



# Fuzzing

Brute Force Vulnerability Discovery



Michael Sutton  
Director, iDefense Labs

[msutton@idefense.com](mailto:msutton@idefense.com)

Where it all comes together:

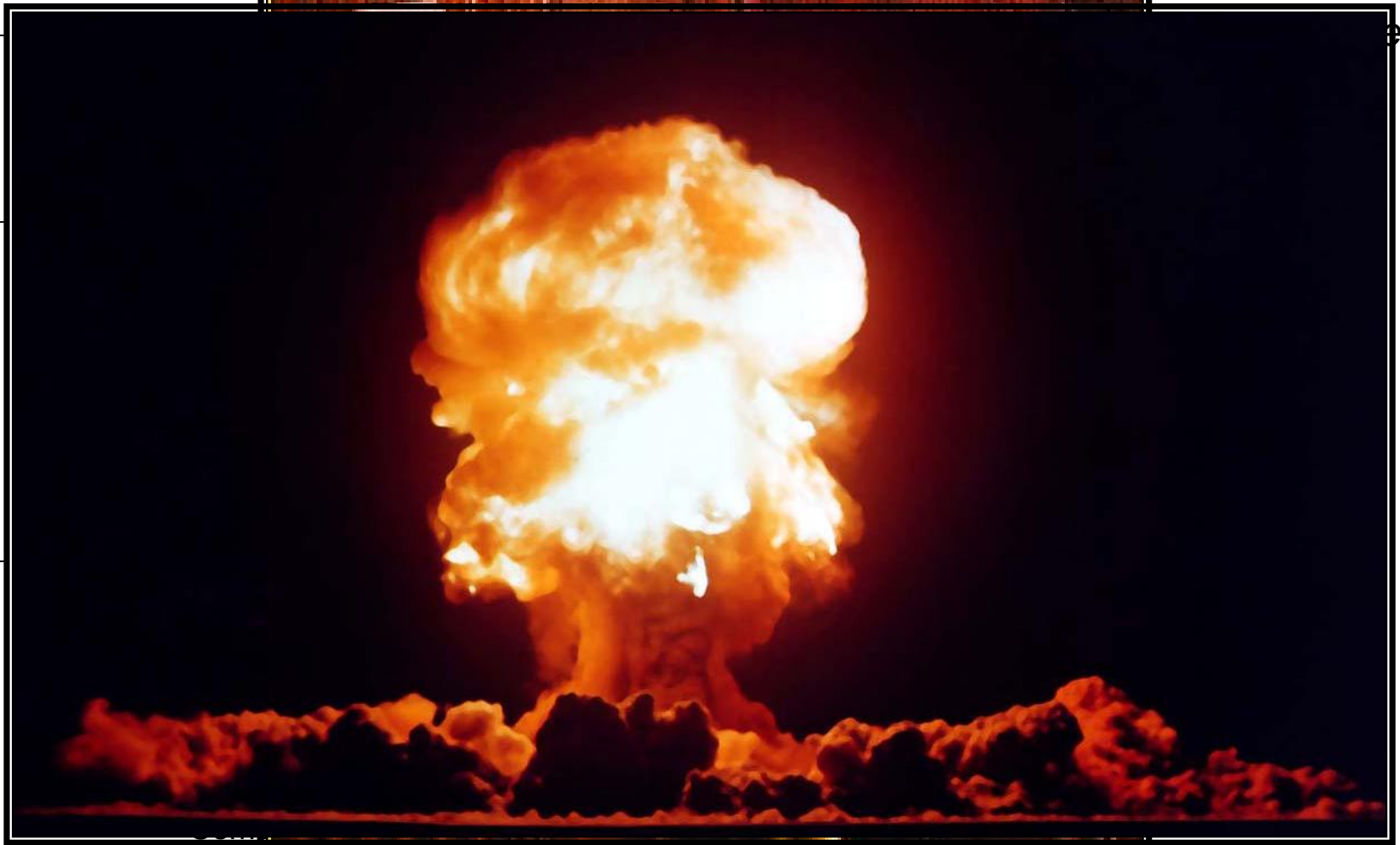
# Agenda

---

- + Background
  - What is fuzzing and who should do it?
- + Phases
  - What are the various stages when fuzzing a target?
- + Fuzzer classes
  - What can be fuzzed?
- + Automation
  - Making the theoretical practical
- + Tools/Demos
  - FileFuzz
  - WebFuzz
  - COMRaider
- + Advanced topics
- + The future or fuzzing



# Vulnerability Discovery Methodologies – Black Box





# What is Fuzzing?

+ “Fuzz testing is a technique for attaching a program to a program and asserting that it will not crash.”

The great thing about fuzzing is that it is simple, and it can be automated.

- [Wiki](#)

+ “Unexpected behavior is a common result of fuzzing.”

- [Microsoft](#)

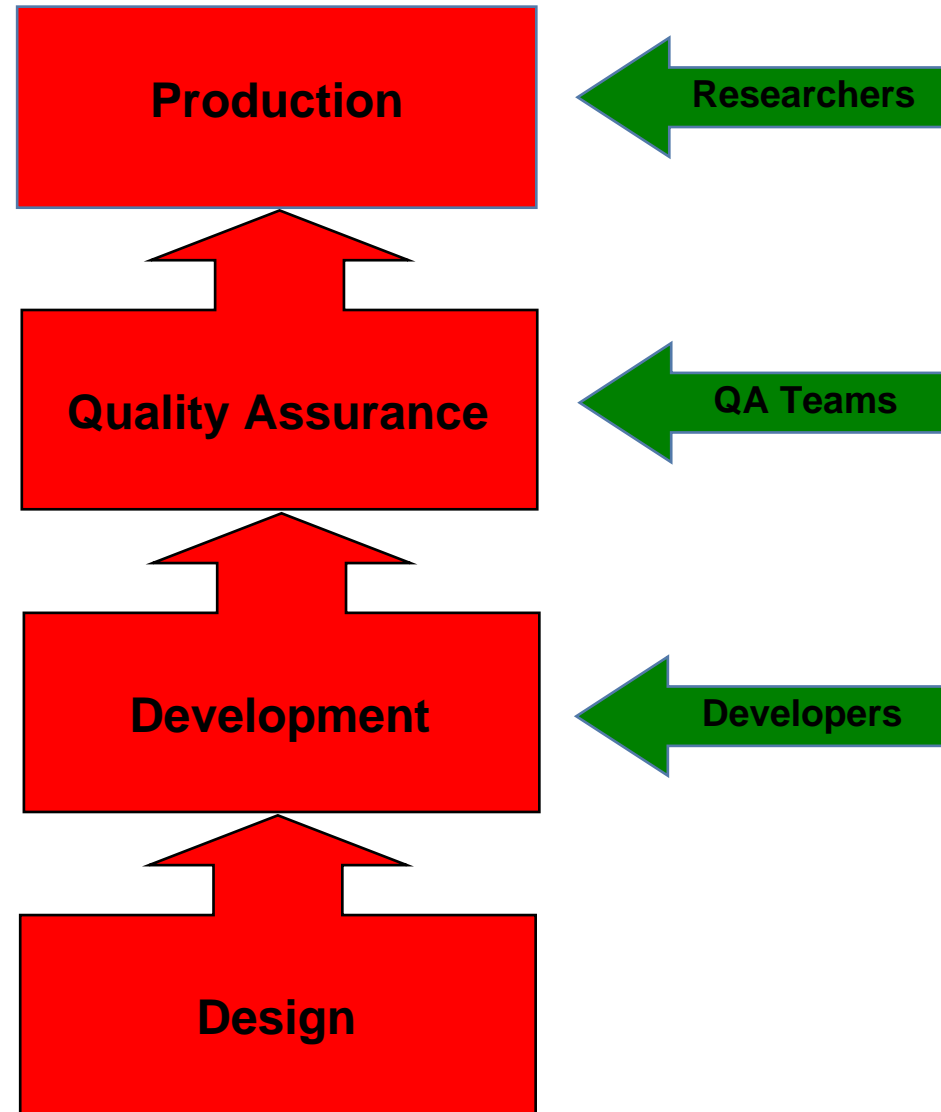


basic idea is to "fuzz" (i.e., "fuzz" the program). If the program crashes, the developer can then investigate the cause.

extremely

# Who should fuzz?

- + Security researchers
  - Reactive
- + QA Teams
  - Proactive
- + Developers
  - Proactive



