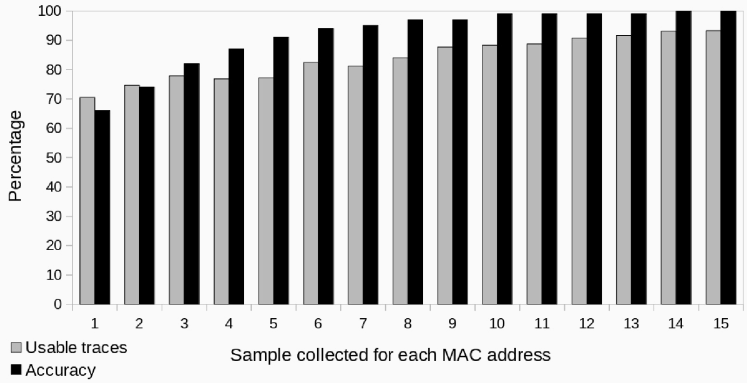

Rogue AP
WPA3 auth
Victim
Spy process
Trace parsing
Leaked information
TOP SECRET
Offline dictionary attack

Core Idea


Rogue AP
WPA3 auth
Victim
Spy process
TOP SECRET
Leaked information
Offline dictionary attack
Trace parsing
password
qwerty123
654123
Remaining passwords

Practical Results

Achieve very reliable results with only 10 measurements per MAC address



Practical Results

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

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

Practical Results

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

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

Practical Results

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

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

Responsible Disclosure

IWD v1.9 ✓

2020-08-03	sae: Fix a side channel leak on the password	Daniel DE ALMEIDA BRAGA	2	-40/+135

FreeRadius to be fixed in 3.0.22

merge constant time fixes from "master" ···

```
Based on a patch from Daniel De Almeida Braga.

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