NetSpectre
Read Arbitrary Memory over Network

Michael Schwarz\textsuperscript{1}
Martin Schwarz\textsuperscript{1}
Moritz Lipp\textsuperscript{1}
Jon Masters\textsuperscript{2}
Daniel Gruss\textsuperscript{1}

\textsuperscript{1}Graz University of Technology \textsuperscript{2}Red Hat
The goal

We want to build a Spectre attack which...

We want to build a Spectre attack which...

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
We want to build a Spectre attack which...

- is capable of leaking secrets from a remote system
We want to build a Spectre attack which...

- is capable of leaking secrets from a remote system
- has neither physical access nor code execution on system
The goal

We want to build a Spectre attack which...

- is capable of leaking secrets from a remote system
- has neither physical access nor code execution on system
- does not rely on software vulnerabilities
CVSS v3 for CVE-2017-5753 (Spectre)

**Attack Vector**

<table>
<thead>
<tr>
<th>Network</th>
<th>Adjacent Network</th>
<th>Local</th>
<th>Physical</th>
</tr>
</thead>
</table>

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
CVSS v3 for CVE-2017-5753 (Spectre)

**Attack Vector**

<table>
<thead>
<tr>
<th>Network</th>
<th>Adjacent Network</th>
<th>Local</th>
<th>Physical</th>
</tr>
</thead>
</table>

**Attack Complexity**

| Low | High |
CVSS v3 for CVE-2017-5753 (Spectre)

### Attack Vector

<table>
<thead>
<tr>
<th>Network</th>
<th>Adjacent Network</th>
<th>Local</th>
<th>Physical</th>
</tr>
</thead>
</table>

### Attack Complexity

- Low
- High

### Privilege Required

- None
- Low
- High
CVSS v3 for CVE-2017-5753 (Spectre)

**Attack Vector**

<table>
<thead>
<tr>
<th>Network</th>
<th>Adjacent Network</th>
<th>Local</th>
<th>Physical</th>
</tr>
</thead>
</table>

**Attack Complexity**

| Low | High |

**Privilege Required**

| None  | Low  | High |

**User Interaction**

| None | Required |
Spectre-PHT (aka Spectre Variant 1)

index = 0

if (index < 4)

then

glyph[data[index]]

else

{}

index = 0

if (index < 4)

then

glyph[data[index]]

else

{}

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
Spectre-PHT (aka Spectre Variant 1)

```
index = 0

if (index < 4)
    glyph[data[index]]
else
    {}
```
```
index = 0

if (index < 4)
    glyph[data[index]]
else
    { }
```
index = 0

if (index < 4)

then

glyph[data[index]]

else

{ }

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
Spectre-PHT (aka Spectre Variant 1)

index = 0

if (index < 4) {
    glyph[data[index]]
} else {
    ...

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
index = 1

if (index < 4)

\begin{align*}
\text{glyph}[\text{data}[\text{index}]]
\end{align*}

\text{else}
index = 1

\[
\text{if } (\text{index} < 4) \\
\text{then}
\]

\[
\text{glyph}[\text{data}[\text{index}]]
\]

\[
\text{else}
\]

\[
\}
\]

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
index = 1

if (index < 4) then

else

Shared Memory

A B
C D E
F G H
I J K
L M N
O P Q
R S T
U V W
X Y Z

Speculate

glyph[data[index]]

Memory

D
data[0]

A

D

DATA

TA

KEY

...
index = 1

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>F</td>
<td>G</td>
</tr>
<tr>
<td>I</td>
<td>J</td>
</tr>
<tr>
<td>L</td>
<td>M</td>
</tr>
<tr>
<td>O</td>
<td>P</td>
</tr>
<tr>
<td>R</td>
<td>S</td>
</tr>
<tr>
<td>U</td>
<td>V</td>
</tr>
<tr>
<td>X</td>
<td>Y</td>
</tr>
</tbody>
</table>

if (index < 4)

then

glyph[data[index]]

else

{ }

data[0]
data[1]
data[2]
data[3]

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
Spectre-PHT (aka Spectre Variant 1)

index = 1

if (index < 4)
  glyph[data[index]]
else
  { }

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
index = 1

if (index < 4) then
    glyph[data[index]]
else

Shared Memory

A B
C D E
F G H
I J K
L M N
O P Q
R S T
U V W
X Y Z

Memory

{ DATA
  data[0]
  data[1]
  data[2]
  data[3]
  KEY...

