Evading Suricata IDS

Researching evasions for the SMB protocol

Bastien Del-Valle, Louis Jacotot

TELECOM Nancy

$ whoami

Graduate engineers from TELECOM Nancy

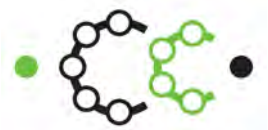
Bastien Del-Valle

- Now : Security expert
- Threat research & analysis

Louis Jacotot

- Now : Security researcher @Synacktiv
- Vulnerability research & exploitation
- French offensive security company
- ~ 90 ninjas, hiring !

Our research project



TELECOM Nancy

- French IT engineering school
- Research project in second year (early 2020)

”Evading Suricata IDS”

- Supervised by Philippe Antoine from Catena cyber
- Suricata developer & teacher at TELECOM Nancy

Evasions

General definition

- Techniques that prevent the detection of attacks

Application to NIDS

- Network traffic matches the semantic of a rule...
- ... but the NIDS does not raise an alert

Then ?

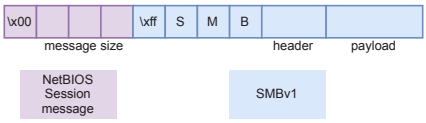
- Fewer logs in case of an incident
- At least simple evasions could be mitigated

SMB protocol

Server Message Block

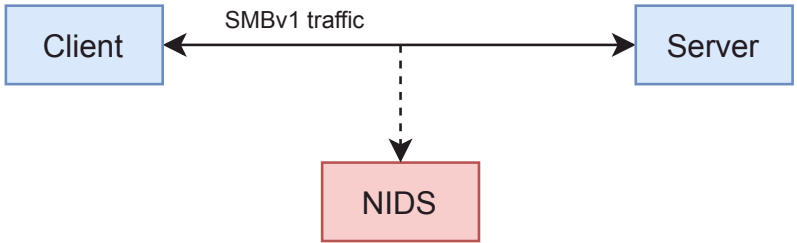
- Share resources (files, printers...), IPC
- MS-CIFS, MS-SMB (SMBv1)
- MS-SMB2 (SMBv2, SMBv3)
- Microsoft SMB, Samba
- CVE-2017-0144, CVE-2020-0796...

SMBv1 over TCP (445)



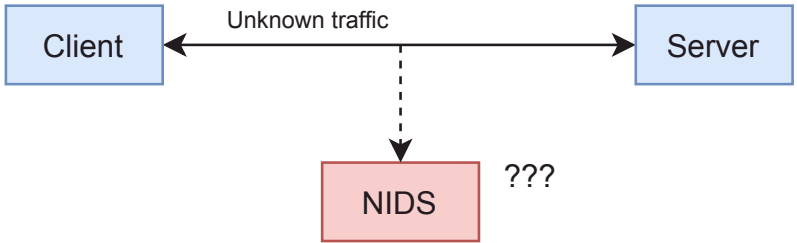
- Requests & responses
- ”AndX” commands can be chained

Threat model



- Client or server performs evasions

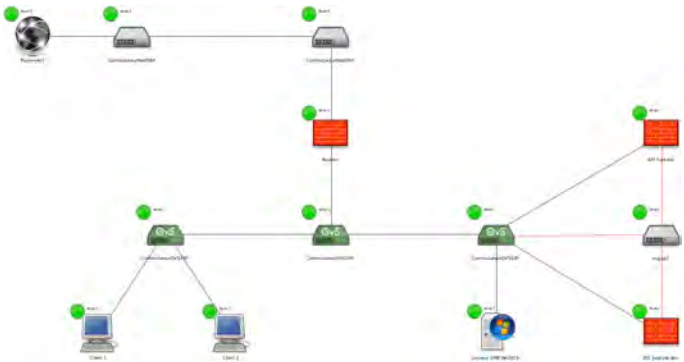
Threat model



- Client or server performs evasions, **not both**

Methodology

- Cyber range
- **SMBv1 over TCP**
- **Client-side evasions**
- Debian client
- Windows Server 2019
- **Suricata 5.0.x**



