Securitybyte
Securing the information DNA

OWASP
AppSec Asia

How to Blackbox Test Almost Anything

Aviram Jenik
CEO
Beyond Security

Who am I?

- CEO of Beyond Security:
  - We develop automated security testing tools:
    - Network vulnerability assessment/management
    - Automated Web Site Security Scans
    - Blackbox testing/fuzzing
  - We operate and maintain SecuriTeam.com
    - One of the largest vulnerability databases on the net
    - Publish vulnerability information and exploit code
    - Open and free
  - SecuriTeam Secure Disclosure
    - Paying researchers who find 0-day vulnerabilities
    - Giving customers early-access to this information

So what do I do?

- I've been doing security for 23 years
- Focusing on Vulnerability research/security testing

So what do I do?

- I've been doing security for 23 years
- Focusing on Vulnerability research/security testing

I like to break things

Is security testing important?

5 random security holes on SecuriTeam.com:

- Excel code execution (memory corruption via malformed XLS file)
- Windows License logging service code execution (heap corruption, via remote RPC call)
- Atheros Driver DoS Vulnerability (can crash the wifi driver via a malformed network packet)
- McAfee Security Manager Authentication Bypass (can bypass authentication via cross-site-scripting attack)
- Novell eDirectory LDAP Null Base DN DoS (DoS attack on Novell eDirectory via unauthenticated

Is security testing important?

Microsoft Office Excel Code Execution Vulnerabilities 13 Nov. 2009

Summary

Attackers using specially crafted XLS files can execute arbitrary code via memory corruptions, invalid index, and invalid pointer errors.

Credit:
The information has been provided by Nicolas JOLY.

Microsoft Windows License Logging Service Heap Corruption Vulnerability 13 Nov. 2009

Summary

This vulnerability allows remote attackers to execute arbitrary code on vulnerable installations of Microsoft Windows. Authentication is not required on certain configurations to exploit this vulnerability.

Novell eDirectory LDAP Null Base DN DoS Vulnerability 4 Nov. 2009

Summary

This vulnerability allows attackers to deny services on vulnerable installations of Novell eDirectory. Authentication is not required in order to exploit this vulnerability.

Credit:
The original article can be found at: http://www.zerodayinitiative.com/advisories/ZDI-09-075

Atheros Driver Reserved Frame DoS Vulnerability 13 Nov. 2009

Summary

The wireless driver in some Wi-Fi access points (such as the ATHEROS-based Netgear WNDAP330) do not correctly parse malformed reserved management frames.

McAfee Security Manager Authentication Bypass and Session Hijacking Vulnerability 13 Nov. 2009

Summary

McAfee Network Security Manager is vulnerable to authentication bypass via HTTP session cookie hijacking. A remote attacker could exploit this vulnerability to hijack an existing session to the Network Security Manager.

What do they have in common?

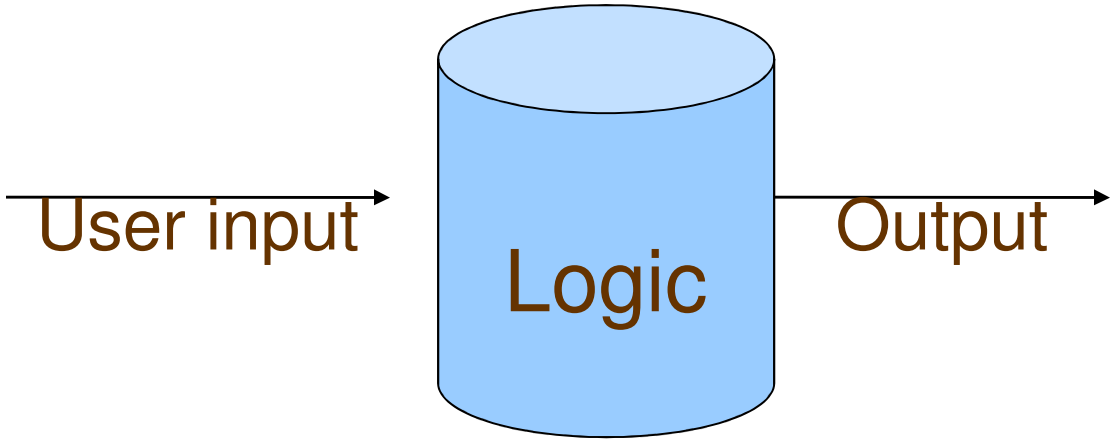
- 5 different products
- Different attack vectors
- All critical vulnerabilities
- All require patch
- Unpatched systems will be extremely vulnerable
- All could have been discovered during development
- None requires special expertise to exploit (hence, relatively straightforward to discover)
- (probably) found via fuzzing – e.g. blackbox testing

