Fuzzing: The SMB case

Laurent Gaffié, stratsec

HackitoErgoSum, Paris, April 2010

stratsec

Agenda

- What is SMB
- How It Works
- Why fuzzing SMB ?
- Approach
- Demo
- Bug discovered in client side
- Bugs discovered in server side
- Client vs Server
- Questions

What is SMB ?

Critical Windows networking component.

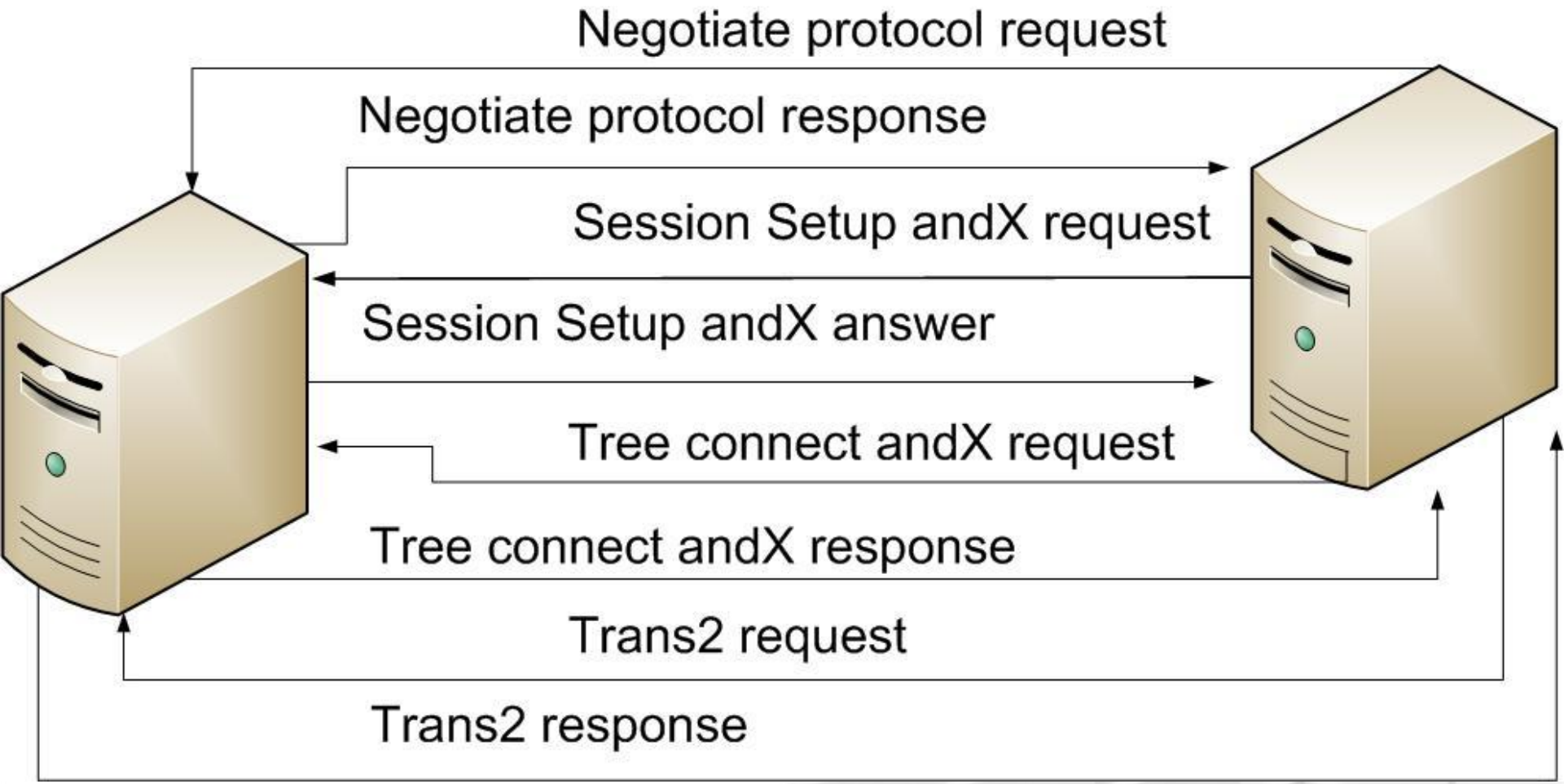
Can be used over: TCP/IP, IPX/SPX, and NetBEUI