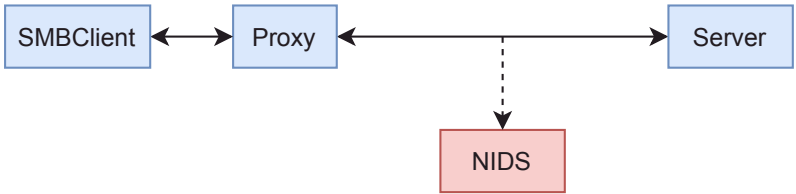
Proxy

How ?

- Developing a SMB client from scratch

How ?

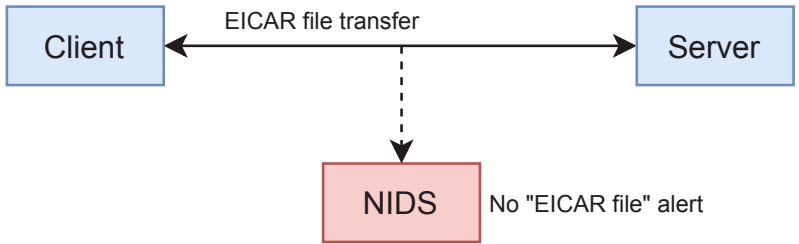
- ~~Developing a SMB client from scratch~~
- Altering SMB traffic ”on the fly” with a proxy



File transfer evasions

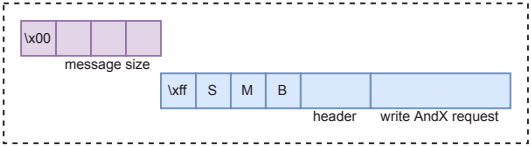
- EICAR test file

```
alert smb any any -> any any (msg:"EICAR file"; flow:established; file_data;
        content: "|58354f2150254041505b345c505a58353428505e2937434329377d244549
                  4341522d5354414e444152442d414e544956495255532d544553542d4649
                  4c452124482b482a|";
        sid:1; rev:1;)
```