File input (XLS)
Network (RPC)
Physical (802.11 frame)
Web (XSS)
Network (LDAP)

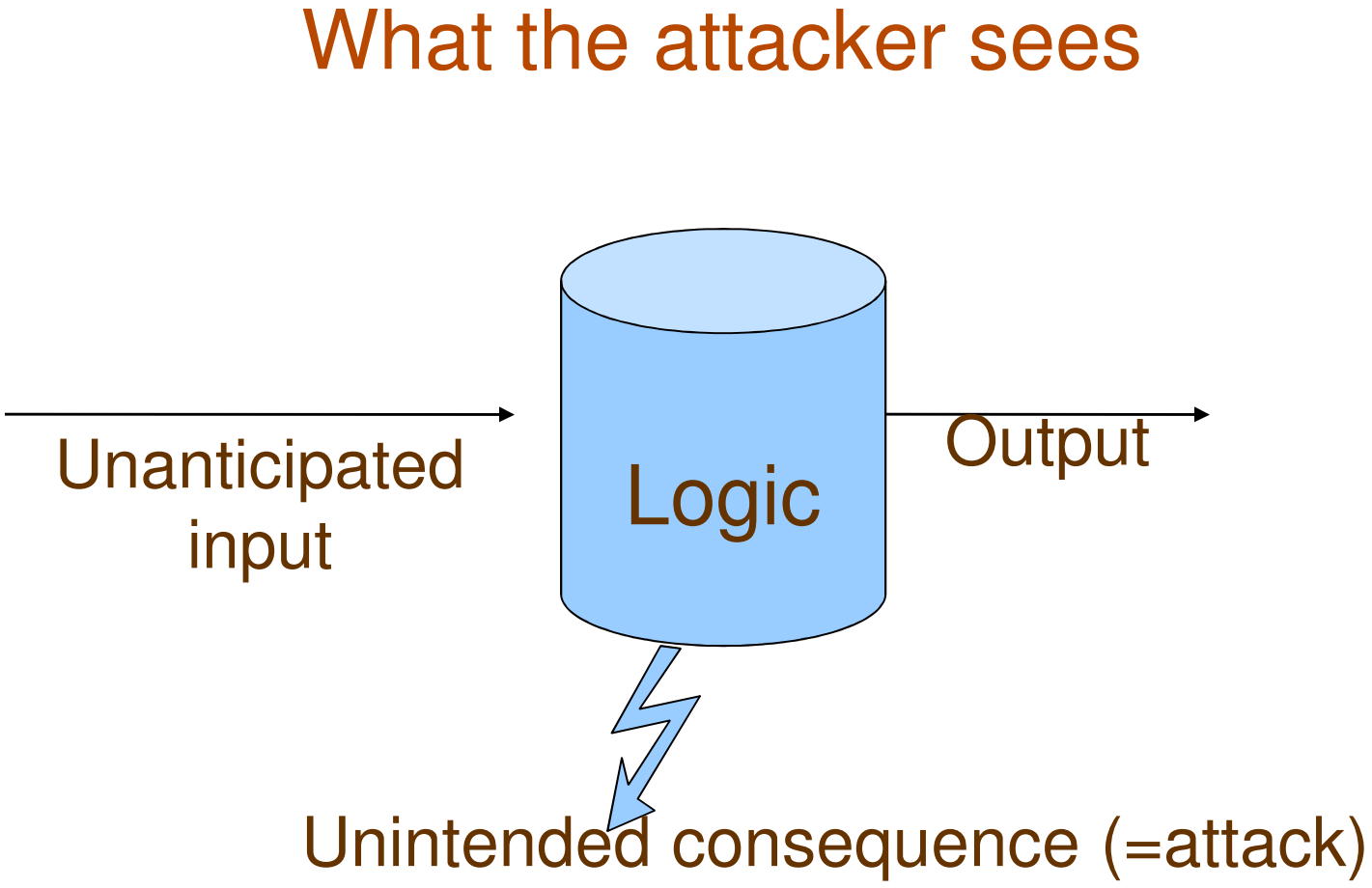
The most secure system

Inputs are the problem

What the programmer sees



Inputs are the problem



What is blackbox testing?