# What can fuzzing do for you?

+ MS06-

- ak
- Ap
- Ev

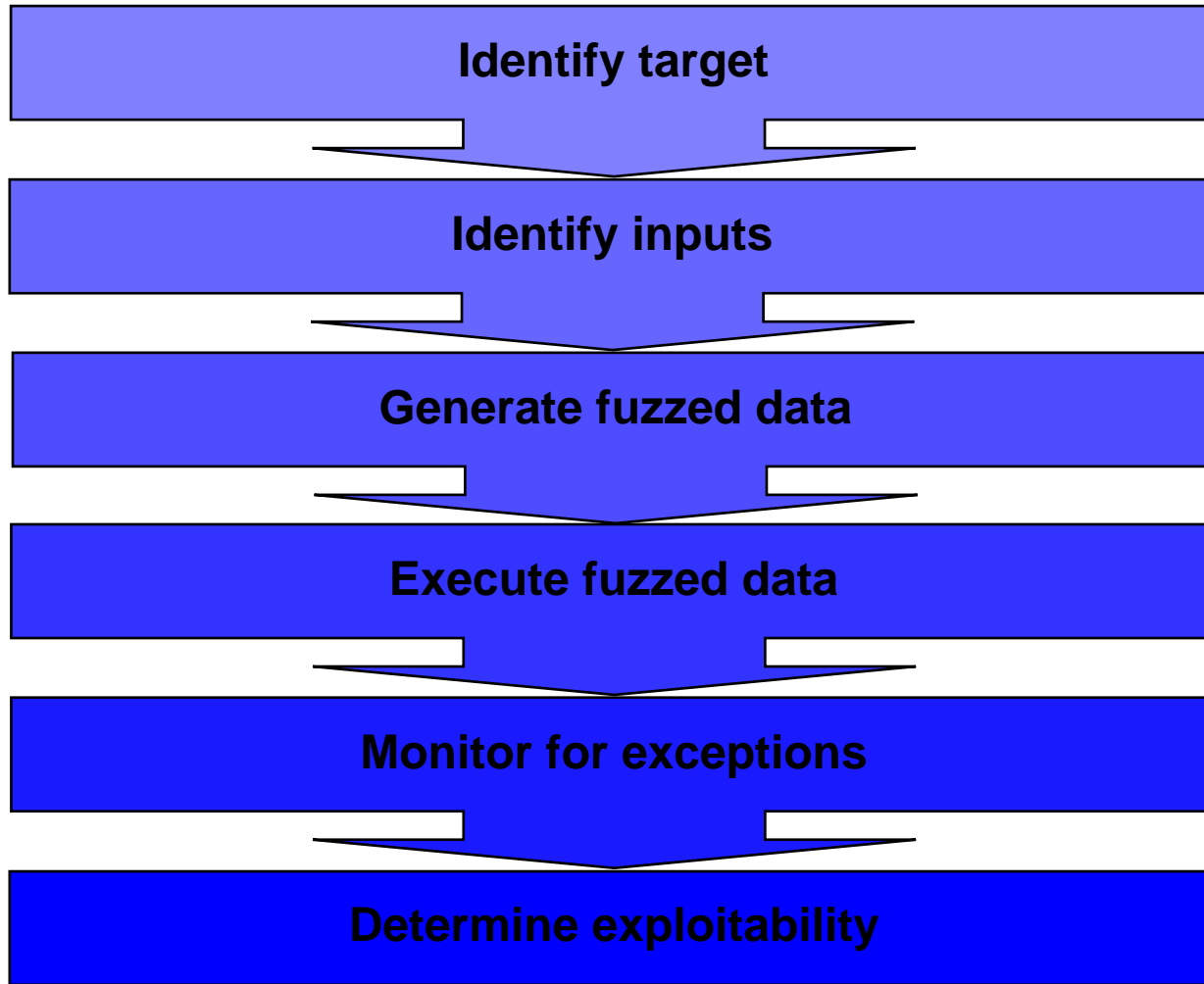
AIF = ap  
 ANF = at  
 apaf = ap  
 ARC = aj  
 recruitme  
 BH = bcl  
 CASH = c  
 CD = clus  
 DED = de  
 DR = dea  
 ERK = ex  
 FADD = ]  
 FasL = Fa  
 FLAME-  
 molecule  
 FLICE = l  
 protease  
 FLIP = F  
 I kappa F  
 I-FLICE  
 IAP = inh  
 ICE = int  
 IGF = ins  
 JNK = c-  
 MAPK =

```
(o)0001760: 04be 0000 2e01 1800 0400 0000 0201 0100 .....
(n)0001760: 0400 0000 2e01 1800 0400 0000 0201 0100 .....
---
(o)0001780: 7777 7777 7720 4220 3d20 7777 7777 7777 www B = www
(n)0001780: 6b61 7070 6120 4220 3d20 696e 6869 6269 kappa B = inhibi
---
(o)0001790: 7777 7777 7777 7777 7777 7777 7777 7720 www
(n)0001790: 746f 7220 6f66 204e 4620 6b61 7070 6120 tor of NF kappa
---
(o)0001850: 0000 0000 0400 0000 2e01 1800 0400 009f .....
(n)0001850: 0000 0000 0400 0000 2e01 1800 0400 0000 .....
---
(o)0001860: 0a01 0100 0900 0000 320a e801 3100 0100 .....2...1...
(n)0001860: 0201 0100 0900 0000 320a e801 3100 0100 .....2...1...
---
(o)0001880: 0000 02b0 0200 0500 0000 0902 ffff ff02 .....
(n)0001880: 0000 0201 0200 0500 0000 0902 ffff ff02 .....
---
(o)00018b0: e801 3700 1b00 0000 7777 7777 7777 7777 ..7....www
(n)00018b0: e801 3700 1b00 0000 464c 4943 4520 3d20 ..7....FLICE =
---
(o)00018c0: 7777 7777 7777 7777 7777 7777 7777 7777 www
(n)00018c0: 696e 6869 6269 746f 7220 6f66 2046 4c49 inhibitor of FLI
---
(o)00018d0: 7777 2000 0c00 0c00 0700 0e00 0d00 0500 ww .....
(n)00018d0: 4345 2000 0c00 0c00 0700 0e00 0d00 0500 CE .....
---
(o)0001920: ffff ff02 0500 0032 1402 0000 0000 0400 .....2.....
(n)0001920: ffff ff02 0500 0000 1402 0000 0000 0400 .....
---
(o)0001940: 0000 320a 0602 2a00 1300 0000 7777 7777 ..2...*....www
(n)0001940: 0000 320a 0602 2a00 1300 0000 4941 5020 ..2...*....IAP
---
(o)0001950: 7777 7777 7777 7777 7777 7777 7777 2000 www
(n)0001950: 3d20 696e 6869 6269 746f 7220 6f66 2000 = inhibitor of
---
(o)0001960: 0722 0d00 0c00 0500 0b00 0500 0500 0a00 .".....
(n)0001960: 0700 0d00 0c00 0500 0b00 0500 0500 0a00 .....
```

em as text  
 ase  
 discover the

anine-

# Phases





# Fuzzer Classes

---

- + Command line arguments
- + Environment variables
  - [Sharefuzz \(www.immunitysec.com\)](http://www.immunitysec.com)
- + Web applications
  - [WebFuzz \(Demo\)](#)
- + File formats
  - [FileFuzz \(Demo – labs.idefense.com\)](#)
- + Network protocols
  - [SPIKE \(www.immunitysec.com\)](http://www.immunitysec.com)
- + Memory
- + COM Objects
  - [COMRaider \(Demo – labs.idefense.com\)](#)
- + Inter-Process Communication (IPC)

# Automation

---

## + Test cases

- **Approach**
  - Pre-generated test cases
- **Tools**
  - PROTOS Test Suites
- **Pro**
  - Consistency
- **Con**
  - Static
  - Time consuming

# Automation

---

## + Brute force fuzzing

- **Approach**
  - Raw byte manipulation
- **Tool(s)**
  - FileFuzz
- **Pro**
  - Simple
- **Con**
  - Inefficient
  - Fails to account for dependent values (e.g. checksums)

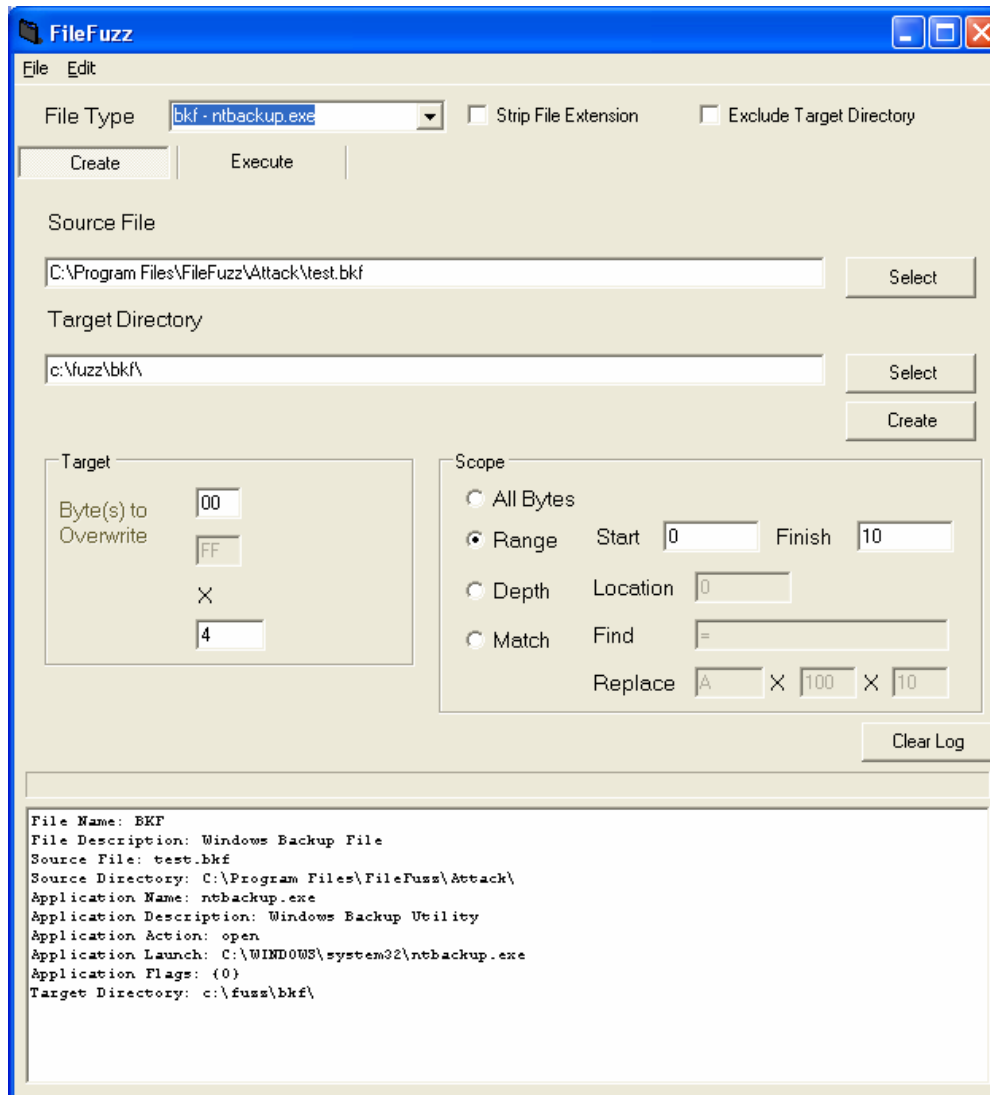
# Automation

---

## + 'Intelligent' fuzzing

- **Approach**
  - Templates developed based on protocol definitions
- **Tools**
  - SPIKE
  - SPIKEfile
- **Pro**
  - Efficient
- **Con**
  - Time consuming