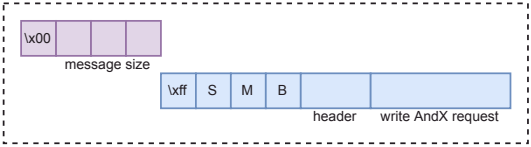


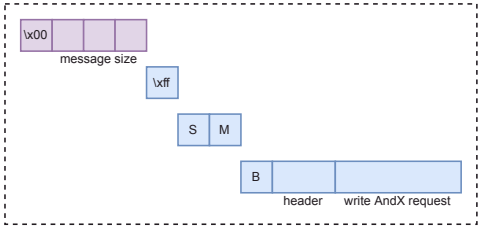
File transfer evasions : Fragmentation

File transfer evasions : Fragmentation

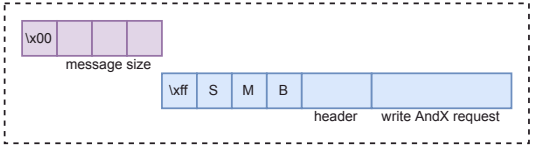


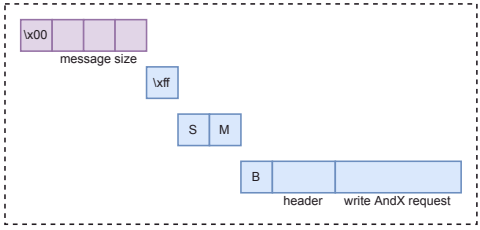
File transfer evasions : Fragmentation

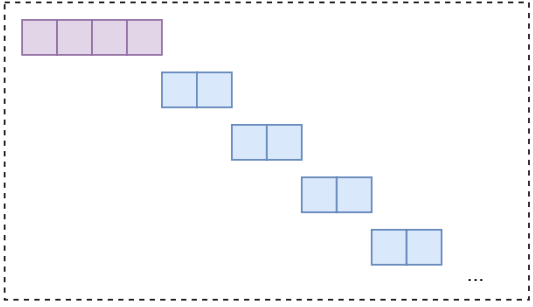




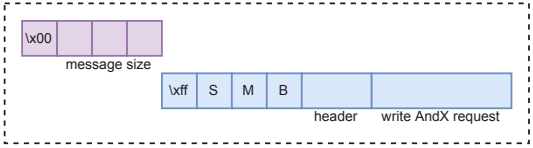
File transfer evasions : Fragmentation

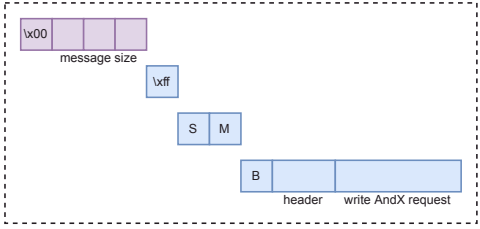


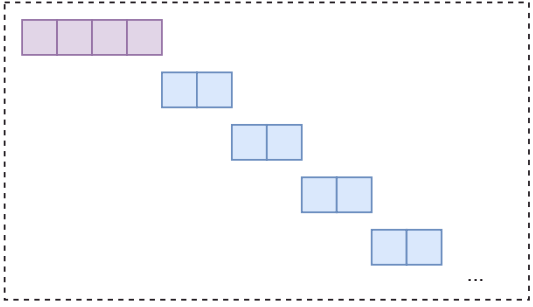




File transfer evasions : Fragmentation

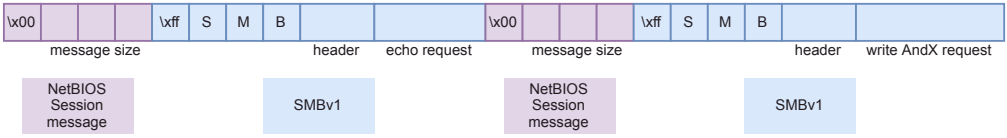






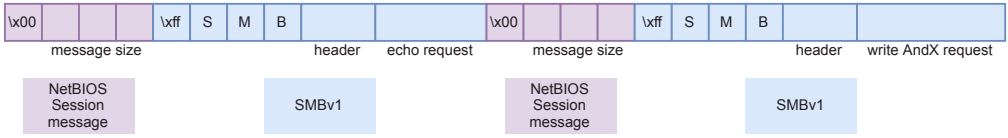
- No evasion

File transfer evasions : Two NetBIOS messages in one TCP packet




```
Transmission Control Protocol, Src Port: 48068, Dst Port: 445, Seq: 757, Ack: 951, Len: 193
NetBIOS Session Service
SMB (Server Message Block Protocol)
 ⊞ SMB Header
 ⊞ Echo Request (0x2b)
NetBIOS Session Service
SMB (Server Message Block Protocol)
 ⊞ SMB Header
 ⊞ Write AndX Request (0x2f)
```

File transfer evasions : Two NetBIOS messages in one TCP packet



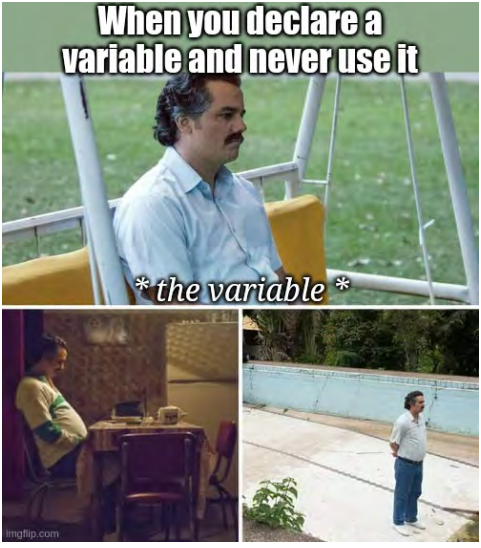

```
Transmission Control Protocol, Src Port: 48068, Dst Port: 445, Seq: 757, Ack: 951, Len: 193
NetBIOS Session Service
SMB (Server Message Block Protocol)
 ⊞ SMB Header
 ⊞ Echo Request (0x2b)
NetBIOS Session Service
SMB (Server Message Block Protocol)
 ⊞ SMB Header
 ⊞ Write AndX Request (0x2f)
```