- Testing by attacking the inputs and observing output/behavior
- Does not use the source code
- Does not assume knowledge about the system

What is blackbox testing?

- Testing by attacking the inputs and observing output/behavior
- Does not use the source code
- Does not assume knowledge about the system

The system is a black box

What is blackbox testing?

- Testing by attacking the inputs and observing output/behavior
- Does not use the source code
- Does not assume knowledge about the system

The system is a black box

This is how almost all security holes are discovered today

How to blackbox almost everything

Step 1: map all your inputs

- File inputs
- Network
  - IP
  - Wireless?
  - RFID?
- Library calls
- Command line parameters

Crazy input moment #1

So you mapped all your inputs.

Crazy input moment #1

So you mapped all your inputs.

Are you sure?

Crazy input moment #1

So you mapped all your inputs.

Are you sure?

Is that a Windows program you're developing?

Crazy input moment #1

So you mapped all your inputs.

Are you sure?

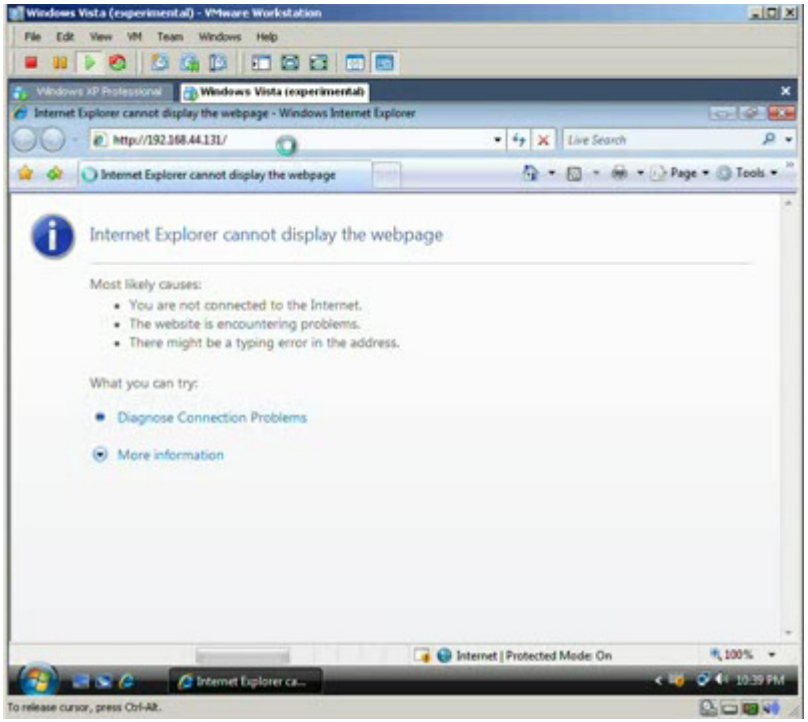
Is that a Windows program you're developing?

How are you handling WM_ messages?