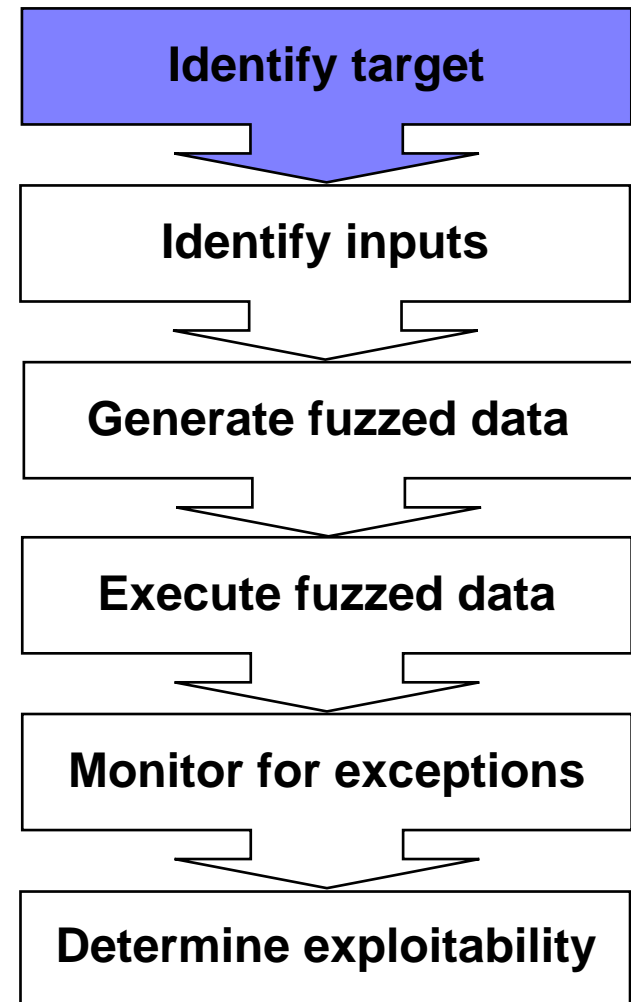
# FileFuzz





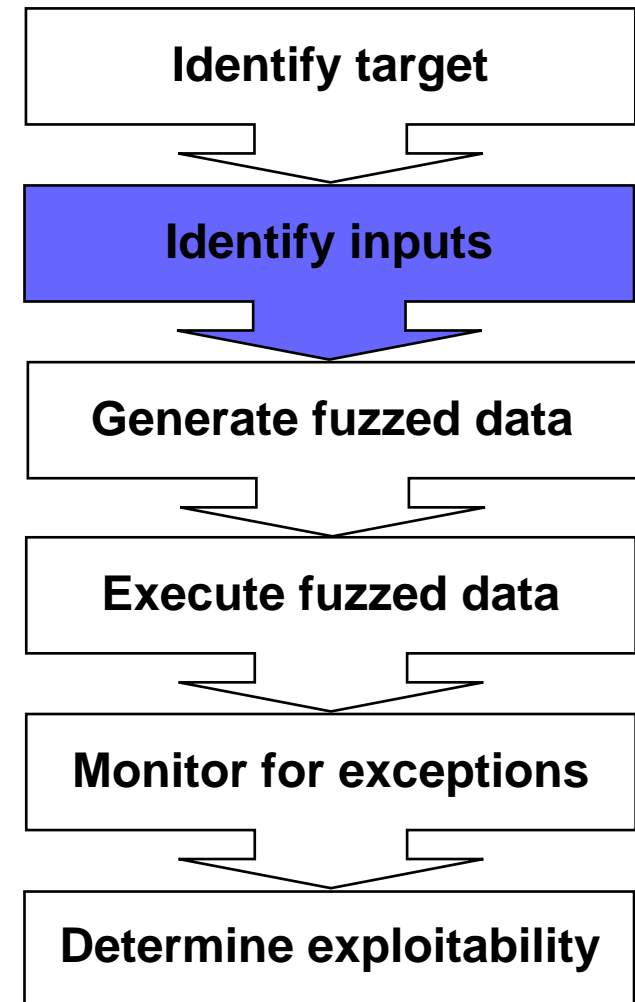
# FileFuzz – Identify Target

- + Application vs. file type
  - One file type → multiple targets
- + Vendor history
  - Past vulnerabilities
- + High risk targets
  - Default file handlers
    - Windows Explorer
    - Windows Registry
  - Commonly traded file types
    - Media files
    - Office documents
    - Configuration files



# FileFuzz – Identify Inputs

- + Proprietary vs. open formats
  - Vendor documents
  - Wotsit.org
  - Google
- + Binary files
  - e.g. images, video, audio, office documents, etc.
  - Headers vs. data
- + Text files
  - e.g. \*.ini, \*.inf, \*.xml
  - Name/value pairs



# FileFuzz – Generate Fuzzed Data

## + Binary files

- **Breadth (All or Range)**

- Identify potential weaknesses

```
FF FF FF FF 00 00 DB FE 0B 00 C5 00 00 01 E8 03 ; ýÿÿÿ..Ûp..Å...è.  
D7 FF FF FF FF 00 DB FE 0B 00 C5 00 00 01 E8 03 ; xÿÿÿÿ..Ûp..Å...è.  
D7 CD FF FF FF FF DB FE 0B 00 C5 00 00 01 E8 03 ; xÍÿÿÿÿÛp..Å...è.
```

- **Depth**

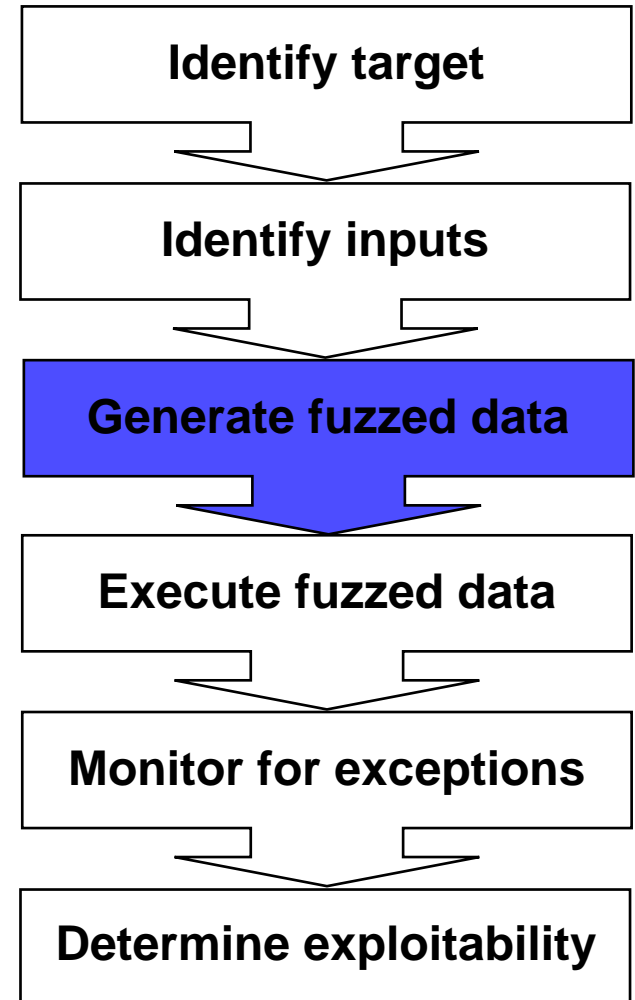
- Determine level of control/influence

```
D7 CD FD 9A 00 00 DB FE 0B 00 C5 00 00 01 E8 03 ; xÍÿš..Ûp..Å...è.  
D7 CD FE 9A 00 00 DB FE 0B 00 C5 00 00 01 E8 03 ; xÍpš..Ûp..Å...è.  
D7 CD FF 9A 00 00 DB FE 0B 00 C5 00 00 01 E8 03 ; xÍÿš..Ûp..Å...è.
```

## + Text Files

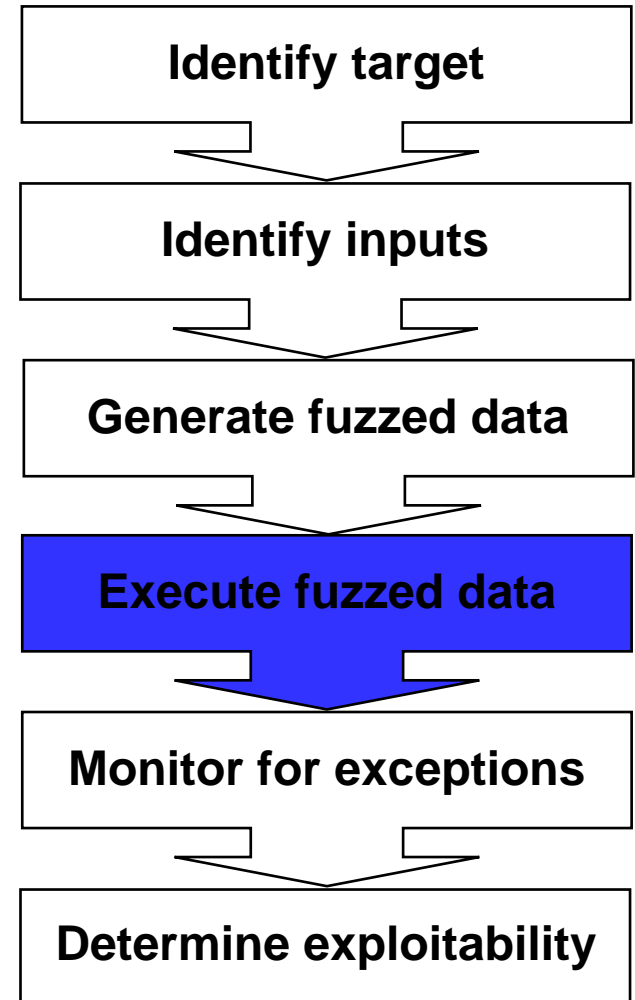
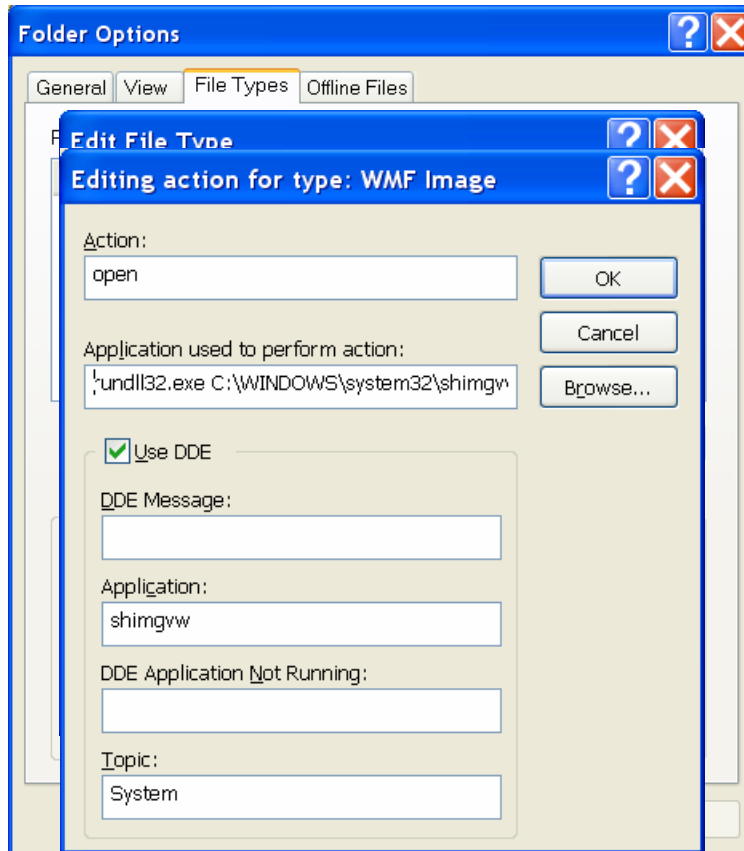
- **name = value**

```
file_size = 10  
file_size = AAAAAA  
file_size = AAAAAAAAAA
```