- No evasion

File transfer evasions : AndX batched messages

- Unused variables

```
1  do_parse!(i,
2            wct:           le_u8
3         >>  _andx_command: le_u8
4         >>  take!(1)       // reserved
5         >>  _andx_offset:  le_u16
6         //  [...]
```



File transfer evasions : AndX batched messages

- Unused variables

```
do_parse!(i,
          wct:           le_u8
      >>  _andx_command: le_u8
      >>  take!(1)       // reserved
      >>  _andx_offset:  le_u16
      //  [...]
```

- Idea : chaining a dummy AndX command to write requests

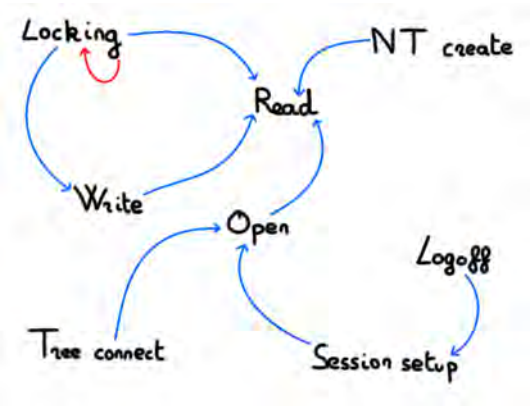
File transfer evasions : AndX batched messages

- Unused variables

```
1  do_parse!(i,
2            wct:            le_u8
3        >>  _andx_command:  le_u8
4        >>  take!(1)        // reserved
5        >>  _andx_offset:   le_u16
6        //  [...]
```

- Idea : chaining a dummy AndX command to write requests

- Valid chains :



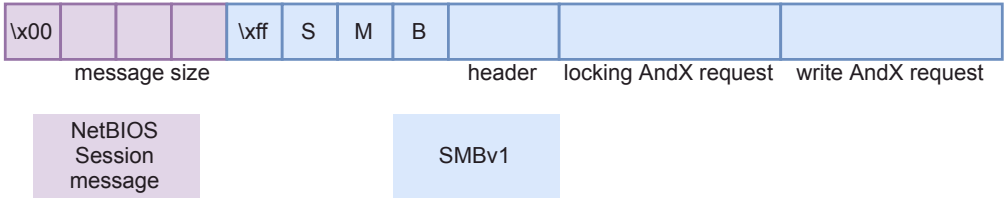
File transfer evasions : AndX batched messages

NetBIOS Session message

SMBv1

```
Transmission Control Protocol, Src Port: 37488, Dst Port: 445, Seq: 757, Ack: 951, Len: 155
NetBIOS Session Service
SMB (Server Message Block Protocol)
 SMB Header
 Locking AndX Request (0x24)
 Write AndX Request (0x2f)
```