Search SecuriTeam.com for **shatter**

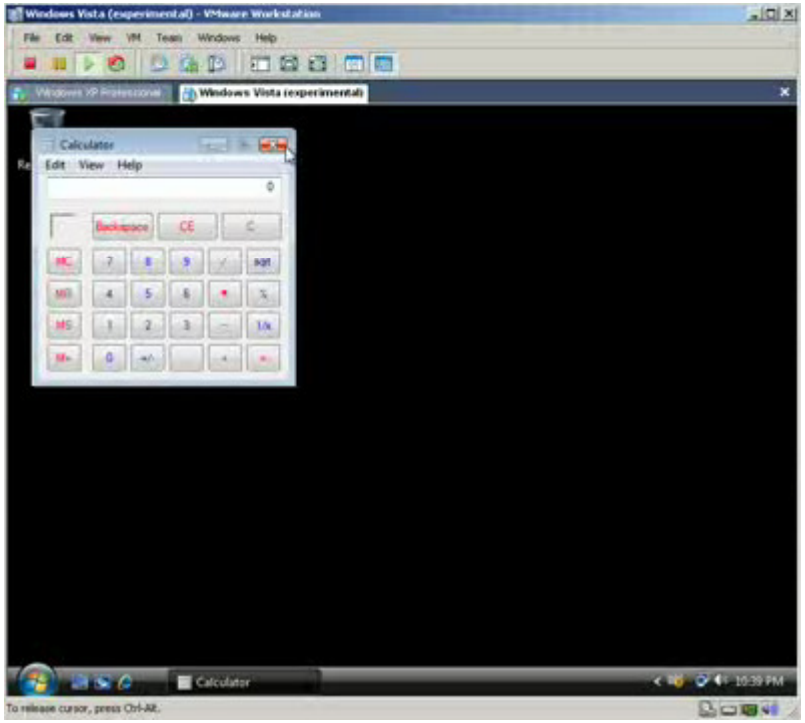
Why file input can be especially dangerous

“preview” - ANI attack



Why file input can be especially dangerous

“preview” - ANI attack



Who determines risk?

Not you!

Who determines risk?

Attackers attack what's easy and not where you ask them

How to blackbox almost everything

- Step 2: determine your “protocols”

Protocols

- Network: your RFC (or spec-based) protocol
- File: accepted file formats
- Library: DLL/ActiveX Interface
- Command line: variable definition

Command line arguments

NAME

cpio - copy files to and from archives

SYNOPSIS

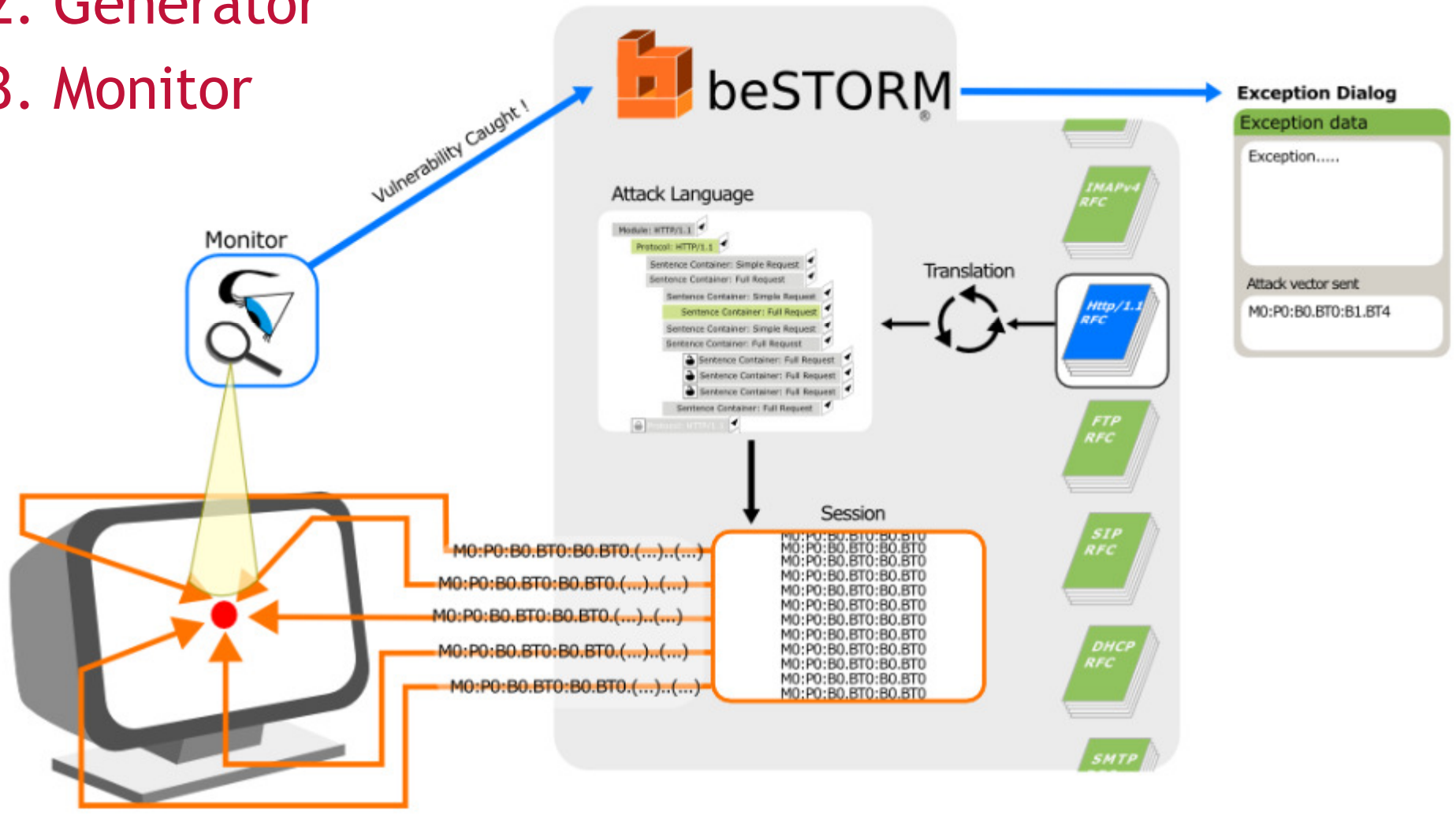
cpio {-o|--create} [-0acvABLV] [-C bytes] [-H format] [-M message] [-O [[user@]host:]archive] [-F [[user@]host:]archive] [--file=[[user@]host:]archive] [--format=format] [--message=message]