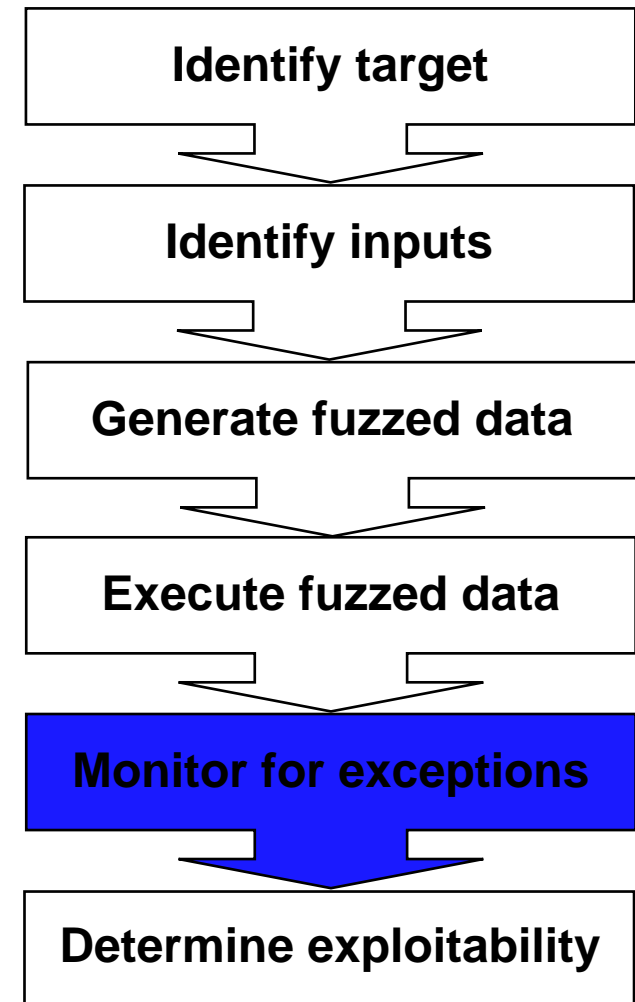
# FileFuzz – Execute Fuzzed Data

- + Command line arguments
  - Windows explorer
    - Tools...Folder Options...File Types



# FileFuzz – Monitor for Exceptions

- + Visual
  - Error messages
  - Blue screen
- + Event logs
  - System logs
  - Application logs
- + Debuggers
- + Return codes
- + Debugging API



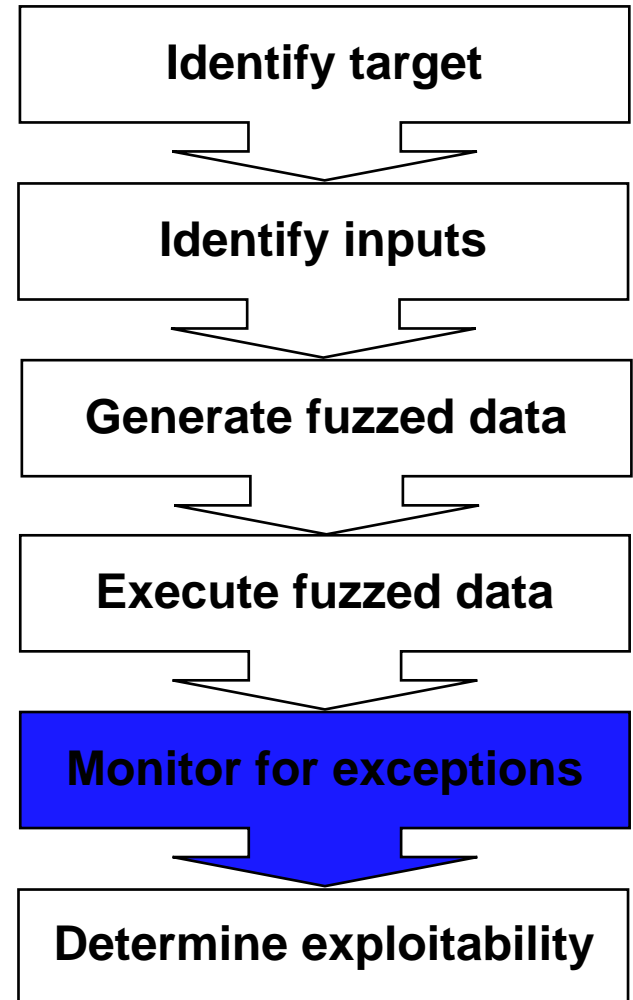


# FileFuzz – Monitor for Exceptions

- + Execute
  - Automated and repeated
- + Monitor
  - Library - libdasm
  - Capture
    - Memory location
    - Registry values
    - Exception type

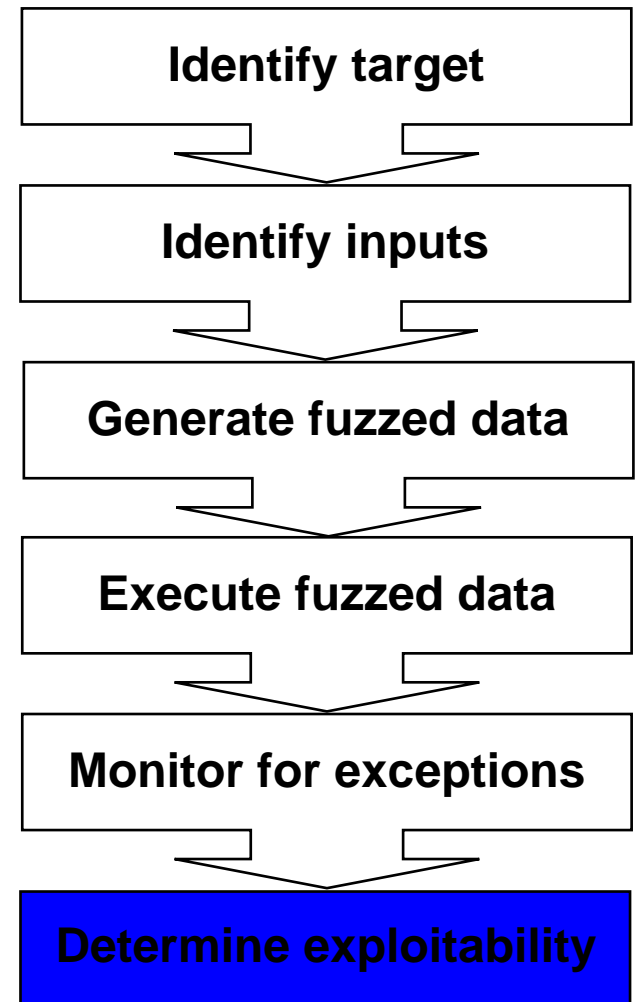
```
[*] "crash.exe" "C:\Program Files\WordPerfect Office
12\Programs\UA120.exe" 2000 /qt c:\fuzz\ast\8.ast
[*] Access Violation
[*] Exception caught at 00403f06 mov eax,[eax+edi*4]
[*] EAX:0014b1b8 EBX:00000005 ECX:00435c00 EDX:0012fbac
[*] ESI:00435c00 EDI:cccccccc ESP:0012fab8 EBP:0012fae8
```

- + Kill
  - Set timeout

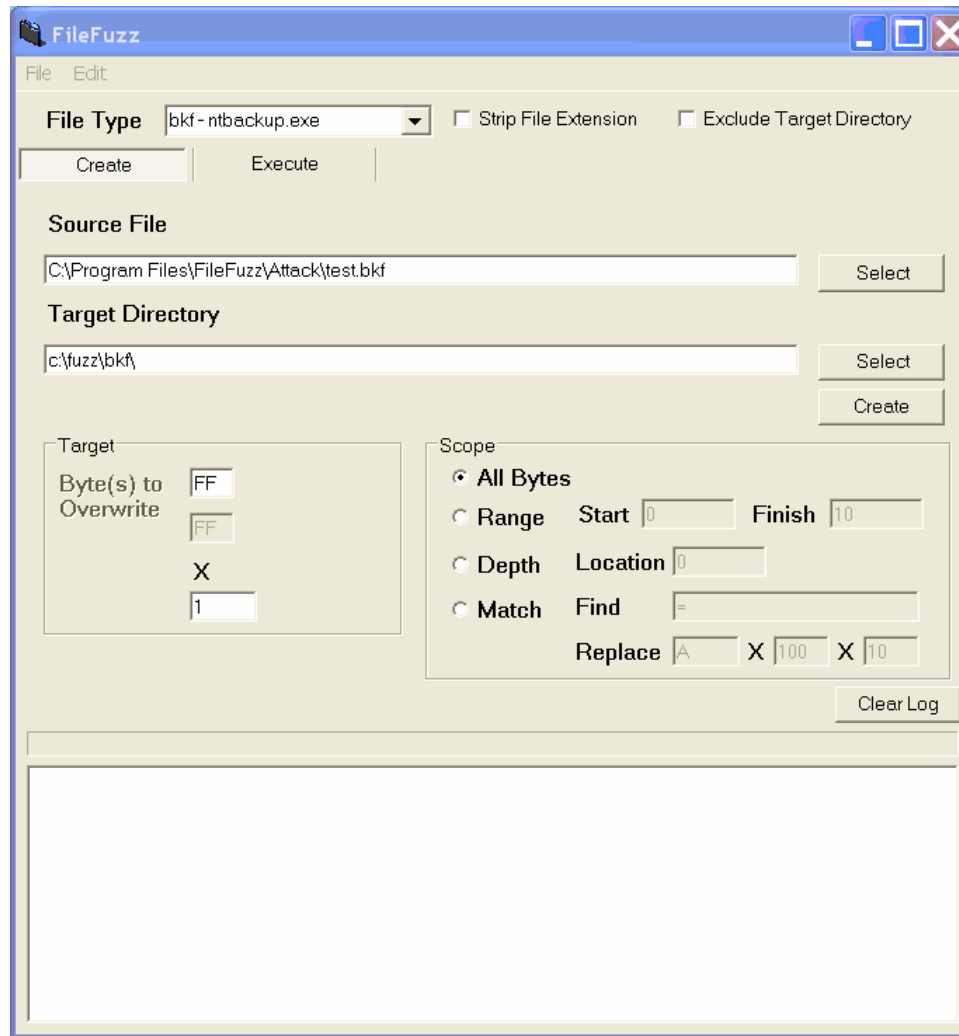


# FileFuzz – Determine Exploitability

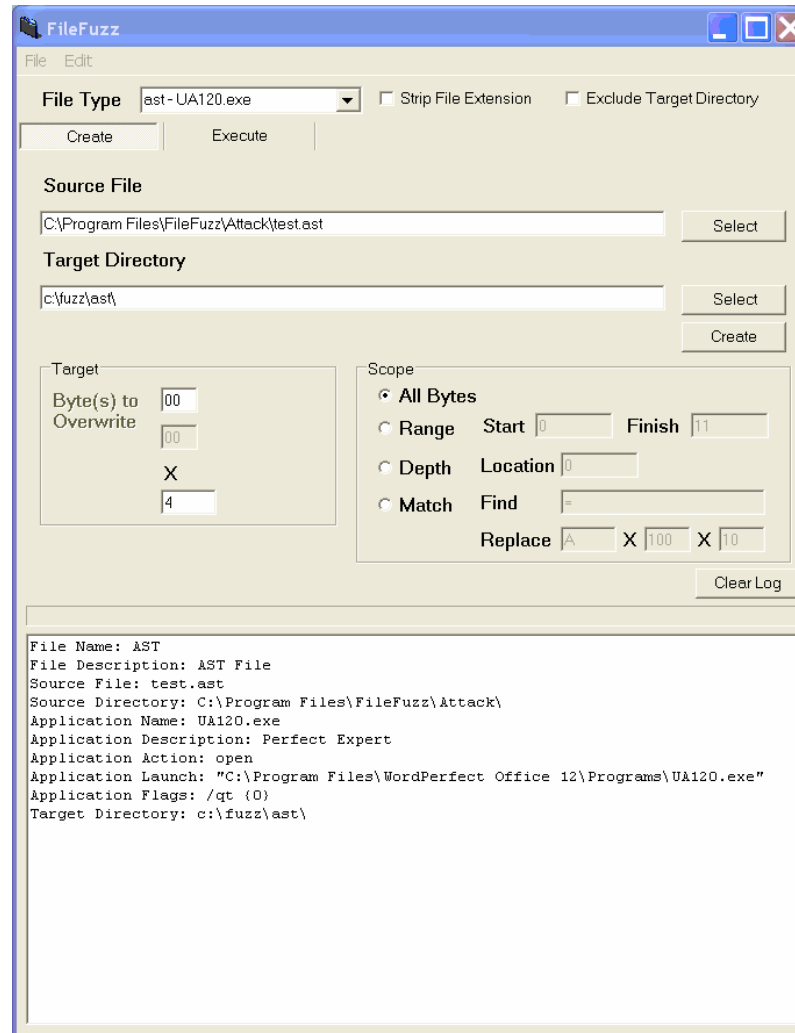
- + Skills
  - Disassembly
  - Debugging
- + Vulnerability types
  - Stack overflows
  - Heap overflows
  - Integer handling
    - Overflows
    - Signedness
  - DoS
    - Out of bounds reads
    - Infinite loops
    - NULL pointer dereferences
  - Logic errors
    - Windows WMF vulnerability (MS06-001)
  - Format strings
  - Race conditions