A protocol for printers, file sharing, serial ports

A Transport layer for DCE/RPC/IPC

How It Works

How It Works


Negotiate protocol request
Negotiate protocol response
Session Setup andX request
Session Setup andX answer
Tree connect andX request
Tree connect andX response
Trans2 request
Trans2 response

How It Works

A SMB packet is composed by 3 parts:

NBT length, SMB Header, SMB Command

SMB Header will contain theses fields:

Flags, Flags2, UID, TID, SIG, PID, MID, etc

SMB Command will contain the specified command parameter + the data (SMB-Data)

How It Works

SMB Header structure, Flags & Flags2 fields :

```
⊟ Flags: 0x18
    0... .... = Request/Response: Message is a request to the server
    .0.. .... = Notify: Notify client only on open
    ..0. .... = Oplocks: OpLock not requested/granted
    ...1 .... = Canonicalized Pathnames: Pathnames are canonicalized
    .... 1... = Case Sensitivity: Path names are caseless
    .... ..0. = Receive Buffer Posted: Receive buffer has not been posted
    .... ...0 = Lock and Read: Lock&Read, Write&Unlock are not supported
⊟ Flags2: 0xc853
    1... .... .... .... = Unicode Strings: Strings are Unicode
    .1.. .... .... .... = Error Code Type: Error codes are NT error codes
    ..0. .... .... .... = Execute-only Reads: Don't permit reads if execute-only
    ...0 .... .... .... = Dfs: Don't resolve pathnames with Dfs
    .... 1... .... .... = Extended Security Negotiation: Extended security negotiation is supported
    .... .... .1.. .... = Long Names Used: Path names in request are long file names
    .... .... .... .0.. = Security Signatures: Security signatures are not supported
    .... .... .... ..1. = Extended Attributes: Extended attributes are supported
    .... .... .... ...1 = Long Names Allowed: Long file names are allowed in the response
```

How It Works

Theses fields Indicates the client's capabilities:

- Oplocks, batch lock, etc
- Unicode strings support
- Long path name support
- Uppercase path
- etc.

Theses fields are set each time in SMB header

How It Works

Each SMB function use it's own parameter.

Most often used parameters:

- BCC (byte count)
- Word count
- AndXOffset (when using andX)
- Max data count
- Data offset
- Total parameter count
- Parameter offset
- Etc.

How It Works

An SMB packet will contain at least 2 length check defined in the packet including a BCC

The packet will be discarded if string > BCC-length

The packet will NOT be discarded if string < BCC-length

Why fuzzing SMB ?

Available since Windows 3.1 for Workgroups

Server and client run with kernel privilege

Considerably modified between Windows releases

Why fuzzing SMB ?

It’s older than 26 years old !

first documentation from IBM is dated 1984

Assumed to be secure, but poorly audited.

It’s fast !

Many linux distribution and Mac OSX use Samba.

Approach

two **steps methodology**

Approach

Step 1:

Research RFC's, books, MS specifications.

Documentation acounts for 90% of the fuzzing process.

Approach

Step 2:

Build a lab reproducing a corporate network including all Windows :

- Windows 3.11 for Workgroups to Windows 7
- Samba
- OS/2
- etc.