How to blackbox almost everything

- Step 3: Start testing

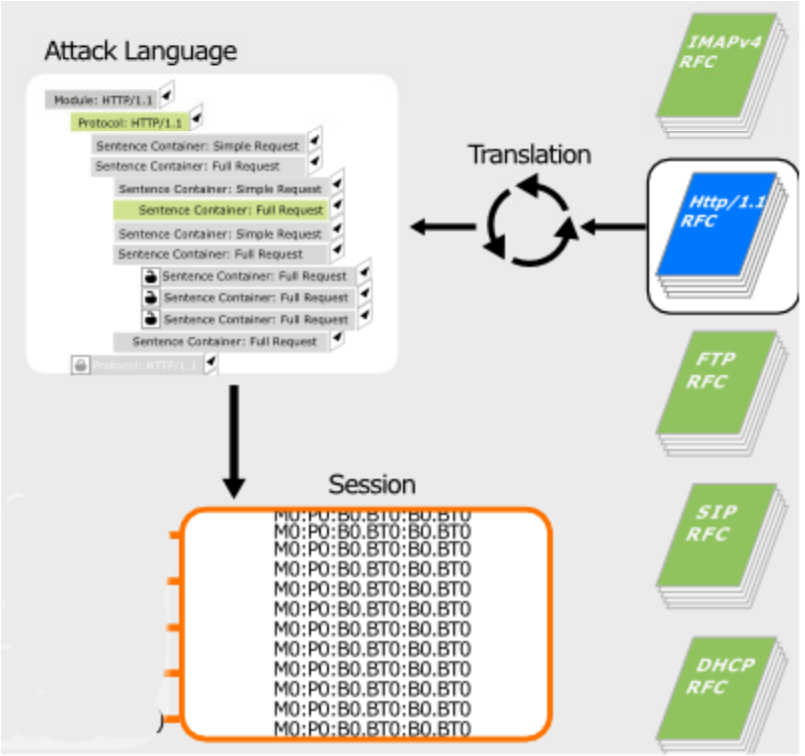
Ingredients

1. Test Module description
2. Generator
3. Monitor



Test Module

- Something that can describe “many” “different” sessions (=attacks)
- Protocol coverage is key