# FileFuzz – Demo (Breadth)



# FileFuzz – Demo (Depth)



# WebFuzz

The screenshot shows the WebFuzz application window. At the top, the Host is set to `www.iddefense.com`, Port is `80`, and Timeout (Milliseconds) is `5000`. The Request Headers section displays the following information:

```
GET / HTTP/1.1
Accept: */*
Accept-Language: en-us
Pragma: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; InfoPath.1)
Host: localhost
Proxy-Connection: Keep-Alive
```

The Responses section contains a table with the following data:

No.	Status	Host	Request
0	200	www.iddefense.com	GET / HTTP/1.1

The Raw Response section is selected, showing the following content:

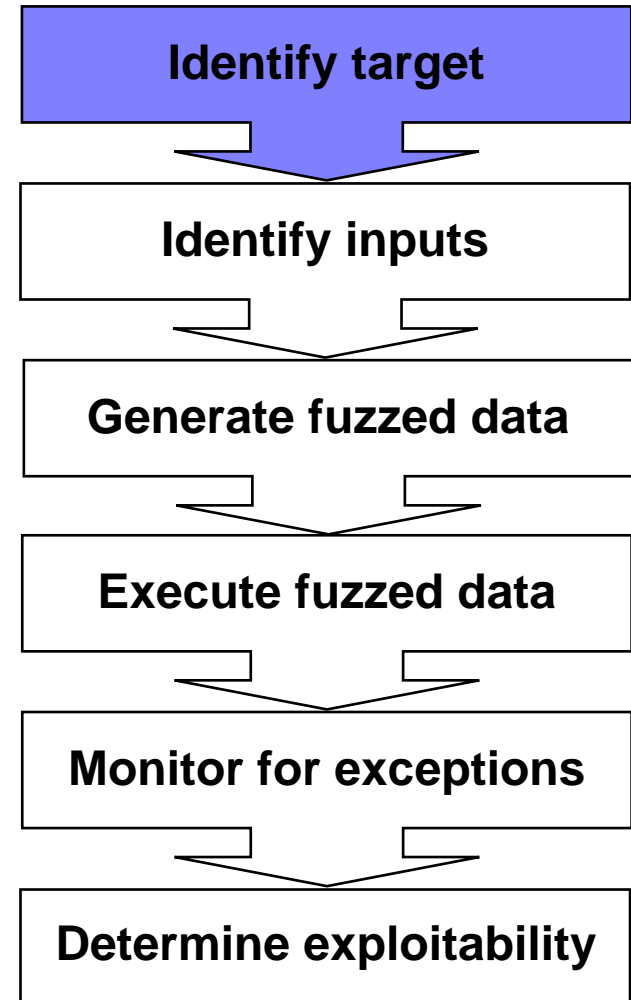
```
HTTP/1.1 200 OK
Date: Fri, 21 Apr 2006 22:24:00 GMT
Server: Apache/2.0.52 (Red Hat)
X-Powered-By: PHP/5.0.4
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=ISO-8859-1

1fb3
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-1">
  <meta HTTP-EQUIV="expires" CONTENT="0">
  <meta HTTP-EQUIV="pragma" CONTENT="no-cache">
  <meta name="keywords" content=" managed security services, network-based security threats, security vulnerability, protect critical data, infrastructure, cyber attacks, security experts, information security, managed firewall, managed security, managed vpn, penetration testing, network security, potential cyber threats, new malicious code, zero-day exploits, hacker groups, cyber crime, cyber terror, advanced warning, threat protection, critical infrastructure, verisign, iddefense, worm, virus, phishing, mss" />
  <meta name="description" content="VeriSign iDefense services deliver comprehensive, actionable intelligence regarding network-based security threats and vulnerabilities which can help organizations proactively protect critical data and infrastructure from attacks. VeriSign iDefense Security Intelligence Services are an important component of VeriSign's Managed Security Services [MSS]." />
  <title>VeriSign iDefense Security Intelligence Services -- Managed Security Services and Information Security for Government and Fortune 500 Organizations // VeriSign iDefense</title>
  <link rel="shortcut icon" href="/favicon.ico" />
  <link rel="icon" href="/favicon.ico" />
```



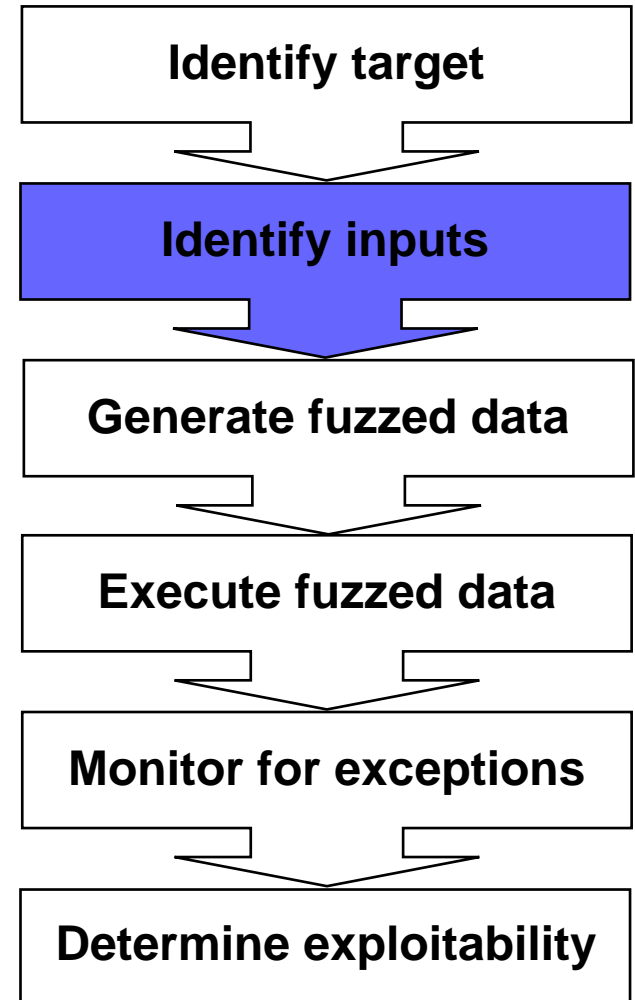
# WebFuzz – Identify Target

- + Server vs. Application
  - Targeting applications can uncover server vulnerabilities
- + Vendor history
  - Past vulnerabilities
- + High risk targets
  - Popular applications
    - Download site counters
    - Google queries (johnny.ihackstuff.com)
  - External applications
    - Wikis
    - Web mail
    - Discussion boards
    - Blogs



# WebFuzz – Identify Inputs

- + Potential input vectors
  - Method
  - Request-URI
  - Protocol
  - Headers
  - Cookies
  - Post data
- + Reconnaissance
  - Web forms
  - Authentication
  - Hidden fields
  - Client side scripting
- + Manual Tools
  - Proxies
  - LiveHTTPHeaders
- + Automated Tools
  - Spiders



# WebFuzz – Generate Fuzzed Data

## + Intelligent fuzzing

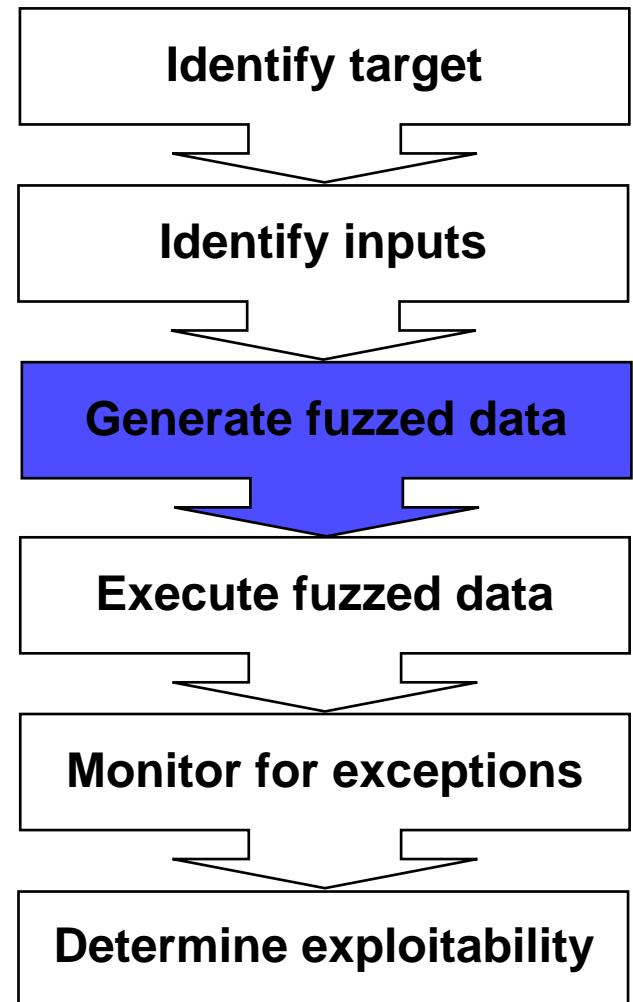
- Start with legitimate web request
- Build template to mutate requests

