VeriSign

iDEFENSE
A VeriSign Company

# Fuzzing
## Brute Force Vulnerability Discovery

Michael Sutton
Director, iDefense Labs

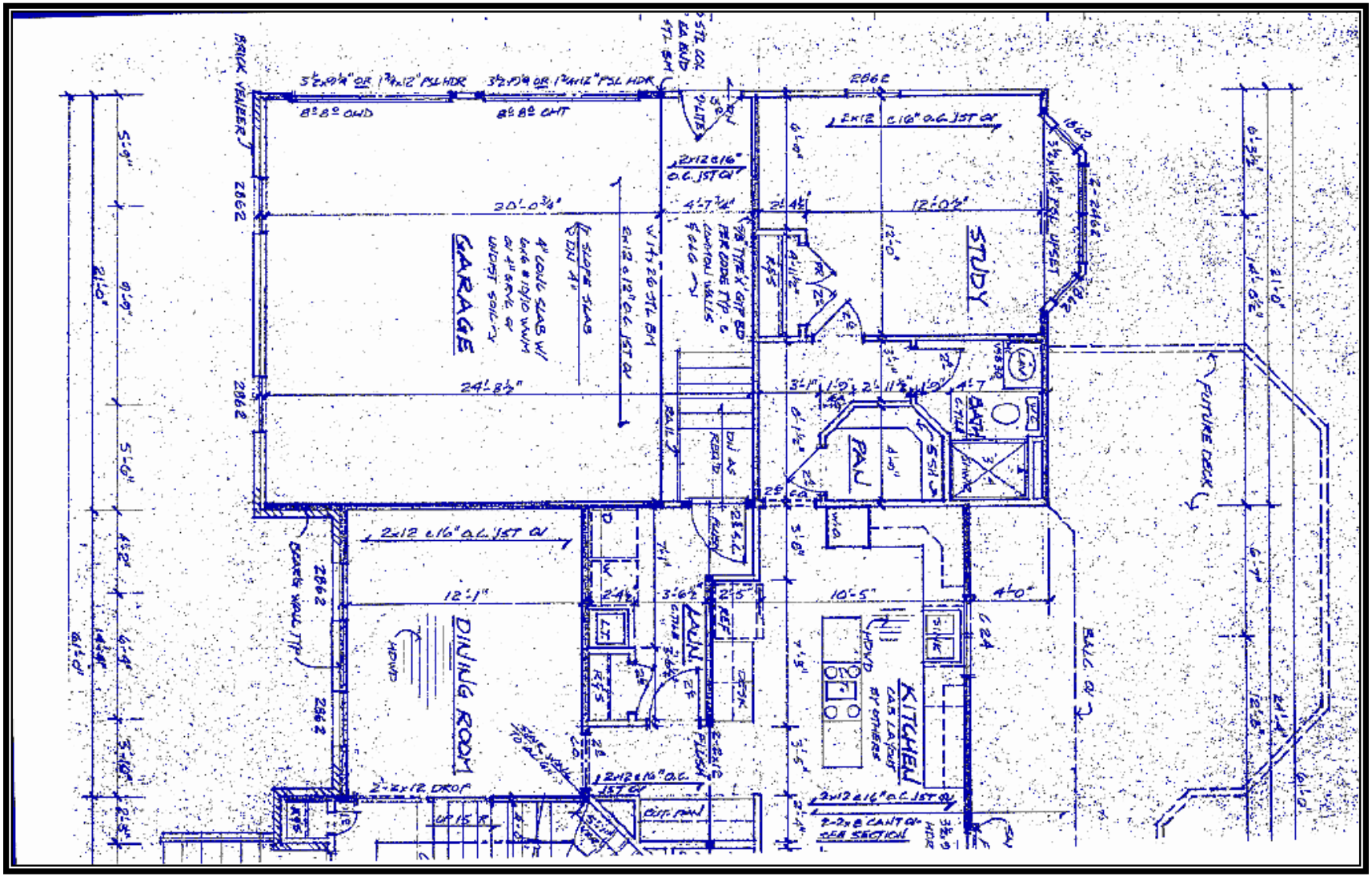
msutton@idefense.com

Where it all comes together.