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
Spectre-PHT (aka Spectre Variant 1)

Shared Memory

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>F</td>
</tr>
<tr>
<td>G</td>
<td>H</td>
</tr>
<tr>
<td>I</td>
<td>J</td>
</tr>
<tr>
<td>K</td>
<td>L</td>
</tr>
<tr>
<td>M</td>
<td>N</td>
</tr>
<tr>
<td>O</td>
<td>P</td>
</tr>
<tr>
<td>Q</td>
<td>R</td>
</tr>
<tr>
<td>S</td>
<td>T</td>
</tr>
<tr>
<td>U</td>
<td>V</td>
</tr>
<tr>
<td>W</td>
<td>X</td>
</tr>
<tr>
<td>Y</td>
<td>Z</td>
</tr>
</tbody>
</table>

Memory

```
index = 2

if (index < 4) {
    glyph[data[index]]
} else {
}
```

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
index = 2

if (index < 4) {
  glyph[data[index]]
} else {

// Speculate

// Memory

// Shared Memory

// T
index = 2

if (index < 4) 
    glyph[data[index]] 
else 
    }

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
index = 2

if (index < 4)
then
glyph[data[index]]

else


index = 2

\[
\text{if (index < 4)} \quad \text{then} \\
\quad \text{glyph}[\text{data}[\text{index}]] \\
\text{else} \quad \}
\]

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
Spectre-PHT (aka Spectre Variant 1)

```
index = 3

if (index < 4)
    glyph[data[index]]
else
    {}
```
index = 3

if (index < 4)
then
glyph[data[index]]

else
{}

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
index = 3

if (index < 4) then
glyph[data[index]]
else
{}

Shared Memory

Memory

data[0]
data[1]
data[2]
data[3]
index = 3

if (index < 4) then
    glyph[data[index]]
else
    {}

index = 3

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
Spectre-PHT (aka Spectre Variant 1)

Shared Memory

```
| A | B |
| C | D | E |
| F | G | H |
| I | J | K |
| L | M | N |
| O | P | Q |
| R | S | T |
| U | V | W |
| X | Y | Z |
```

```javascript
index = 3
if (index < 4)
    glyph[data[index]]
else
```

Memory
```
data[0]
data[1]
data[2]
data[3]
```
Spectre-PHT (aka Spectre Variant 1)

Shared Memory

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>F</td>
</tr>
<tr>
<td>G</td>
<td>H</td>
</tr>
<tr>
<td>I</td>
<td>J</td>
</tr>
<tr>
<td>K</td>
<td>L</td>
</tr>
<tr>
<td>M</td>
<td>N</td>
</tr>
<tr>
<td>O</td>
<td>P</td>
</tr>
<tr>
<td>Q</td>
<td>R</td>
</tr>
<tr>
<td>S</td>
<td>T</td>
</tr>
<tr>
<td>U</td>
<td>V</td>
</tr>
<tr>
<td>W</td>
<td>X</td>
</tr>
<tr>
<td>Y</td>
<td>Z</td>
</tr>
</tbody>
</table>

Memory

\[
\text{if (index < 4)}\]

\[
\text{then}\]

\[
\text{glyph[data[index]]}\]

\[
\text{else}\]

\[
\text{\{}\]

\[
\text{data[0]}\]

\[
\text{data[1]}\]

\[
\text{data[2]}\]

\[
\text{data[3]}\]

\[
\text{\}}\]

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
index = 4

if (index < 4) {
  glyph[data[index]]
} else {
  {}
}
index = 4

if (index < 4) { glyph[data[index]] } else
Spectre-PHT (aka Spectre Variant 1)

```plaintext
index = 4

if (index < 4)
    glyph[data[index]]
else

Shared Memory

A B C D E
F G H I J K
L M N O P Q
R S T U V W
X Y Z

Memory

data[0]
data[1]
data[2]
data[3]

{}
index = 4

if (index < 4)
    glyph[data[index]]
else
    {}

Memory
- data[0]
- data[1]
- data[2]
- data[3]
Spectre without code execution is complicated
Problems

Spectre without code execution is complicated

- Which branch can be exploited
Spectre without code execution is complicated

- Which branch can be exploited
- Cannot observe the cache state
Spectre without code execution is complicated

- Which branch can be exploited
- Cannot observe the cache state
- Spectre gadgets will be different
Spectre without code execution is complicated

- Which branch can be exploited
- Cannot observe the cache state
- Spectre gadgets will be different
- No timing measurement on the attacked system
Spectre without code execution is complicated

- Which branch can be exploited
- Cannot observe the cache state
- Spectre gadgets will be different
- No timing measurement on the attacked system
- How to select the data to leak
• No code can be injected
• No code can be injected
• Public interface (API) accessing data
Exploiting Branches

- No code can be injected
- Public interface (API) accessing data
- Branches in API can be mistrained remotely
Exploiting Branches

- No code can be injected
- Public interface (API) accessing data
- Branches in API can be mistrained remotely
- Attacker only calls the API via network requests
def check_user_privileges(user_id):
    [...]  
    if user_id < len(users):
        if test_bit(privileges, user_id) == True:
            admin = True

    return SUCCESS
def check_user_privileges(user_id):
    
    if user_id < len(users):
        if test_bit(privileges, user_id) == True:
            admin = True
    
    return SUCCESS
def check_user_privileges(user_id):
    ...
    if user_id < len(users):
        if test_bit(privileges, user_id) == True:
            admin = True

    return SUCCESS
def is_admin():
    return admin
def is_admin():
    return admin
def is_admin():
    return admin

- If bit in array was set → admin is cached
def is_admin():
    return admin

• If bit in array was set → admin is cached
• If bit was not set → admin is not cached
API Example

\begin{itemize}
  \item If bit in array was set $\rightarrow$ admin is cached
  \item If bit was not set $\rightarrow$ admin is not cached
  \item Observe cache state via function execution time
\end{itemize}