## + Request format

```
[Method] [Request-URI] HTTP/[Major Version].[Minor Version]
[HTTP Headers]
[Post Data]
```

## + Fuzz Template

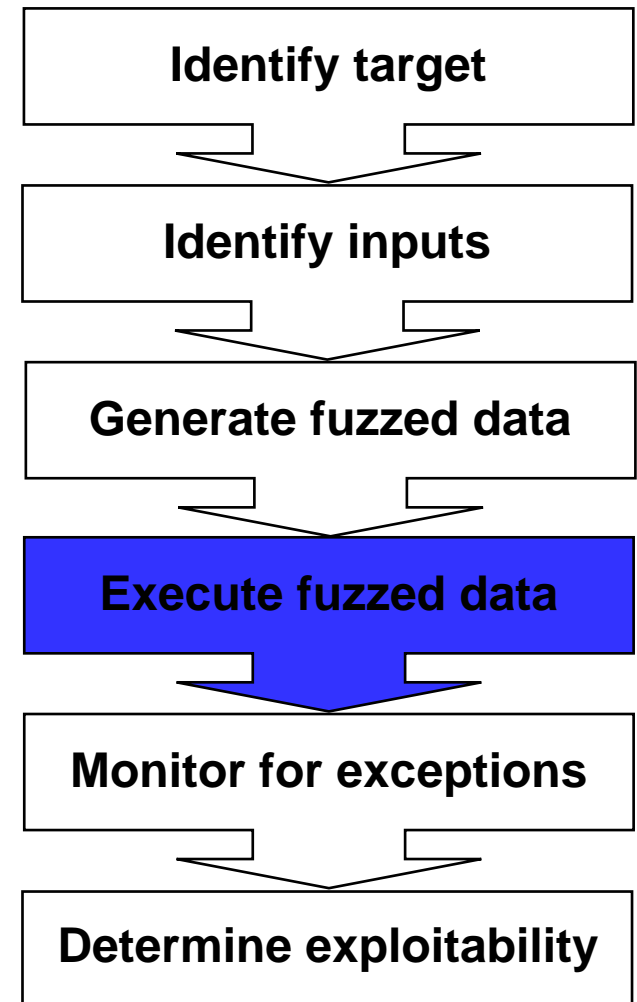
```
[Methods] / [Traversal] / page.html?x=[SQL]&y=[XSS] HTTP/1.1
Accept: */*
Accept-Language: en-us
Pragma: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;
SV1; InfoPath.1)
Host: [Overflow]
Proxy-Connection: Keep-Alive
```



# WebFuzz – Execute Fuzzed Data

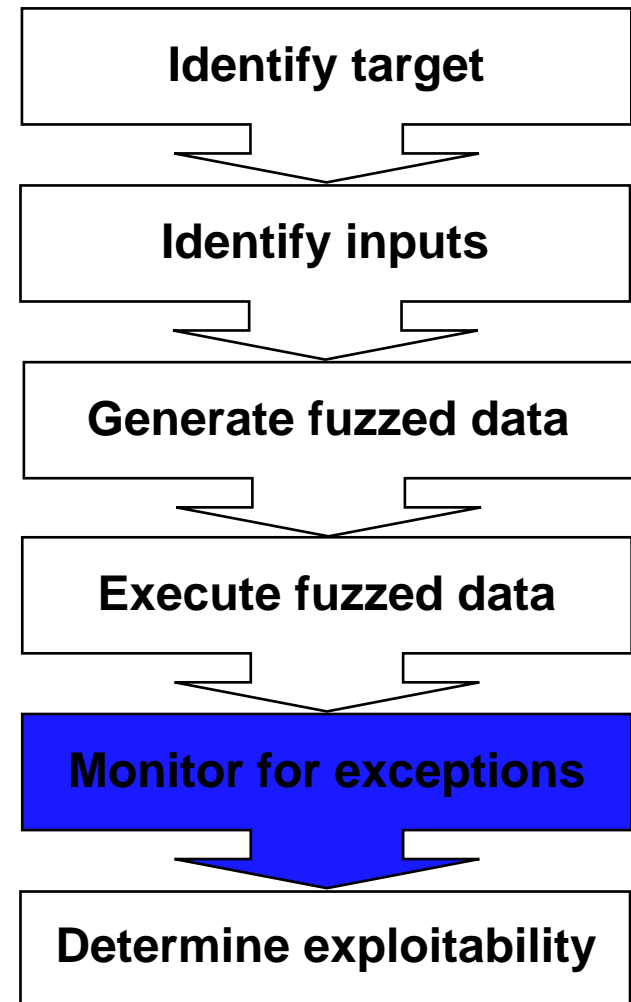
## + Fuzz classes

- Directory traversal
- Format strings
- Overflow
- SQL Injection
- XSS Injection



# WebFuzz – Monitor for Exceptions

- + Execute
  - Automated and repeated
- + Monitor
  - HTML response
    - Error messages
  - Raw response
    - User input
  - Status codes
- + Kill
  - Set timeout





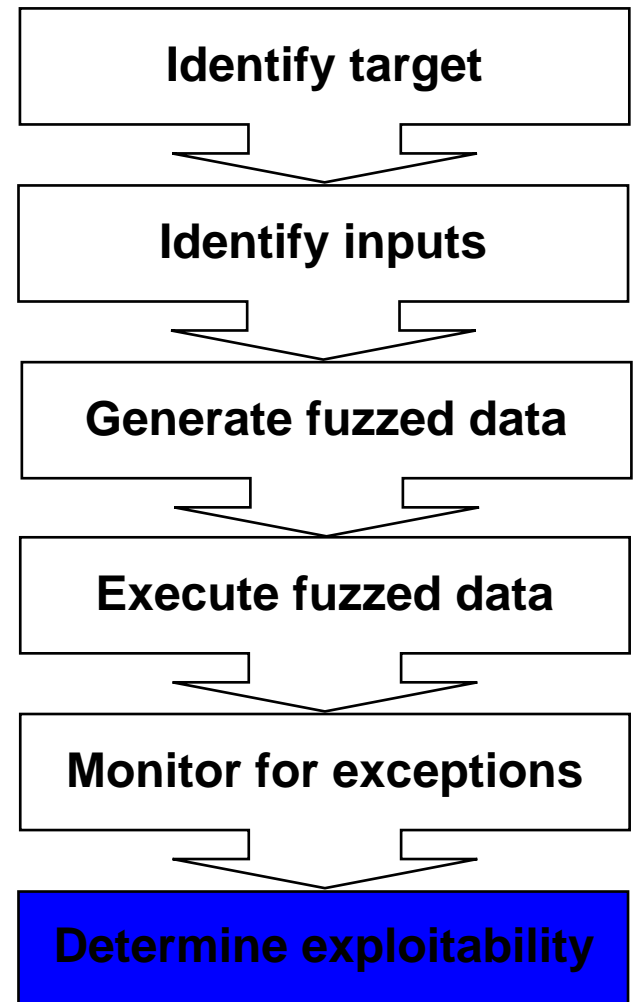
# WebFuzz – Determine Exploitability

## + Skills

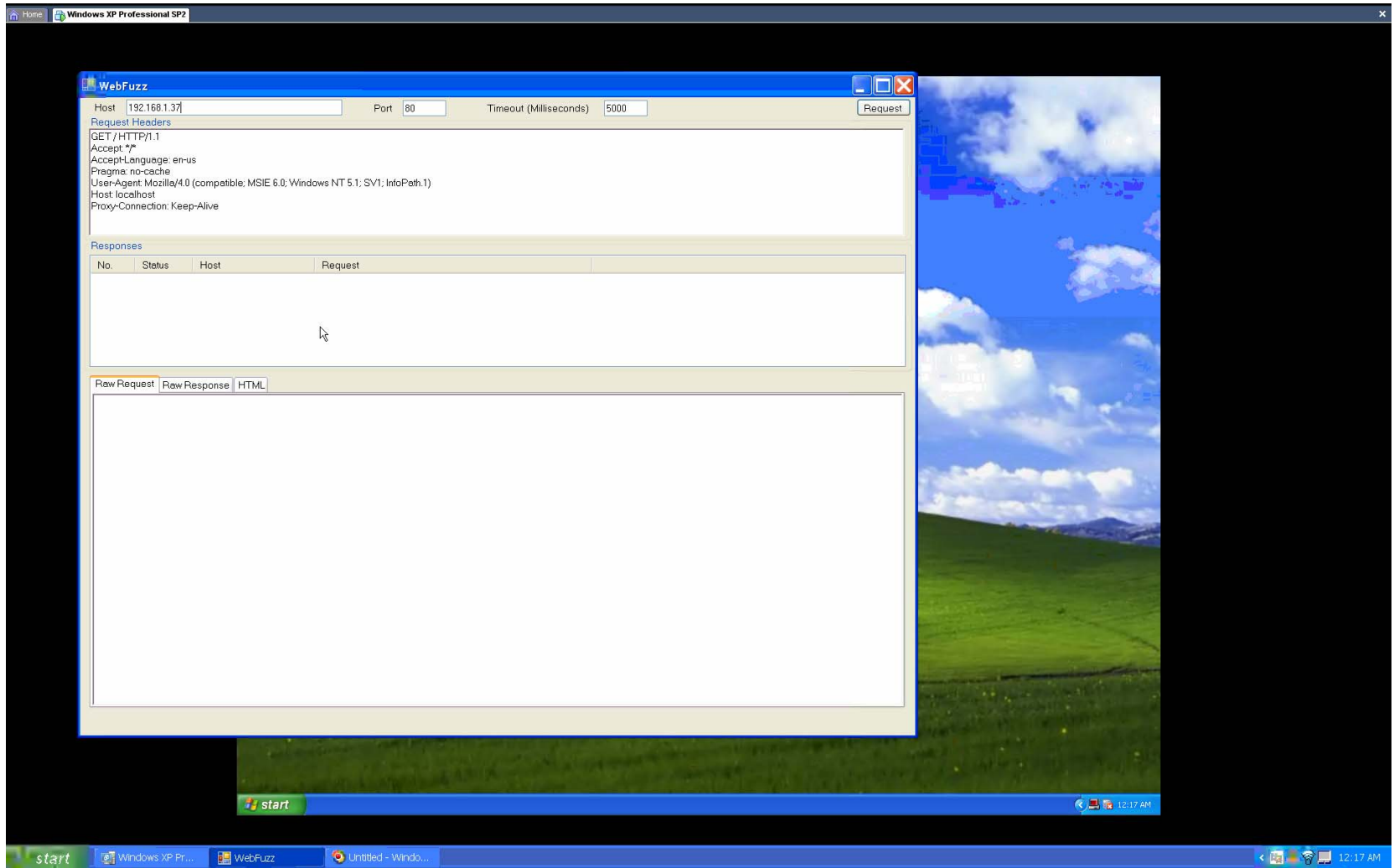
- HTTP
- HTML
- Client side scripting
- SQL