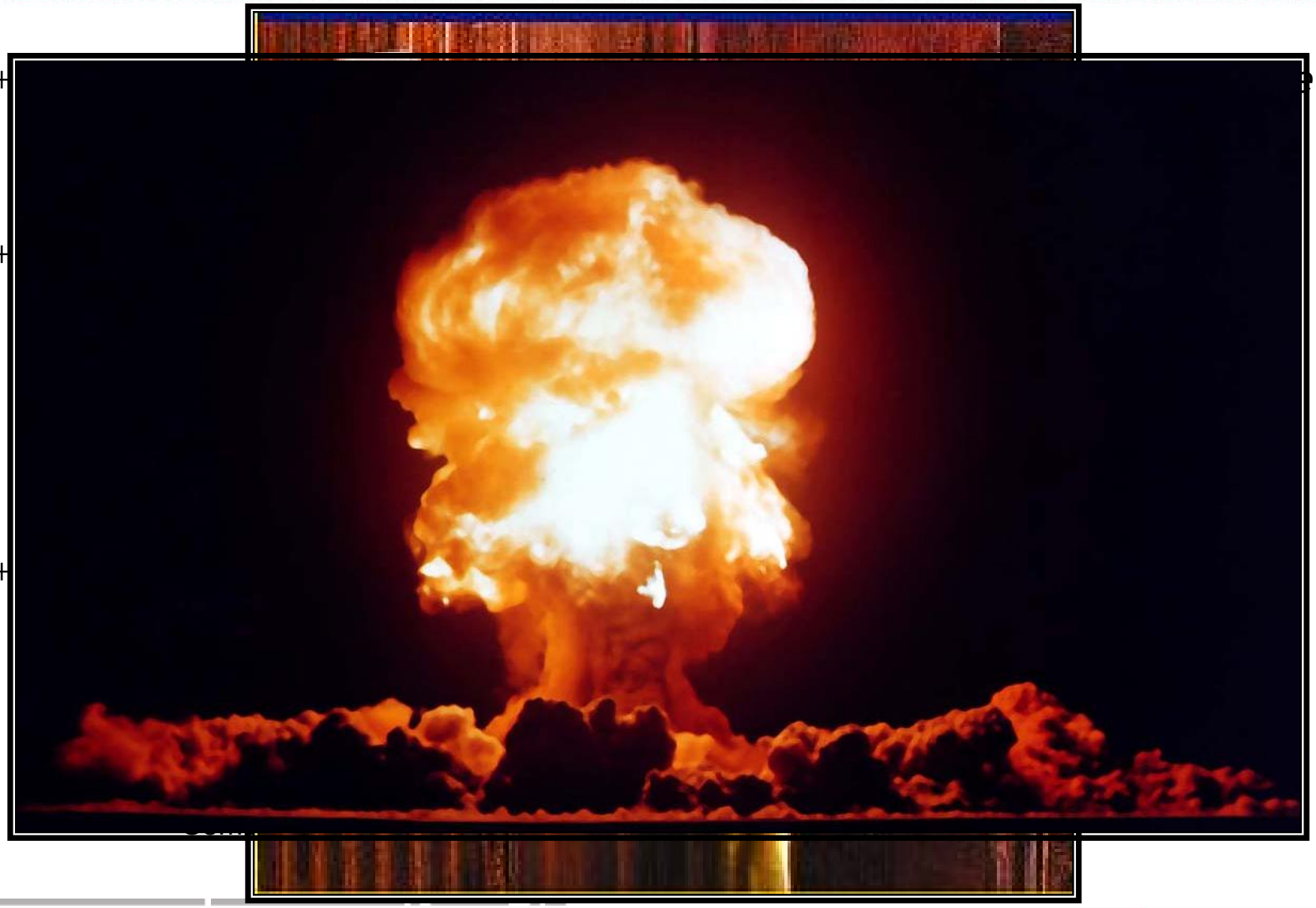
# Agenda

- 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 – White Box



# Vulnerability Discovery Methodologies – Black Box



# What is Fuzzing?

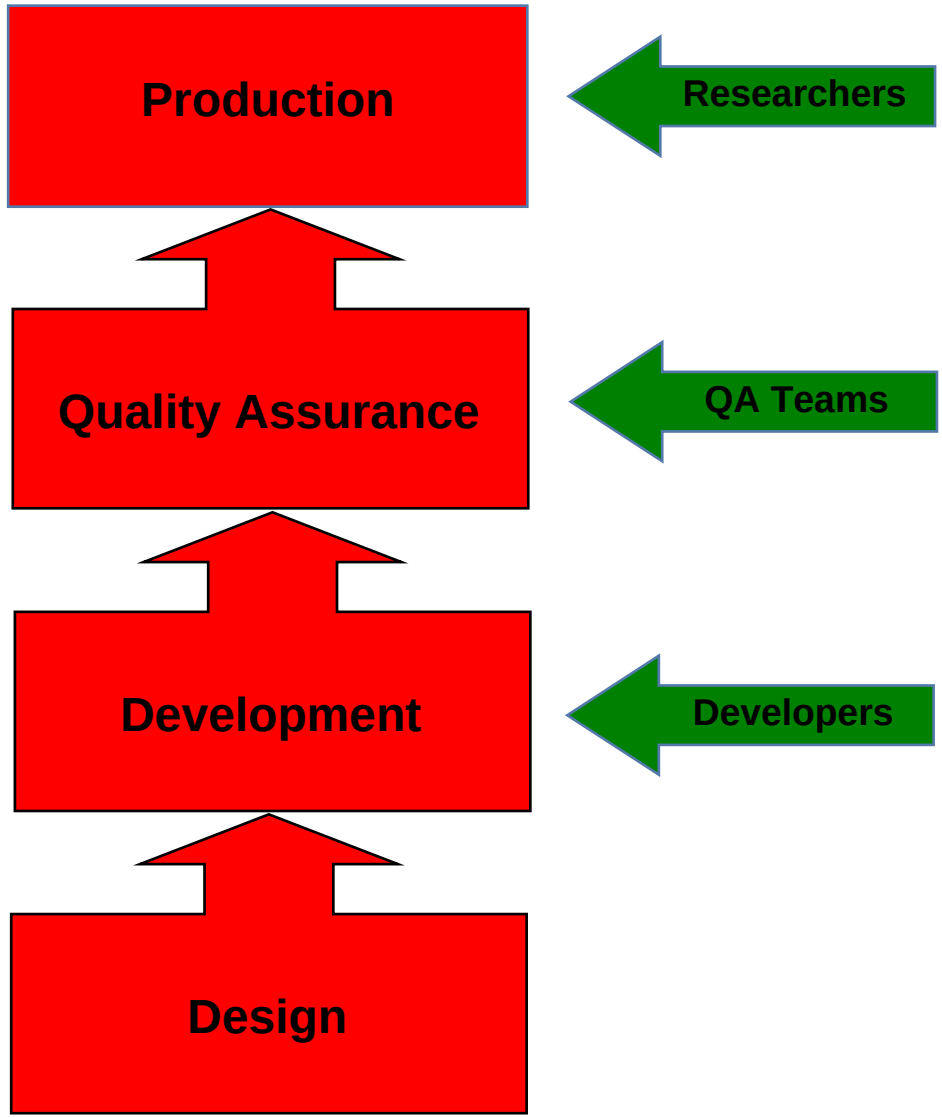
- "Fuzz t… …sic idea is to attach t… …zz"). If the progra… …de asserti…

  The gre… …xtremely simple,…
  - Wik…

- "Unexp…
  - Mic…

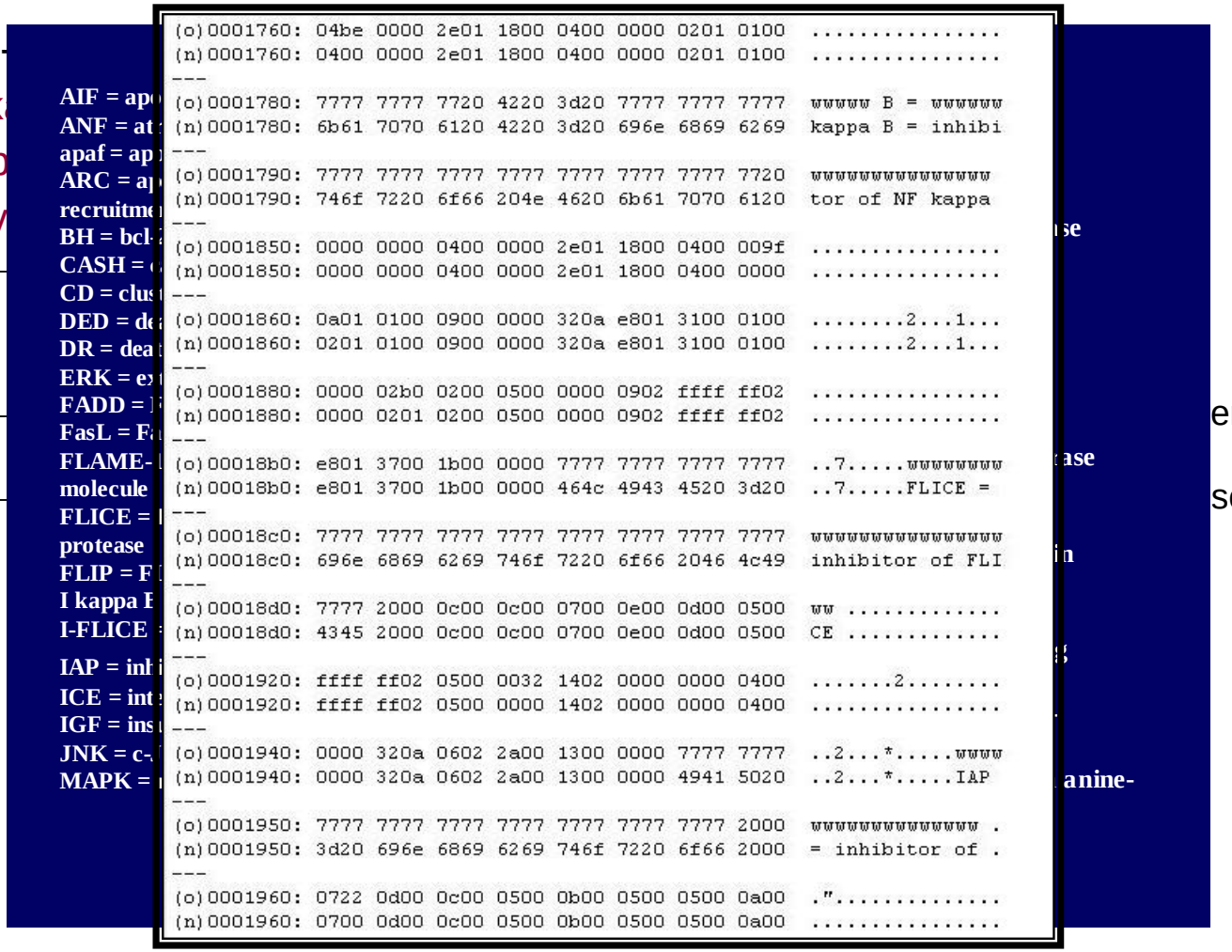
# Who should fuzz?