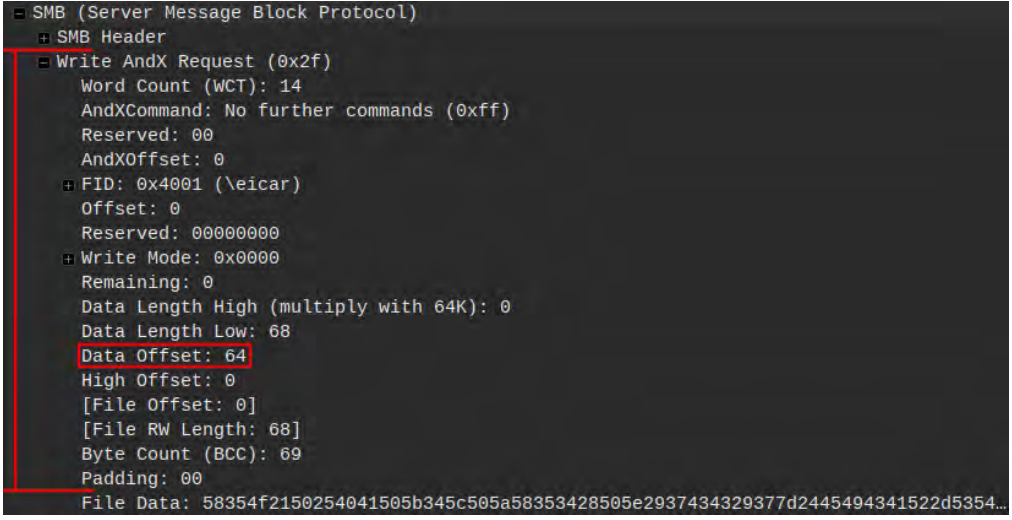
File transfer evasions : AndX batched messages



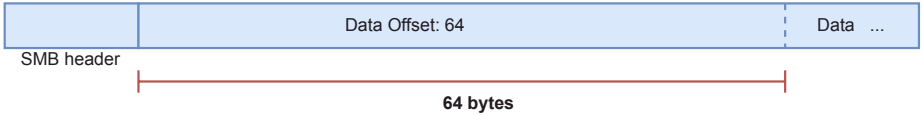

```
Transmission Control Protocol, Src Port: 37488, Dst Port: 445, Seq: 757, Ack: 951, Len: 155
NetBIOS Session Service
SMB (Server Message Block Protocol)
 SMB Header
 Locking AndX Request (0x24)
 Write AndX Request (0x2f)
```

- Evasion !

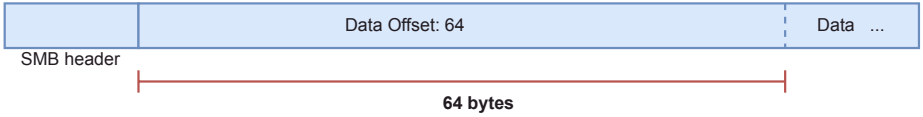
File transfer evasions : Data padding

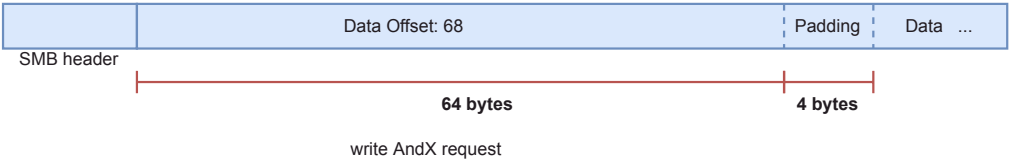


File transfer evasions : Data padding



File transfer evasions : Data padding





File transfer evasions : Data padding

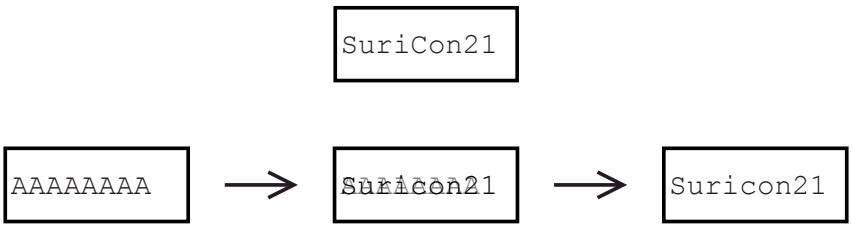
```
Write AndX Request (0x2f)
  Word Count (WCT): 14
  AndXCommand: No further commands (0xff)
  Reserved: 00
  AndXOffset: 0
⊞ FID: 0x400e (\eicar)
  Offset: 0
  Reserved: 00000000
⊞ Write Mode: 0x0000
  Remaining: 0
  Data Length High (multiply with 64K): 0
  Data Length Low: 68
  Data Offset: 68
  High Offset: 0
  [File Offset: 0]
  [File RW Length: 68]
  Byte Count (BCC): 69
  Padding: 00
  File Data: 0000000058354f2150254041505b345c505a58353428505e2937434329377d2445494341…
```