## + Vulnerability types

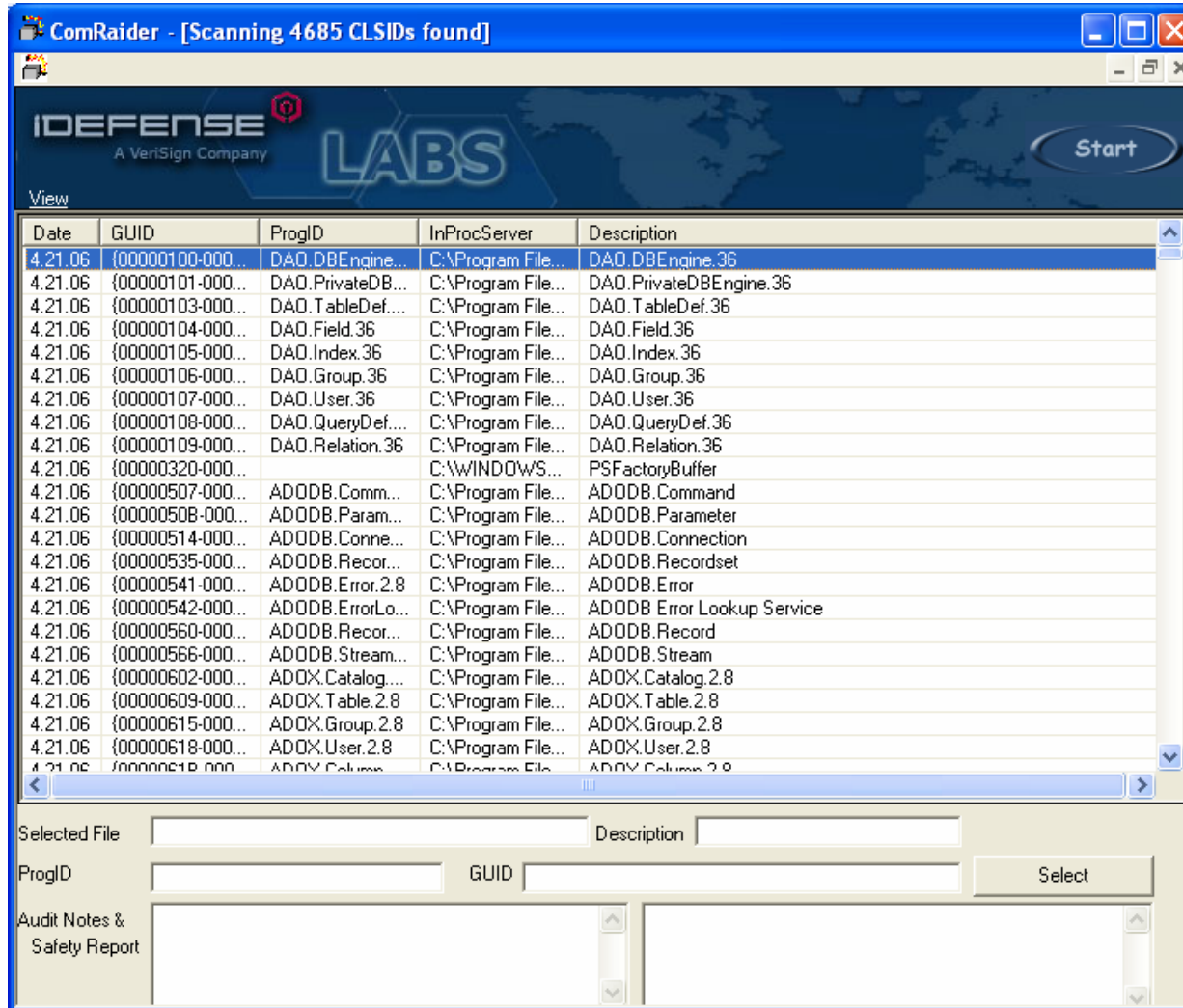
- Denial of service
- Cross site scripting (XSS)
- SQL injection
- Directory traversal/Weak access control
- Weak authentication
- Weak session management (cookies)
- Buffer overflow
- Improperly supported HTTP methods
- Remote Command Execution
- Remote Code Injection
- Vulnerable Libraries
- HTTP Request Splitting
- Format Strings



# WebFuzz - Demo

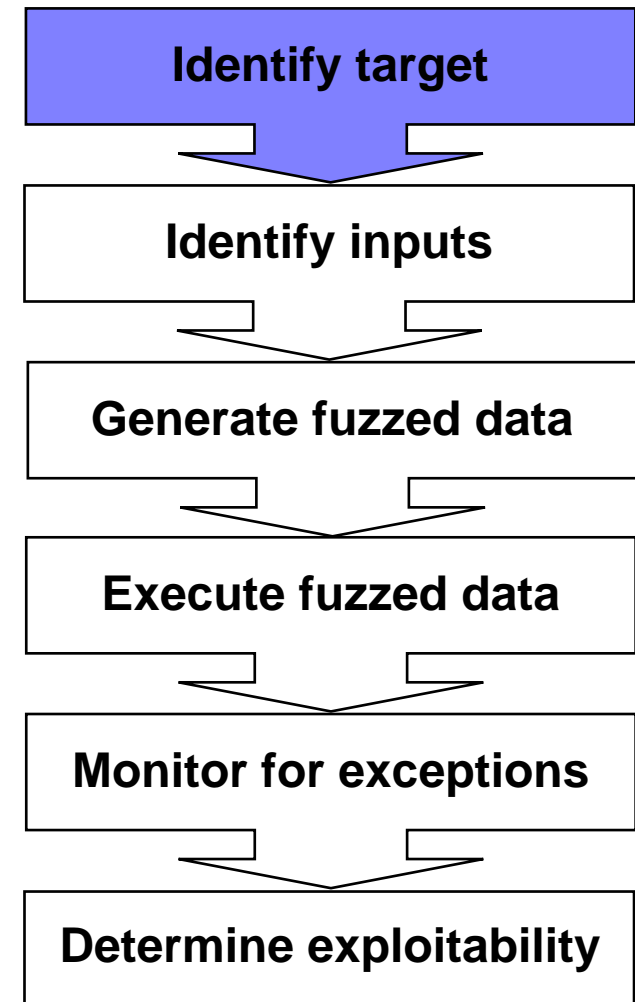


# COMRaider



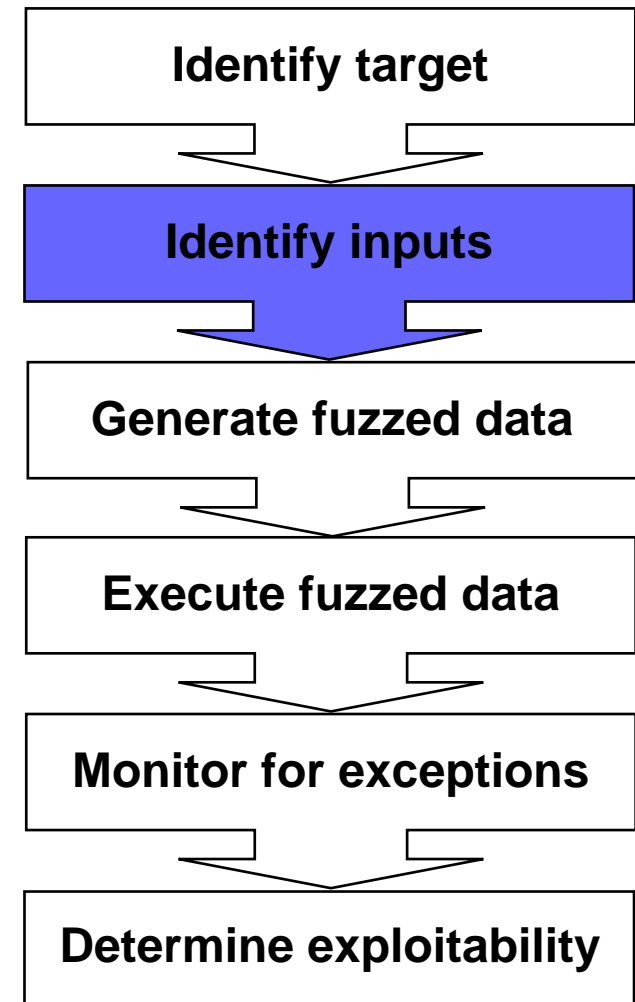
# COMRaider – Identify Target

- + Client side attacks
- + Vendor history
  - Past vulnerabilities
- + High risk targets
  - Popular applications
- + Identify ActiveX controls
  - Choose Active DLL or OCX file directly
  - Scan a directory for registered COM servers
  - Manually enter a GUID
  - Choose from controls that should be loadable in IE



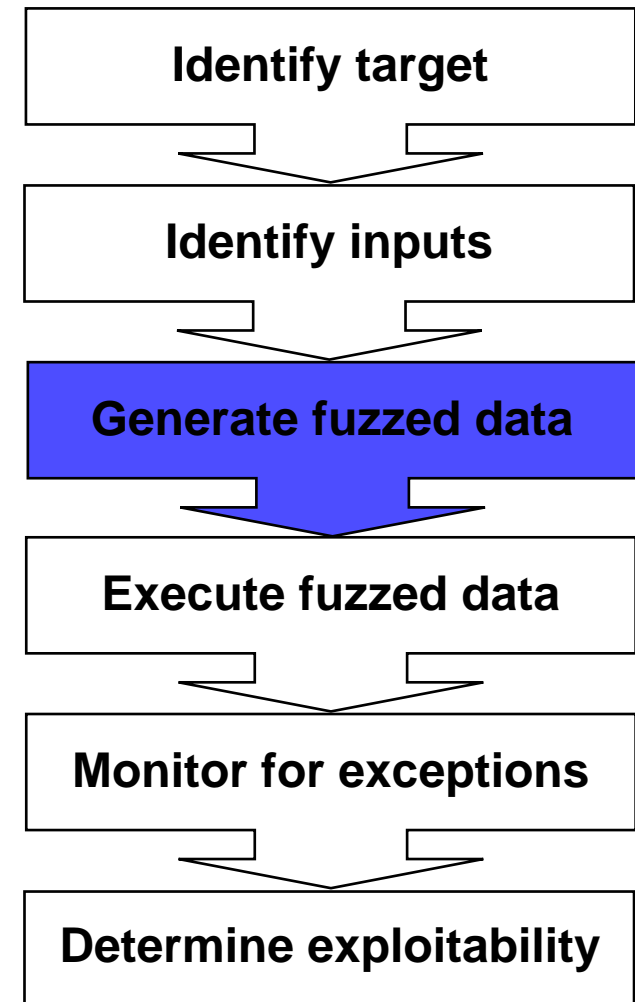
# COMRaider – Identify Inputs

- + Identify fuzzable ActiveX controls
  - Load and parse type library files (\*.tlb) to enumerate interfaces  
or
  - Create a live instance of the object to query and load interface information
- + Scriptable ActiveX controls
  - Accessible by web servers via Internet Explorer
    - Controls marked as Safe for Scripting or implementing IObjectSafety
    - Controls support IDispatch or IDispatchEx interfaces