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

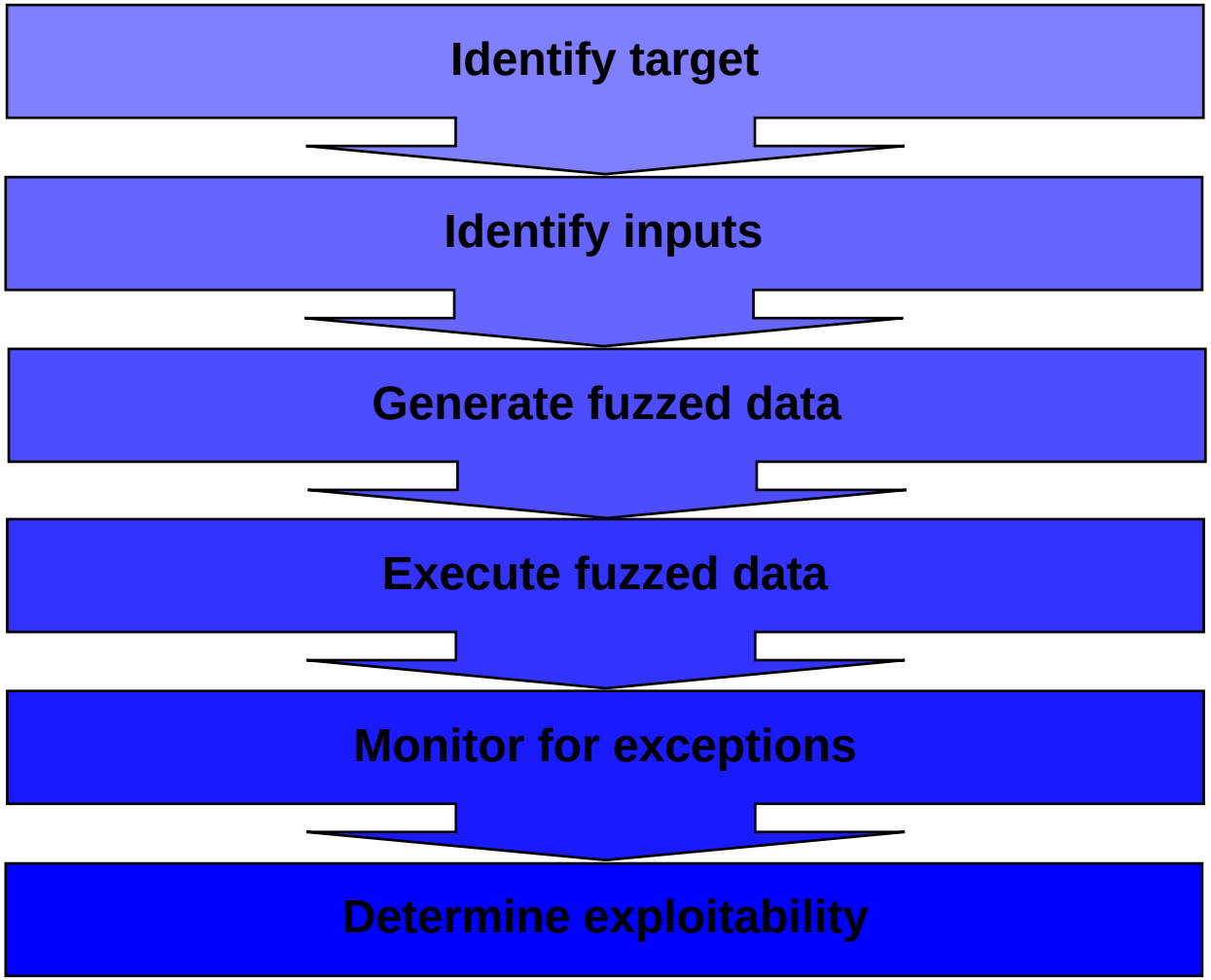


# What can fuzzing do for you?

+ MS06-
  - ak
  - Ap
  - Ev



# Phases



# Fuzzer Classes

- Command line arguments
- Environment variables
  - Sharefuzz (www.immunitysec.com)
- Web applications
  - WebFuzz (Demo)
- File formats
  - FileFuzz (Demo – labs.idefense.com)
- Network protocols
  - SPIKE (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