Example: beSTORM BSP file format

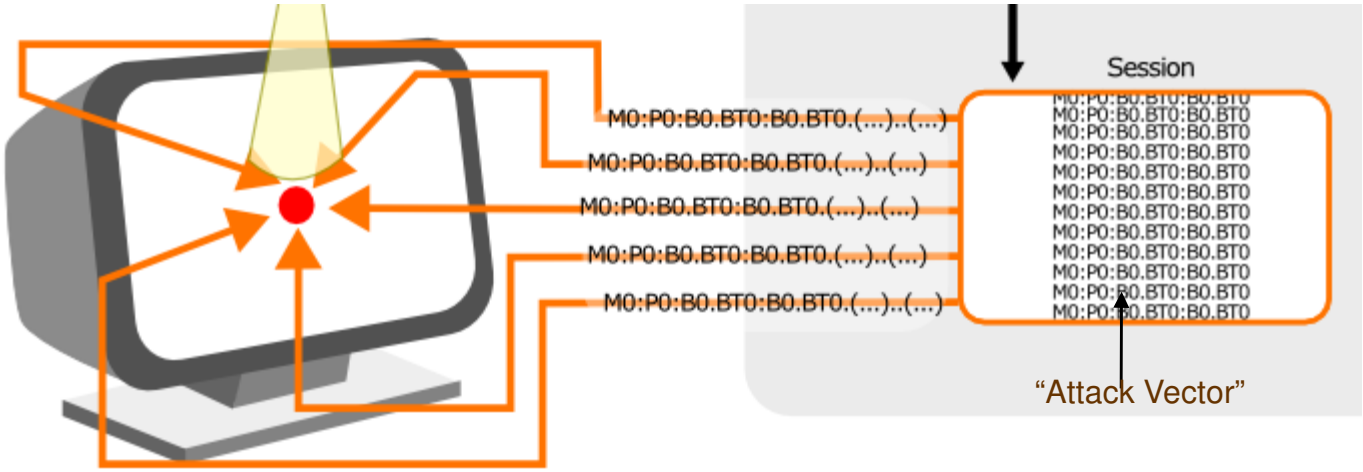
```
<SC Name=”ICAP Request”>
 <SE Name=”ICAP Method”>
  <S Name=”ICAP Method Enumerating”>
   <E Name=”ICAP Methods”>
    <C Name=”REQMOD Method” ASCIIValue=”REQMOD” />
    <C Name=”OPTIONS Method” ASCIIValue=”OPTIONS” />
   </E>
  </S>
  <S Name=”ICAP Method Overflow”>
   <B Name=”ICAP Method Overflowing” ASCIIValue=”RESPMOD” />
  </S>
 </SE>
 <S Name=”Request Line”>
  <C Name=”Space” ASCIIValue=” “ />
  <B Name=”icap Prefix” ASCIIValue=”icap” />
  <C Name=”ColonSlashes” ASCIIValue=”://“ />
  <B Name=”Address” ASCIIValue=”10.50.10.71“ />
 </S>
</SC>
```

Example: beSTORM BSP for file fuzzing

```
<M Name="TGA" >
        <P Name="TGA Protocol" >
           <SP Name="Writer"  Library="File Utils.dll" Procedure="Write">
              <S Name="Path" >  <VB Name="Whatever" Description="Path to store files"
    NoDefaultTypes="1" ASCIIValue="c:\\temp" />  </S>
              <S Name="Directory Splitter" >
                 <VB Name="Whatever" Description="Directory Splitter size"
    NoDefaultTypes="1" ASCIIValue="2" />
              </S>
              <S Name="Extension" >
                 <VB Name="Whatever" Description="Extension" NoDefaultTypes="1"
    ASCIIValue="tga" />
              </S>
              <SC Name="Data" >
                 <S Name="Color-mapped images" >
                    <L Name="Identsize"  ConditionedName="Image Identification Field"
    Size="1" />
                    <B Name="Colour Map Type"  Default="0x00" MaxBytes="1" />
                    <B Name="Image Type Code"  Default="0x02" MaxBytes="1" />
                    <B Name="Color Map Origin"  Default="0x00,0x00" MaxBytes="2" />
                    <B Name="Color Map Length"  Default="0x00,0x00" MaxBytes="2" />
                    <L Name="Color Map Entry Size"  ConditionedName="Color map data"
    Size="1" />
```