```python
def is_admin():
    return admin
```

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
• Cannot measure time directly on the attacked system
Timing Measurement

- Cannot measure time directly on the attacked system
- Network latency depends on API execution time
• Cannot measure time directly on the attacked system
• Network latency depends on API execution time
→ Measure the network roundtrip time
Timing Measurement

- Cannot measure time directly on the attacked system
- Network latency depends on API execution time
  → Measure the network roundtrip time
- Reveals whether the variable is cached
Network Measurement
• After measuring variable is always cached
• After *measuring* variable is always *cached*
• How do we *evict* the variable?
• After measuring variable is always cached
• How do we evict the variable?
• Constantly evict the cache via a file download
Resetting Cache State

- After measuring variable is always cached
- How do we evict the variable?
- Constantly evict the cache via a file download
- Thrash+Reload $\rightarrow$ crude form of Evict+Reload
Victim
if (x < bitstream_length)
  if(bitstream[x])
    flag = true
if (x < bitstream_length)
  if (bitstream[x])
    flag = true
if \( x < \text{bitstream\_length} \)

\[
\text{if}(\text{bitstream}[x])
\]

\[
\text{flag} = \text{true}
\]
\begin{verbatim}
if (x < bitstream_length)
    if (bitstream[x])
        flag = true
\end{verbatim}
if (x < bitstream_length)
    if(bitstream[x])
        flag = true
if (x < bitstream_length)
  if (bitstream[x])
    flag = true
    send(flag)
NetSpectre - The Big Picture

\[
\text{if } (x < \text{bitstream\_length}) \\
\text{if} (\text{bitstream}[x]) \\
\text{flag} = \text{true}
\]

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
• **Mistrain** branch predictor with in-bounds requests
• **Mistrain** branch predictor with in-bounds requests
• **Evict** everything from cache via file download
• **Mistrain** branch predictor with in-bounds requests
• **Evict** everything from cache via file download
• **Leak** a bit: do nothing (‘0’) or cache a memory location (‘1’)

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
• **Mistrain** branch predictor with in-bounds requests
• **Evict** everything from cache via file download
• **Leak** a bit: do nothing (‘0’) or cache a memory location (‘1’)
• **Measure** function latency which uses the memory location
Leaking byte ’d’ (0)
Leaking byte ’d’ (01 )
Leaking byte 'd' (011 )
Leaking byte 'd' (0110 )
Leaking byte ’d’ (01100 )
Leaking byte 'd' (011001)
Leaking byte 'd' (0110010)
Leaking byte ‘d’ (01100100)
• Several possible attack targets
• Several possible attack targets
• Different impacts depending on target
• Several possible attack targets
• Different impacts depending on target

Web/FTP Servers
(user gadget)
• Several possible attack targets
• Different impacts depending on target

Web/FTP Servers
(user gadget)

SSH Daemons
(user gadget)
• Several possible attack targets
• Different impacts depending on target

Web/FTP Servers
(user gadget)

SSH Daemons
(user gadget)

Network Drivers
(kernel gadget)
• No indirection, *simple array access*
• No indirection, simple array access