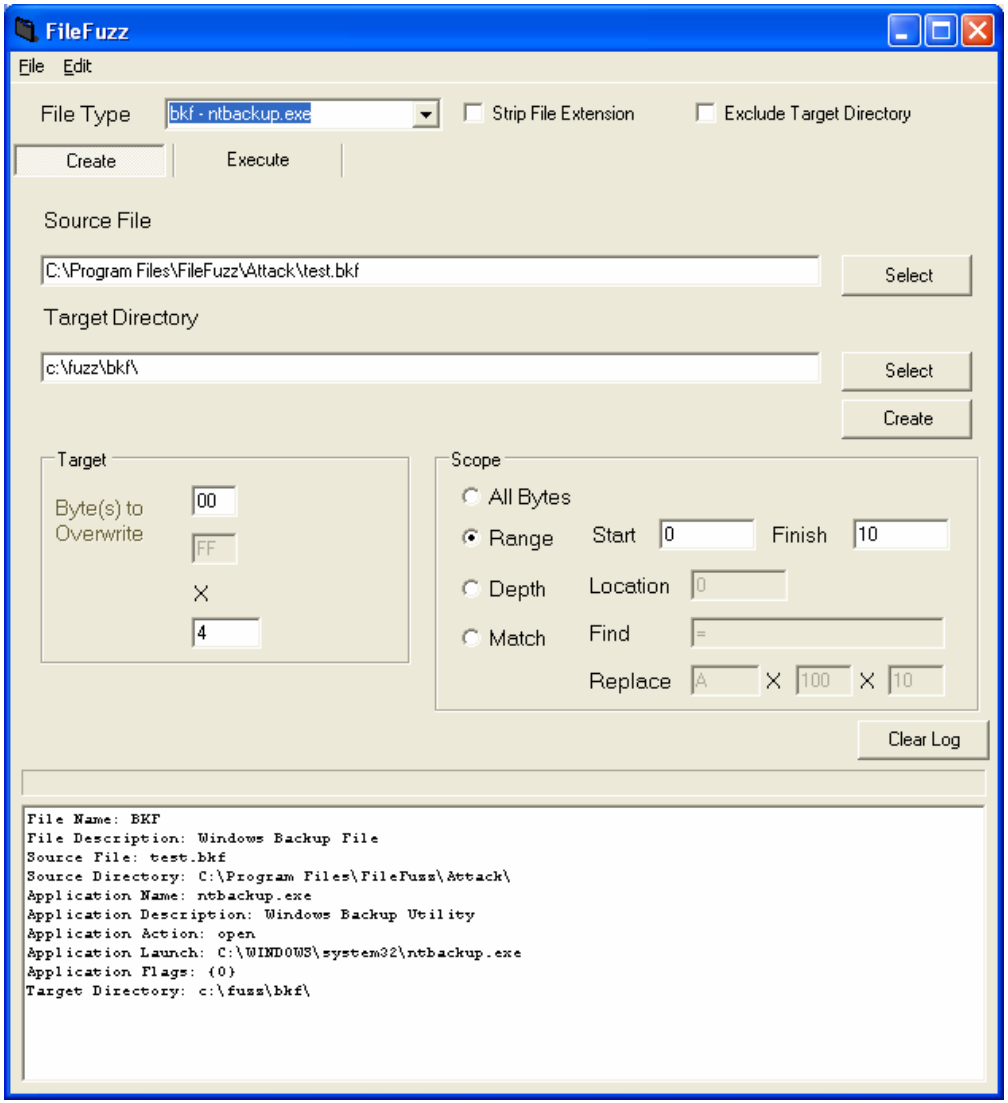
+ Brute force fuzzing

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

# Automation

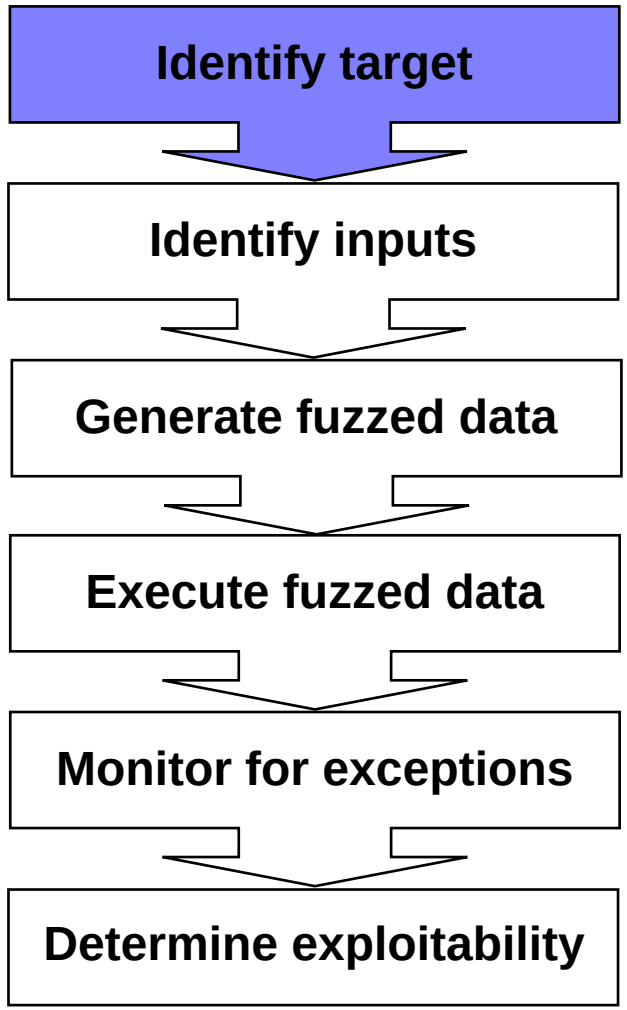
- ‘Intelligent’ fuzzing
  - Approach
    - Templates developed based on protocol definitions
  - Tools
    - SPIKE
    - SPIKEfile
  - Pro
    - Efficient
  - Con
    - Time consuming

# FileFuzz



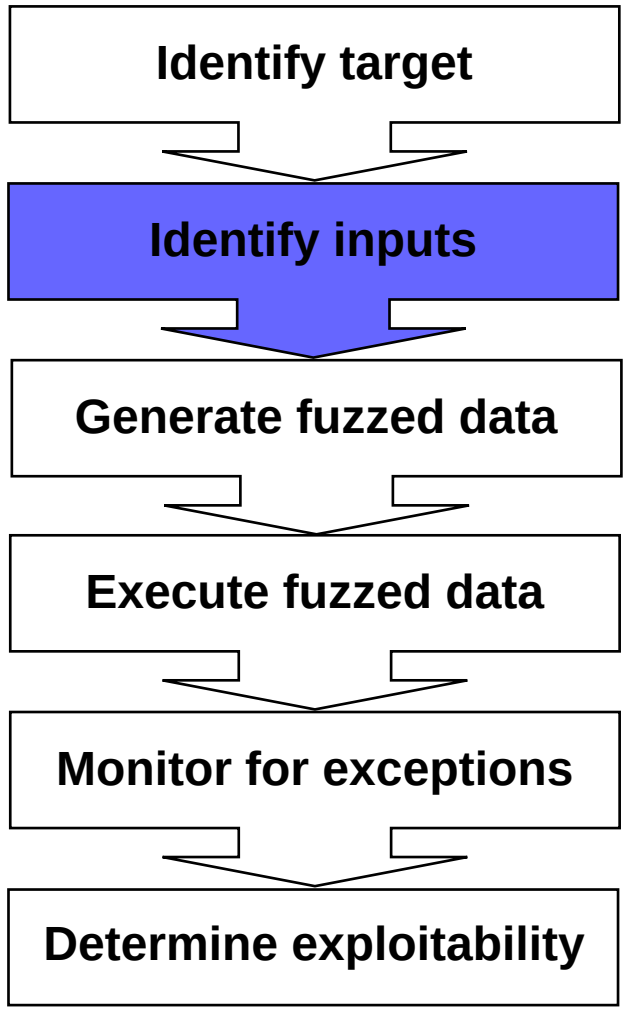
# FileFuzz – Identify Target

- 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

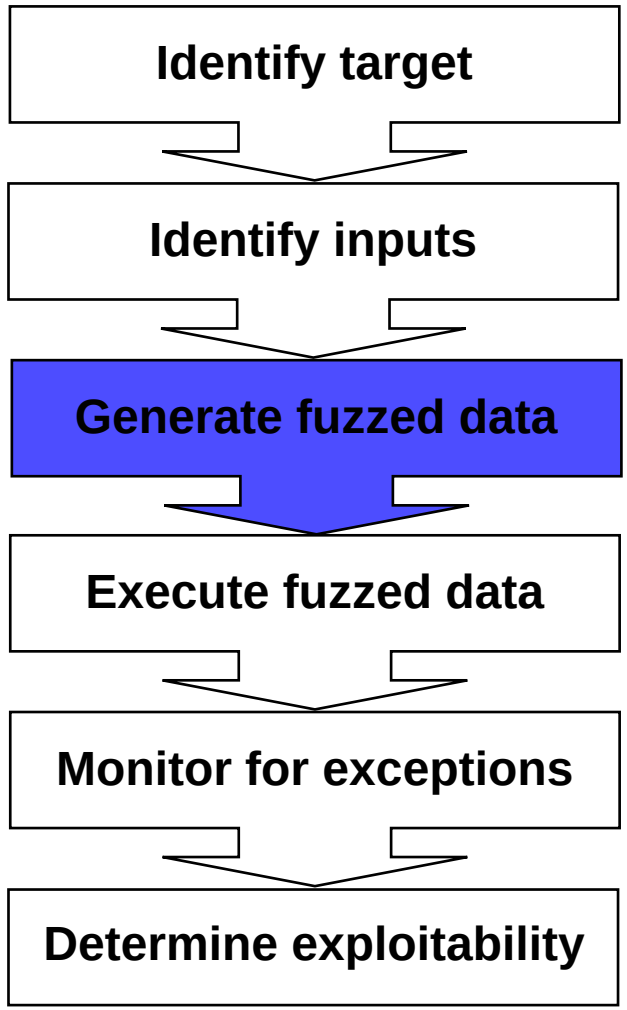
```
FF FF FF FF 00 00 DB FE 0B 00 C5 00 00 01 E8 03 ; ÿÿÿÿ..Ûþ..Å...è.
D7 FF FF FF FF 00 DB FE 0B 00 C5 00 00 01 E8 03 ; ×ÿÿÿÿ.Ûþ..Å...è.
D7 CD FF FF FF FF DB FE 0B 00 C5 00 00 01 E8 03 ; ×ÍÿÿÿÿÛþ..Å...è.
```

  - Depth
    - Determine level of control/influence

```
D7 CD FD 9A 00 00 DB FE 0B 00 C5 00 00 01 E8 03 ; ×Íý..Ûþ..Å...è.
D7 CD FE 9A 00 00 DB FE 0B 00 C5 00 00 01 E8 03 ; ×Íþš..Ûþ..Å...è.
D7 CD FF 9A 00 00 DB FE 0B 00 C5 00 00 01 E8 03 ; ×Íÿš..Ûþ..Å...è.
```