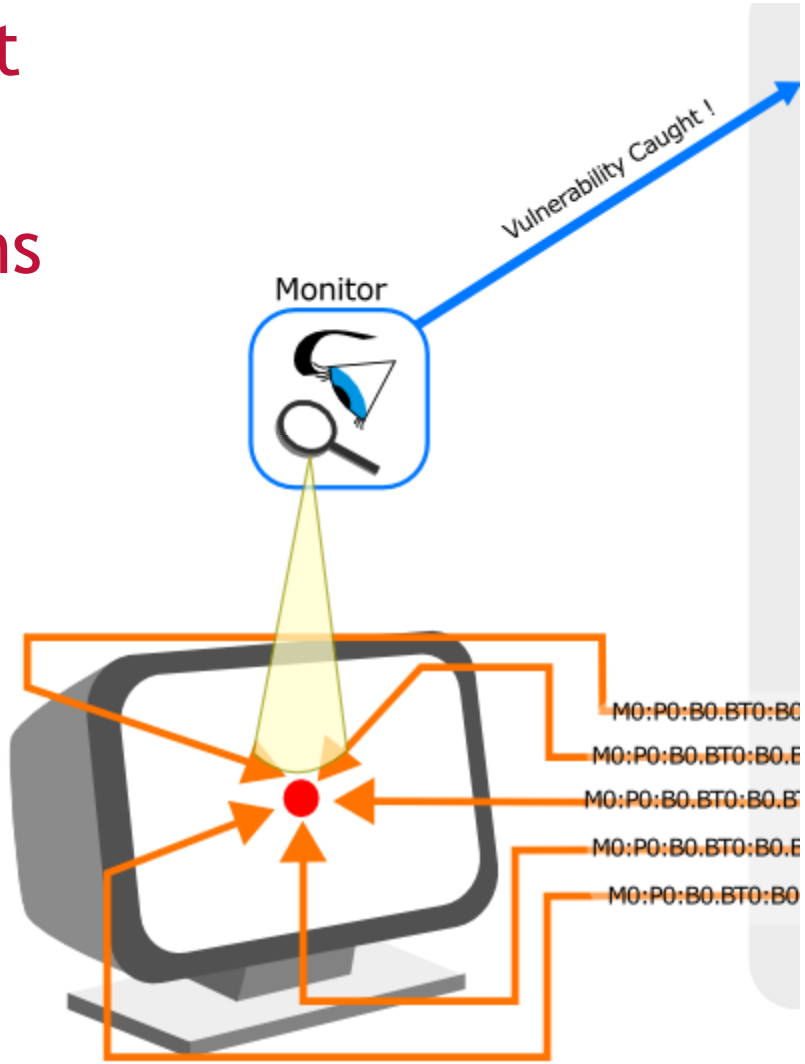
Generator

- Something that can take the module description and send it to the program:
  - Over the network
  - By creating a file
  - By invoking a DLL function



Monitor

- Possibly the most important component
- So you're generating millions of attacks: but how do you know you succeeded?



Monitoring

Monitor for:

- Memory exceptions (“first chance exceptions”)
- Program stops responding
- Errors in Logs (via regex)

- Recommendation: try to connect the monitor with the generator to correlate

Easy to use and extend

- Windbg
- gdb

Hard to do:

- Embedded

Key factors

- Automation
- Re-creating the attacks
- Ensuring protocol coverage (not code coverage!)

Cool and recent fuzzing attacks

MONDAY, SEPTEMBER 7, 2009

[Updated]Windows Vista/7 : SMB2.0 NEGOTIATE PROTOCOL REQUEST Remote B.S.O.D.

==========
- Release date: September 7th, 2009
- Discovered by: Laurent Gaffié
- Severity: High
==========

I. VULNERABILITY

Windows Vista, Server 2008 < R2, 7 RC :
SMB2.0 NEGOTIATE PROTOCOL REQUEST Remote B.S.O.D.

SRV2.SYS fails to handle malformed SMB headers for the NEGOTIATE PROTOCOL REQUEST functionnality.
The NEGOTIATE PROTOCOL REQUEST is the first SMB query a client send to a SMB server, and it's used to identify the SMB dialect that will be used for futher communication.

WEDNESDAY, NOVEMBER 11, 2009

Windows 7 / Server 2008R2 Remote Kernel Crash

netbios_header = struct.pack(">i", len(''.join(SMB_packet))+SMB_packet
(The netbios header provide the length of the incoming smb{1,2} packet)

If netbios_header is 4 bytes smaller or more than SMB_packet, it just blow !

Find both at: http://g-laurent.blogspot.com

My all time favorites

- ANI file attack
- RFID (www.RFIDguardian.org)
- PDF, XLS, JPG, BMP
- All file-based attacks
- Network printer attacks
- Rain Forest Puppy MSADC attack

Thank you!

aviram@beyondsecurity.com

www.beyondsecurity.com