```java
if (x < array_length)
    y = array[x];
```
ASLR Break Idea

0xffffffff600000

vsyscall
gettimeofday

API

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
ASLR Break Idea

0x???????
array[x]

0xffffffff600000
vsyscall
gettimeofday

API

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
ASLR Break Idea

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
ASLR Break Idea

array[x]

0x???????

cached

0xffffffff600000

vsyscall
gettimeofday

API

x

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
ASLR Break Idea

0x???????
array[x]
cached
vsyscall
gmtimeofday
0xfffffffffffffff600000

API

x
ASLR Break Idea

- `array[x]`
- `cached`
- `vsyscall_gettimeofday`
- `0x???????`
- `0xffffffff600000`

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
ASLR Break Idea

0x???????
array\[x\]

cached
gettimeofday

0xffffffff600000

vsyscall
gettimeofday

API

x

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
ASLR Break Idea

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
• 256-bit instructions need a lot of power
• 256-bit instructions need a lot of power
  → On Intel, disabled by default, enabled on first use
• 256-bit instructions need a lot of power
  → On Intel, disabled by default, enabled on first use
• Requires some time to power up
- 256-bit instructions need a lot of **power**
  → On Intel, **disabled by default**, enabled on first use
- Requires some time to power up
- Measure execution time of AVX instruction
- 256-bit instructions need a lot of **power**
  → On Intel, **disabled by default**, enabled on first use
- Requires some time to **power up**
- Measure execution time of AVX instruction
  → **Leak** timing information
AVX Latency

Latency [cycles]

Cases

- Powered down
- Warmed up

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
if (x < bitstream_length)
    if(bitstream[x])
        _mm256_instruction();
AVX Cooldown

Wait time [$\mu$s]

Latency

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
1. **Mistrain** branch predictor with in-bounds requests
1. **Mistrain** branch predictor with in-bounds requests
2. **Wait** for AVX unit to power off (1ms)
1. **Mistrain** branch predictor with in-bounds requests
2. **Wait** for AVX unit to power off (1ms)
3. **Leak** a bit: do nothing (‘0’) or power AVX unit (‘1’)

Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, Daniel Gruss
AVX-based NetSpectre

1. Mistrain branch predictor with in-bounds requests
2. Wait for AVX unit to power off (1ms)
3. Leak a bit: do nothing (‘0’) or power AVX unit (‘1’)
4. Measure function latency which uses AVX instruction
Response time [CPU cycles]

Latency

- Powered down unit
- Warmed up unit
• **Local** Network  
  (1,000,000 measurements/bit)
• **Local Network**  (1 000 000 measurements/bit)

30 min/byte
• **Local Network**  (1,000,000 measurements/bit)

30 min/byte

8 min/byte
Attack Results

- **Local Network**  (1 000 000 measurements/bit)
  - 30 min/byte

- **Cloud**  (20 000 000 measurements/bit)
  - 8 min/byte
Attack Results

- **Local Network** (1,000,000 measurements/bit)
  - 30 min/byte
- **Cloud** (20,000,000 measurements/bit)
  - 1 h/bit
  - 8 min/byte
• NetSpectre requires a fast and stable network connection
• NetSpectre requires a fast and stable network connection
  • Local networks
• NetSpectre requires a **fast and stable network** connection
  - Local networks
  - Data centers (VM to VM attack)
Limitations

- NetSpectre requires a **fast and stable network connection**
  - Local networks
  - Data centers (VM to VM attack)
- Internet speeds improve (e.g., fiber, 5G)
Limitations

• NetSpectre requires a fast and stable network connection
  • Local networks
  • Data centers (VM to VM attack)
• Internet speeds improve (e.g., fiber, 5G)
→ possible in the near future?
Limitations

- NetSpectre requires a fast and stable network connection
  - Local networks
  - Data centers (VM to VM attack)
- Internet speeds improve (e.g., fiber, 5G)
  → possible in the near future?
- Attack speeds can be drastically improved
Limitations

- NetSpectre requires a **fast and stable network connection**
  - Local networks
  - Data centers (VM to VM attack)
- Internet speeds improve (e.g., fiber, 5G)
  → possible in the **near future**?
- Attack speeds can be drastically **improved**
  - Better signal processing/filtering
Limitations

- NetSpectre requires a fast and stable network connection
  - Local networks
  - Data centers (VM to VM attack)
- Internet speeds improve (e.g., fiber, 5G)
  → possible in the near future?
- Attack speeds can be drastically improved
  - Better signal processing/filtering
  - Dedicated measuring hardware
• Speculative execution leaks secrets without exploiting bugs
• Spectre attacks are not limited to local attackers
• Spectre attacks have a larger impact than assumed
NetSpectre
Read Arbitrary Memory over Network

Michael Schwarz
Martin Schwarz
Moritz Lipp
Jon Masters
Daniel Gruss

\(^1\)Graz University of Technology \(^2\)Red Hat