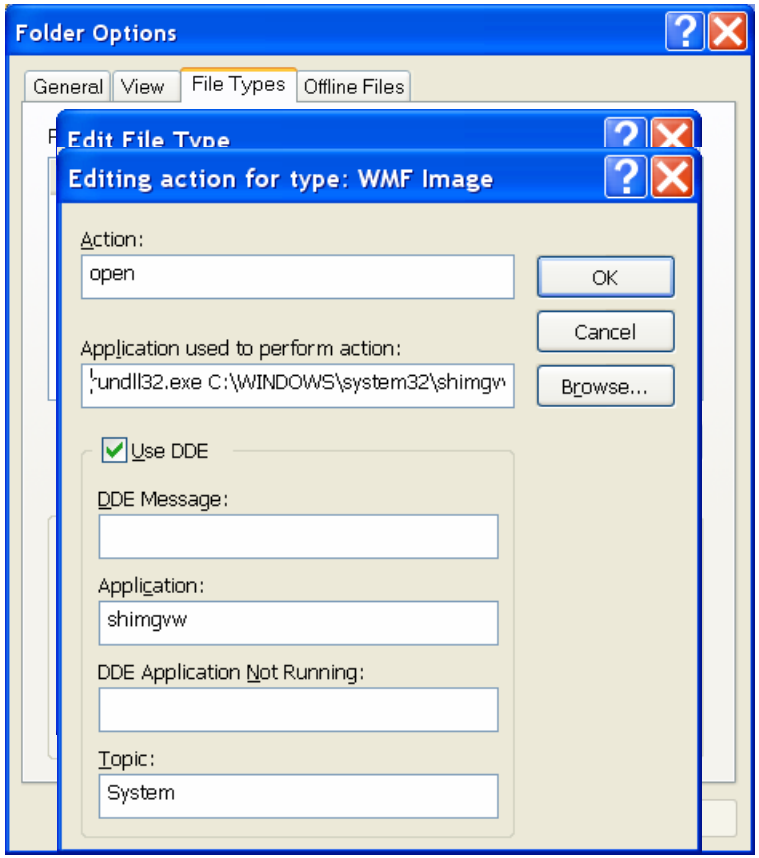
+ Text Files
  - name = value

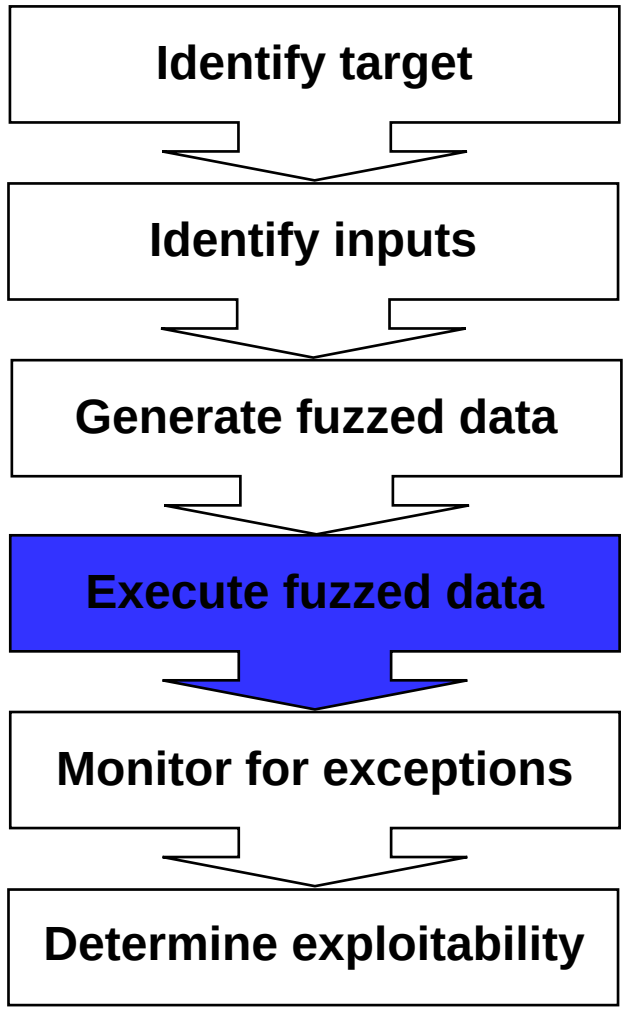
```
file_size = 10
file_size = AAAAA
file_size = AAAAAAAAAA
```



# FileFuzz – Execute Fuzzed Data

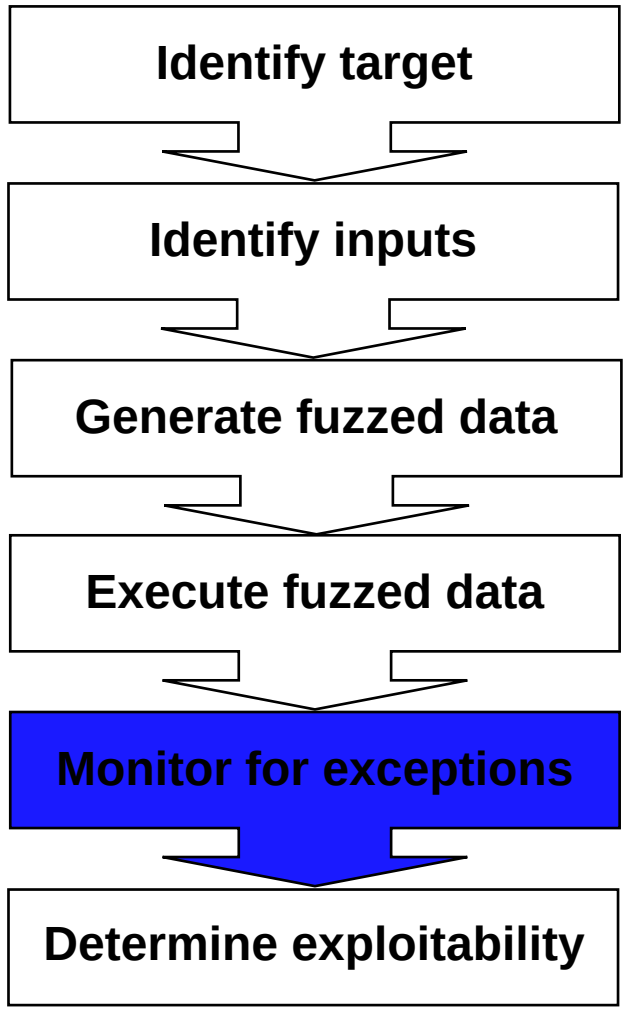
+ Command line arguments
  - Windows explorer
    - Tools…Folder Options…File Types