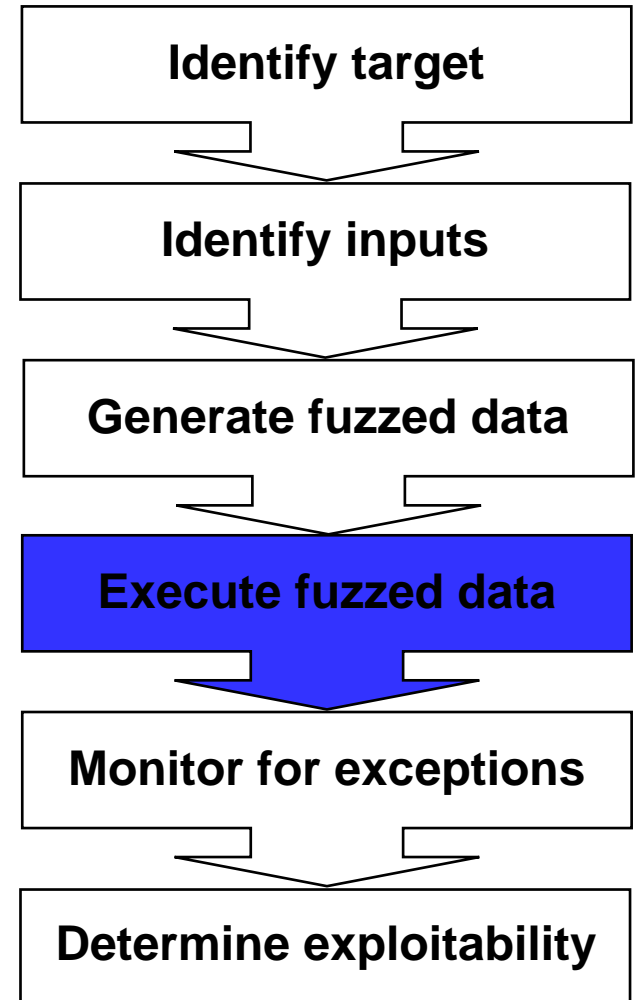
# COMRaider – Generate Fuzzed Data

- + Examine each function and identify variable types to determine fuzzing scenarios
  - **Supported**
    - Ints
    - Longs
    - Doubles
    - Strings
    - Variants
  - **Not supported**
    - Singles
    - Bytes
    - Booleans
- + Dynamically created Windows Script Files (\*.wsf)



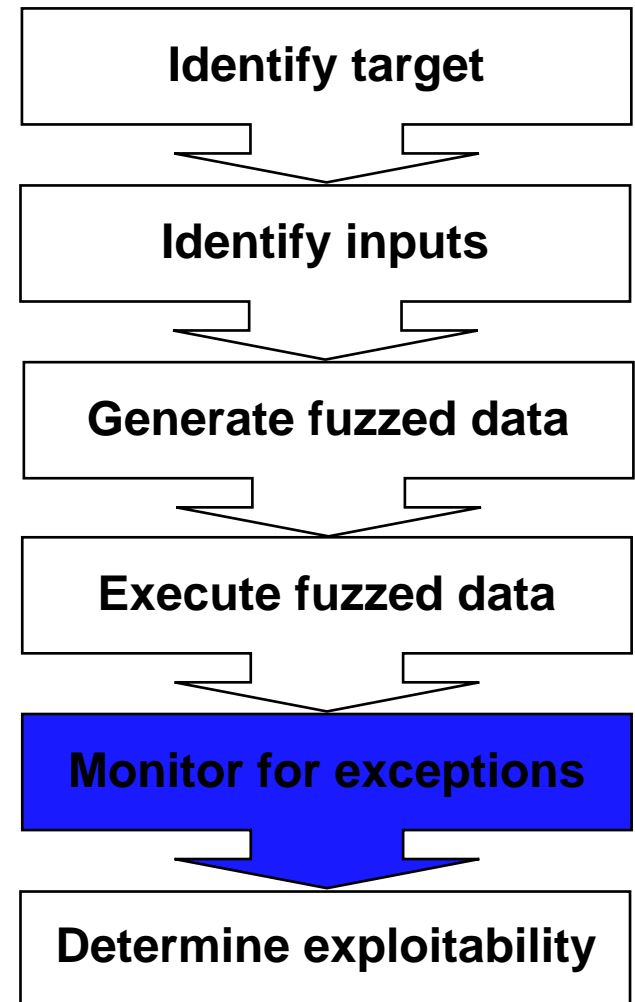
# COMRaider – Execute Fuzzed Data

- + Windows Script Host (wscript.exe) used to execute \*.wsf files



# COMRaider – Monitor for Exceptions

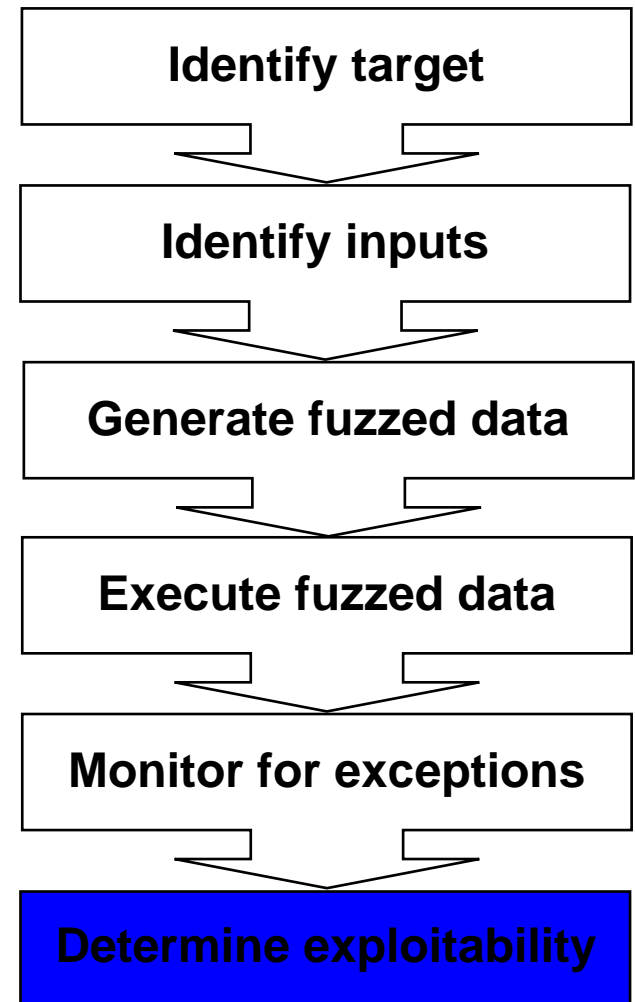
- + Execute
  - Automated and repeated
- + Monitor
  - Debugger - crashmon.dll
    - Record handled/unhandled exceptions
  - Window logger
    - Record/clear error dialogs
    - Record modal windows
- + Kill
  - 8 second timeout



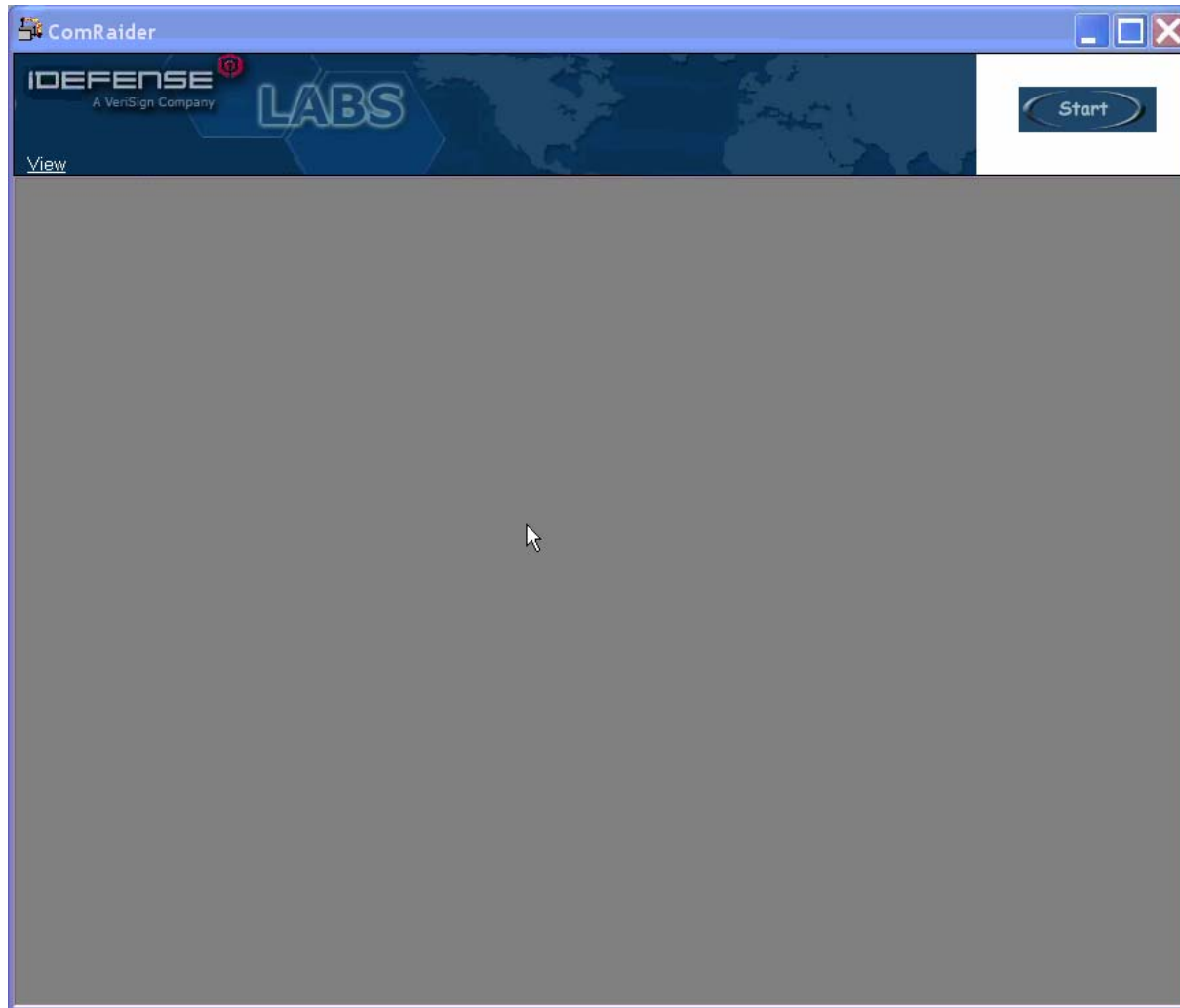


# COMRaider – Determine Exploitability

- + Skills
  - Disassembly
  - Debugging
- + Distributed auditing
  - Audit results uploaded to and downloaded from central MySQL server
- + Exceptions logged
  - Exception code
  - SEH chain
  - Call stack
  - Register values
  - Recent/future opcodes
  - Argument dump
  - Stack dump



# COMRaider - Demo



# Advanced Topics

---

- + Fuzzing Frameworks
- + Automated structure identification
- + Fuzzer tracking (code coverage)
- + Intelligent exception detection and processing

# The Future of Fuzzing

---

## + Tools

- Frameworks
- Integrated test environments
- Commercial tools

## + People

- Wider audience
- Proactive fuzzing – the shift from offense to defense

# Questions