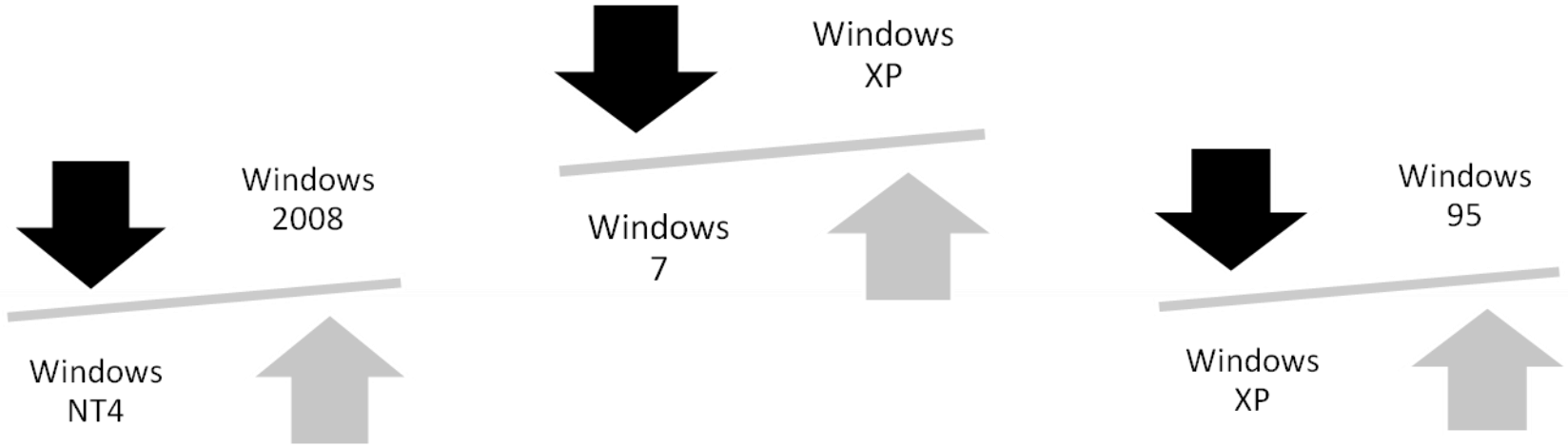
Approach

Set an Active Directory environment in the lab, with clients running :

- Windows 98
- Windows NT4
- Windows XP
- Windows 7

Approach

Capture SMB communication between different Windows versions and build a pcap database:



Approach

Variation SMB messaging occurs when:

Different OS versions are used as the SMB server/client

Hosts are configured to use an Active Directory service.

Approach

Targeting retro-compatibility is a key to SMB fuzzing

It allows users to play with old functions that are still supported but not used anymore

Approach

Do not use an SMB framework/lib.

Build your own complete SMB client.

Create a specially dedicated fuzzing library for the client, and update it frequently.

Approach

Independently fuzz XP, 2003, Vista, 7, 2008 systems. Preferably on physical machines.

Do a first round fuzzing, unstructured (dumb fuzzing) for each :

- Functions
- Opcode
- Target one SMB function at the time.

Approach

The targeted function packets are usually small, considering the fuzzer's speed

4 minutes per function is more than enough to catch the first round bugs.

Results are always surprising ...

Demo

DEMO

Bugs discovered in client side

Windows Vista/2008 (only)

SMB DCE/RPC PIPE Null Pointer Deref (CVE-2010-0476)

Windows 7/2008R2 (only)

Trans2 Buffer Overflow (CVE-2010-0270)

Windows 7/2008R2 (only)

NBT length infinite loop (CVE-2009-3676)

Bugs discovered in client side

Windows 7, Vista

Session setup SMBv2 DOS (CVE-2010-0477)

Windows 2000, XP

Negotiate Protocol Pool Overflow (CVE-2010-0016)

Windows 7, Vista

Negotiate Protocol Race Condition (CVE-2010-0017)

Bugs discovered in client side

Windows XP

SMB/NetBT TDI client DoS

Bugs discovered in server side

Windows Vista, 7 RC

SMBv2 negotiate protocol overflow (CVE-2009-3103)

Windows 2000, XP, Vista, 7

Pool overflow

Samba

Session Setup Null pointer deref

Bugs discovered in server side

Samba

Session Setup AndX Uninitialized variable read DoS

Netware 6.5 SP8

Session Setup AndX username Stack Overflow

Bugs discovered

These bugs have been found via first round fuzzing using a user assisted approach.

Each of these bugs took less than two minutes to discover while fuzzing.

Client vs Server

Server side fuzzing is pretty fast.

Client side fuzzing is considerably slower.

Client vs Server

SMBv1 contain about 100 different commands.

Fuzzing both client and server increases the possible results.

Client side vulnerabilities can be triggered transparently via IE, NBNS spoofing, browser cache poisoning, etc.

Client vs Server

Server side vulnerabilities do not require user interaction and can often be exploited without authentication.

As a result vulnerabilities can be targeted by worms.

Client vs Server

Client side vulnerabilities never require user authentication, and could be triggered in some case with no user interaction

Questions ? (=

Contact: laurent{dot}gaffie{at}stratsec{dot}net