# FileFuzz – Monitor for Exceptions

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

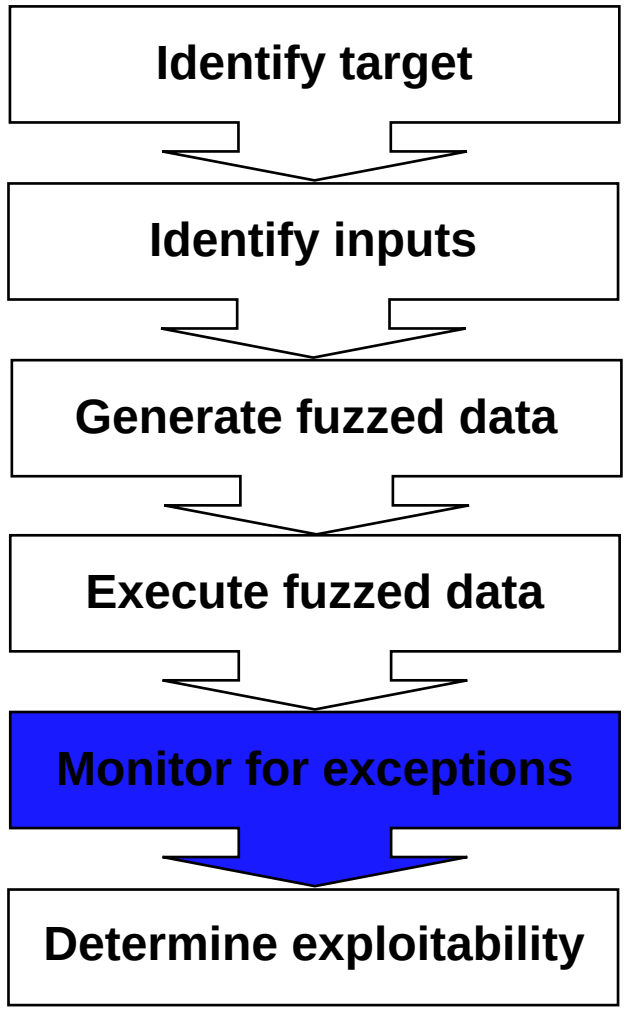


# FileFuzz – Monitor for Exceptions

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

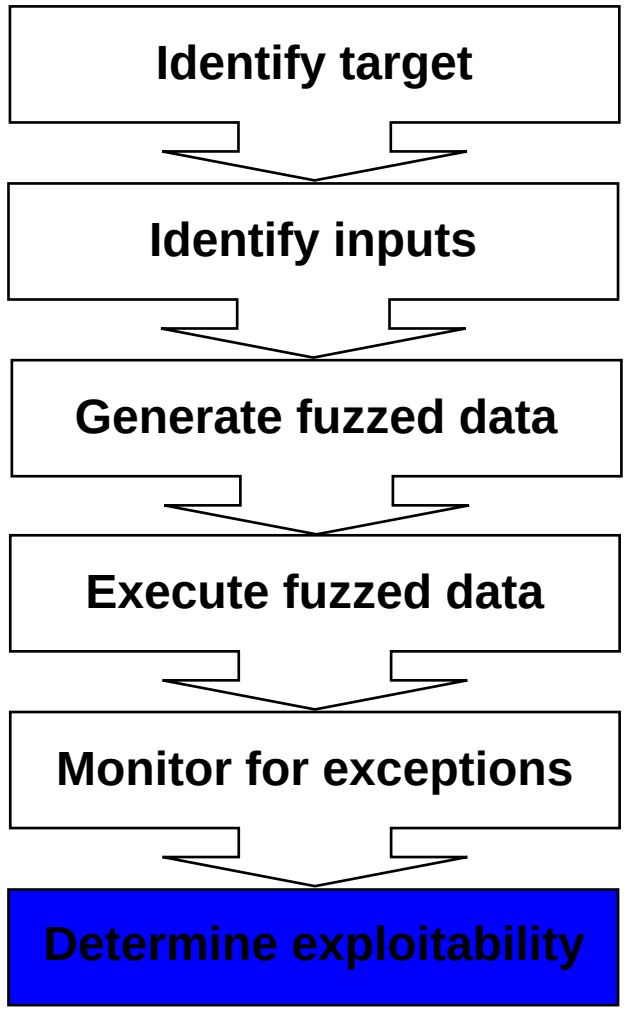
```
[*] "crash.exe" "C:\Program Files\WordPerfect Office
12\Programs\UA120.exe" 2000 /qt c:\fuzz\ast\8.ast
[*] Access Violation
[*] Exception caught at 00403f06 mov eax,[eax+edi*4]
[*] EAX:0014b1b8 EBX:00000005 ECX:00435c00 EDX:0012fbac
[*] ESI:00435c00 EDI:cccccccc ESP:0012fab8 EBP:0012fae8
```

+ Kill
  - Set timeout

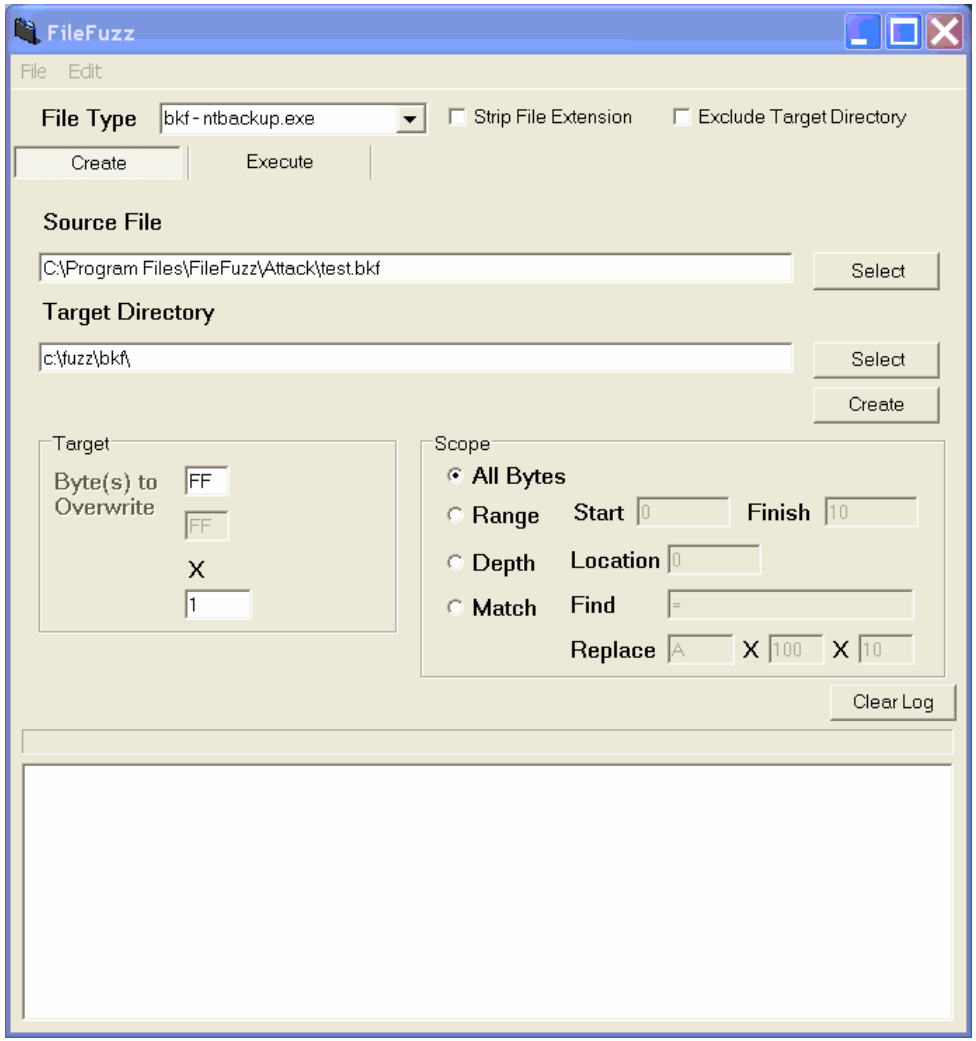


# FileFuzz – Determine Exploitability

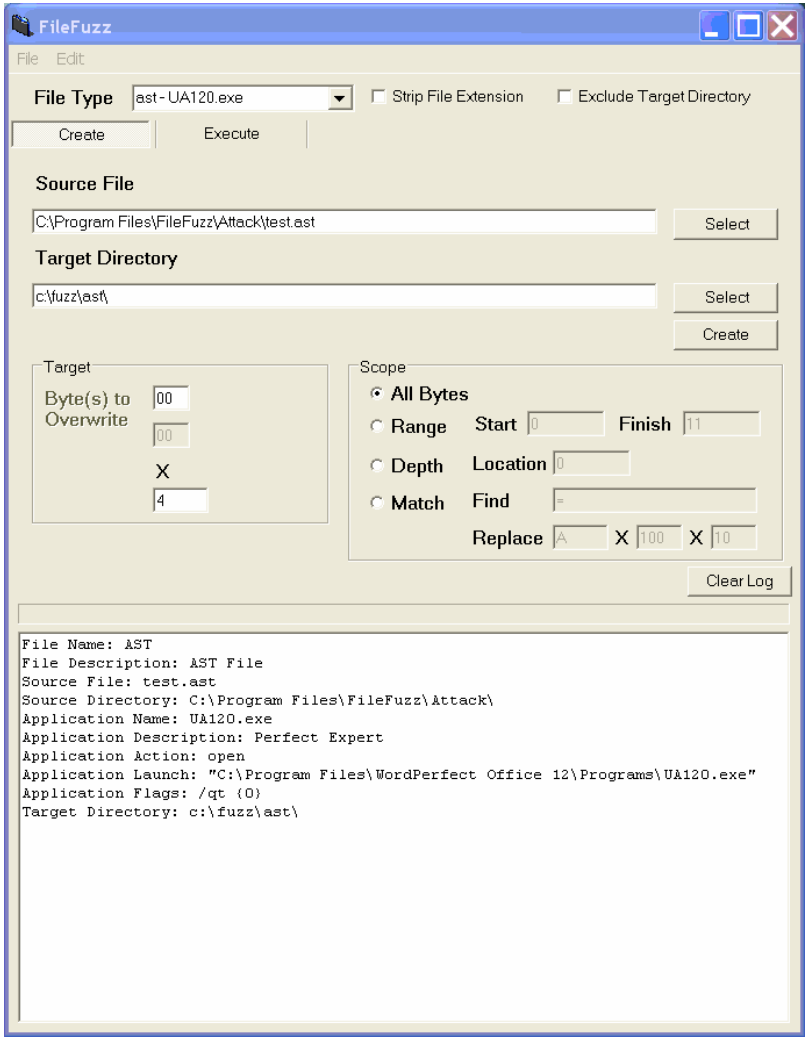
+ Skills
  - Disassembly
  - Debugging
+ Vulnerability types
  - Stack overflows
  - Heap overflows
  - Integer handling
    - Overflows
    - Signedness
  - DoS
    - Out of bounds reads
    - Infinite loops
    - NULL pointer dereferences
  - Logic errors
    - Windows WMF vulnerability (MS06-001)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