File transfer evasions : Data padding

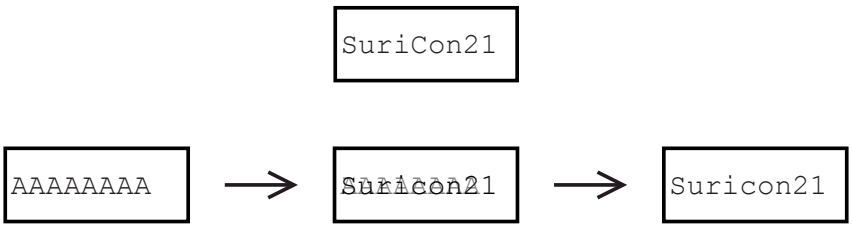
- Evasion
- Wireshark dissector parsing error

File transfer evasions : Data overwriting

File transfer evasions : Data overwriting



File transfer evasions : Data overwriting



- Missing data overwriting support
- Allows to write the EICAR file without detection
- Evasion

Named pipes evasions

- Suricata keywords
- ”smb_named_pipe”
- Tree connect AndX request

```
alert smb any any -> any any (msg:"IPC$ named pipe"; flow:established;
        smb_named_pipe; content:"IPC$";
        sid:1; rev:1;)
```

Named pipes evasions : Encoding

- Header ”unicode” flag

Named pipes evasions : Encoding

- Header ”unicode” flag
- Idea : flip its value and change the named pipe encoding

```
1 00000000  5c 5c 31 39 32 2e 31 36  38 2e 31 2e 31 5c 49 50  |\\192.168.1.1\IP|
2 00000010  43 24 00                                          |C$.|
```

Named pipes evasions : Encoding

- Header ”unicode” flag
- Idea : flip its value and change the named pipe encoding

```
1 00000000  5c 5c 31 39 32 2e 31 36  38 2e 31 2e 31 5c 49 50  |\\192.168.1.1\IP|
2 00000010  43 24 00                                          |C$.|
```

```
1 00000000  5c 00 5c 00 31 00 39 00  32 00 2e 00 31 00 36 00  |\.\.1.9.2...1.6.|
2 00000010  38 00 2e 00 31 00 2e 00  31 00 5c 00 49 00 50 00  |8...1...1.\.I.P.|
3 00000020  43 00 24 00 00 00                                 |C.$...|
```

Named pipes evasions : Encoding

- Header ”unicode” flag
- Idea : flip its value and change the named pipe encoding

```
1 00000000  5c 5c 31 39 32 2e 31 36  38 2e 31 2e 31 5c 49 50  |\\192.168.1.1\IP|
2 00000010  43 24 00                                          |C$.|
```

```
1 00000000  5c 00 5c 00 31 00 39 00  32 00 2e 00 31 00 36 00  |\.\.1.9.2...1.6.|
2 00000010  38 00 2e 00 31 00 2e 00  31 00 5c 00 49 00 50 00  |8...1...1.\.I.P.|
3 00000020  43 00 24 00 00 00                                 |C.$...|
```

- No evasion

Conclusion

Results

- **Proxy rules** to test evasions ”on the fly”
- **3 SMBv1 evasions** found & reported
- Network PCAP for each technique
- **Suricata-verify tests**

Next

- More evasions ?
- Server-side evasions, SMBv2, SMBv3
- Other protocols

Do you have any questions ?

Thank you for your attention

TELECOM Nancy

Ingénieurs du numérique • Inspiring your digital future