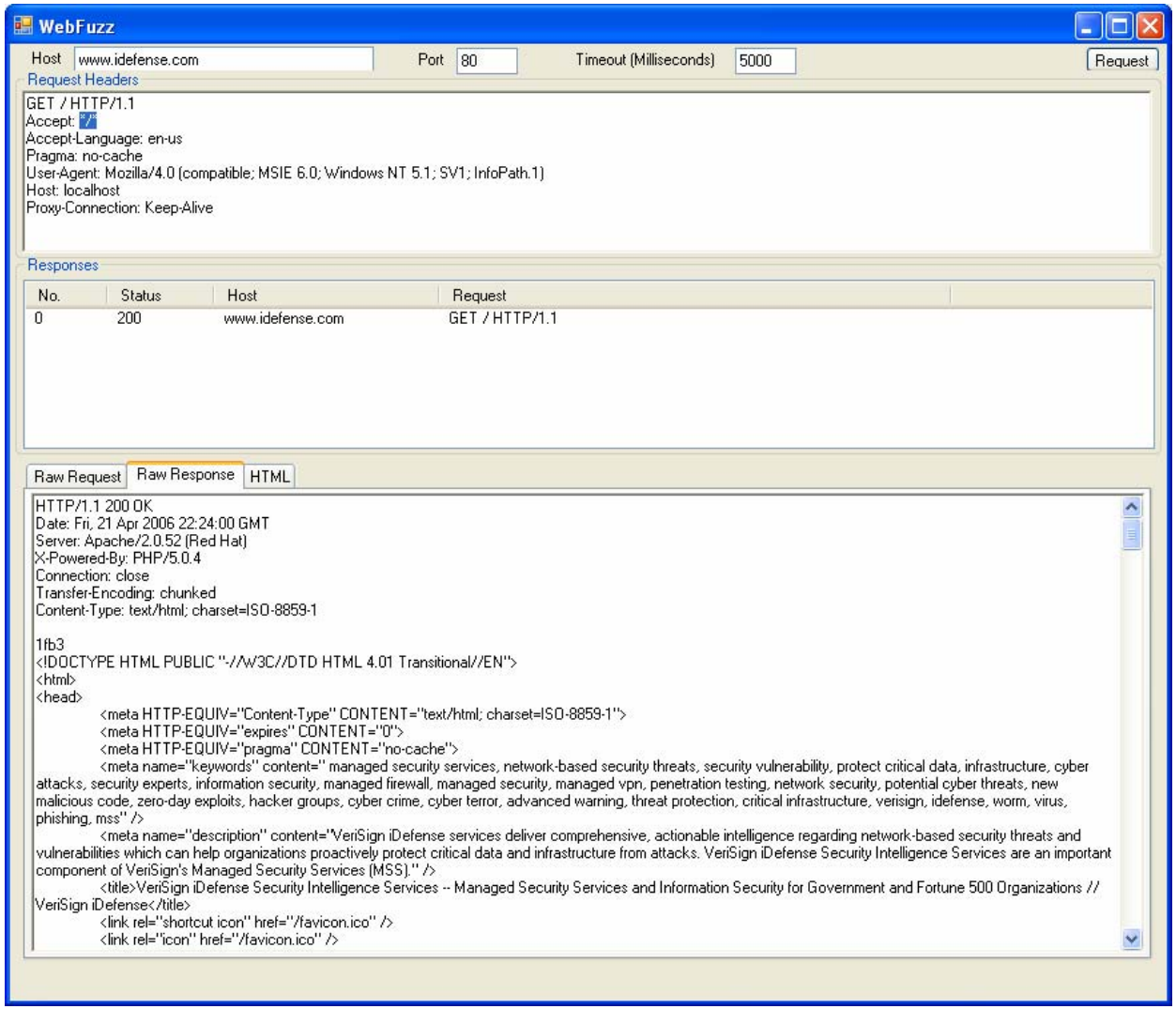
# FileFuzz – Demo (Breadth)



# FileFuzz – Demo (Depth)

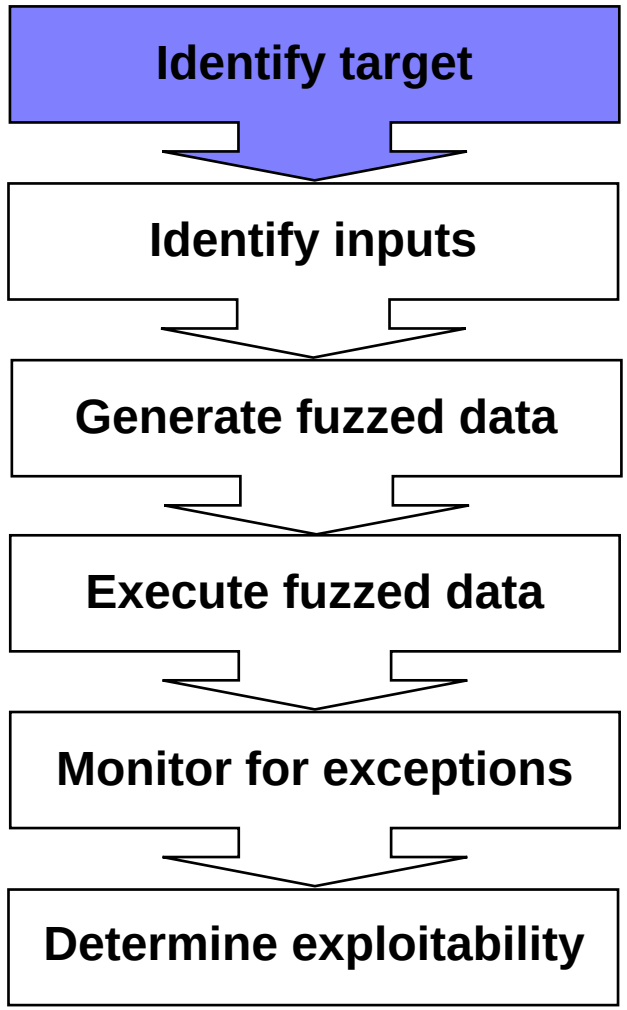


# WebFuzz



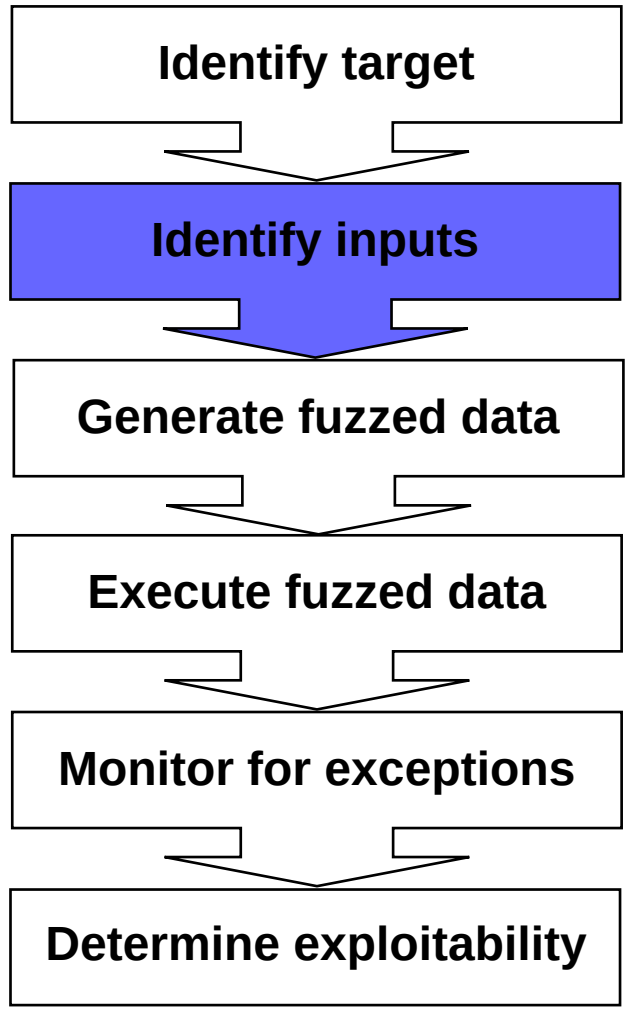
# WebFuzz – Identify Target

- 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

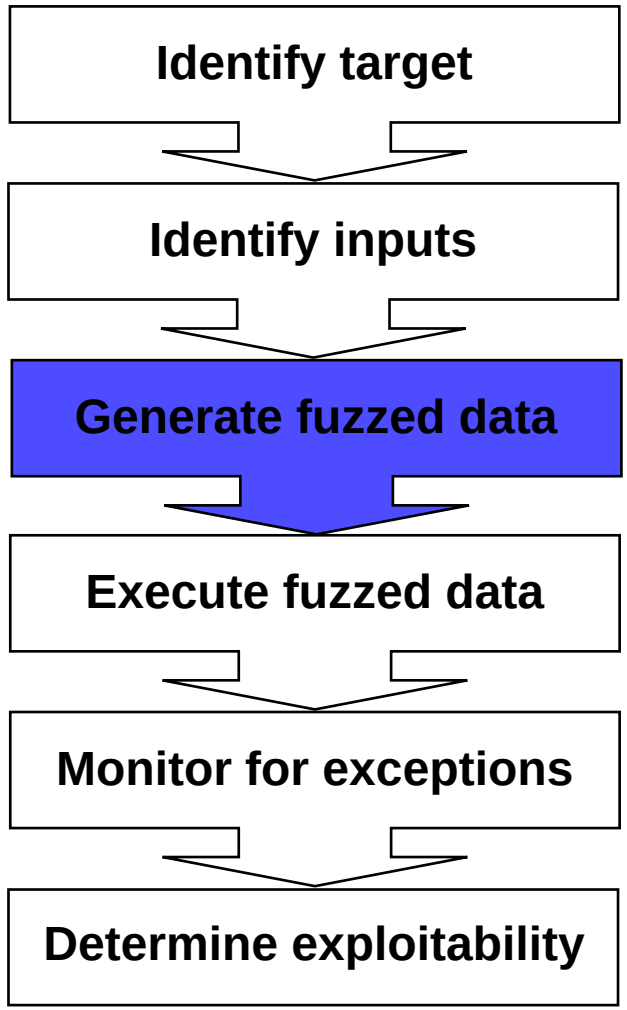
+ Intelligent fuzzing
  - Start with legitimate web request
  - Build template to mutate requests

+ Request format

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

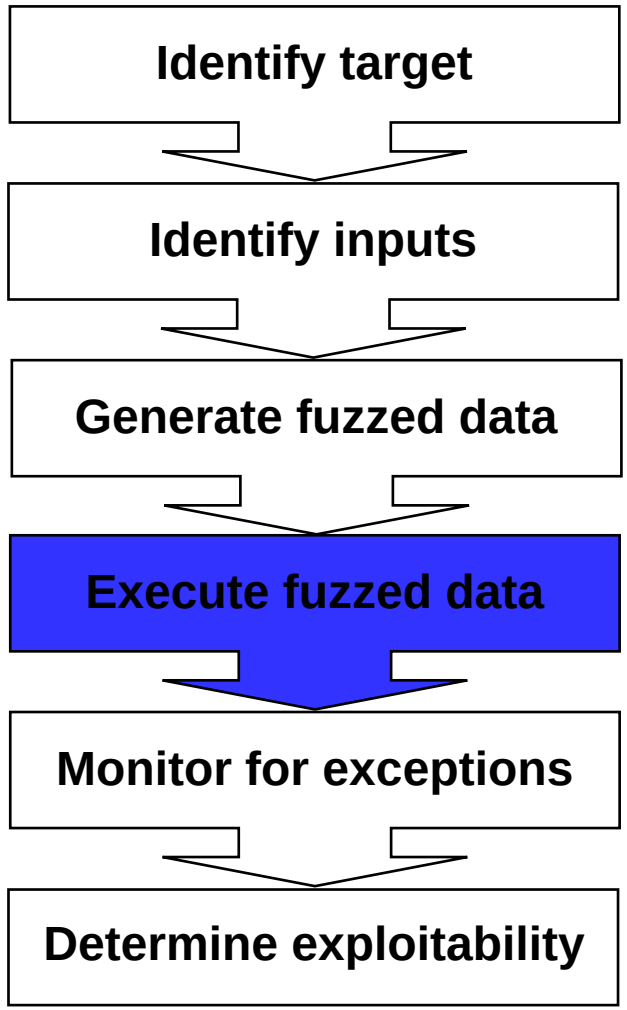
+ Fuzz Template

```
[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
```



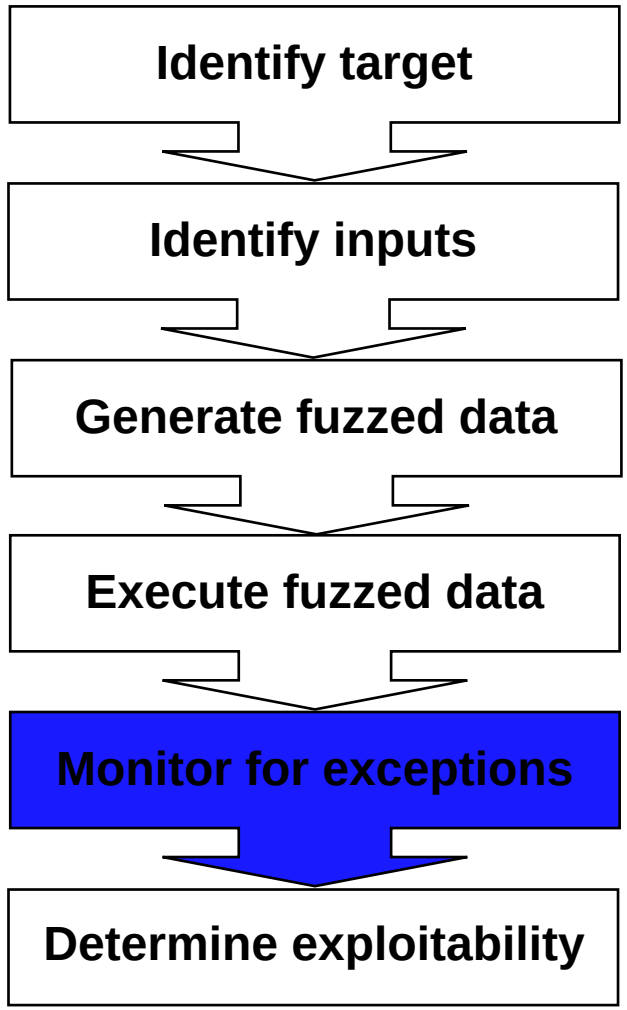
# WebFuzz – Execute Fuzzed Data

+ Fuzz classes
  - Directory traversal
  - Format strings
  - Overflow
  - SQL Injection
  - XSS Injection



# WebFuzz – Monitor for Exceptions

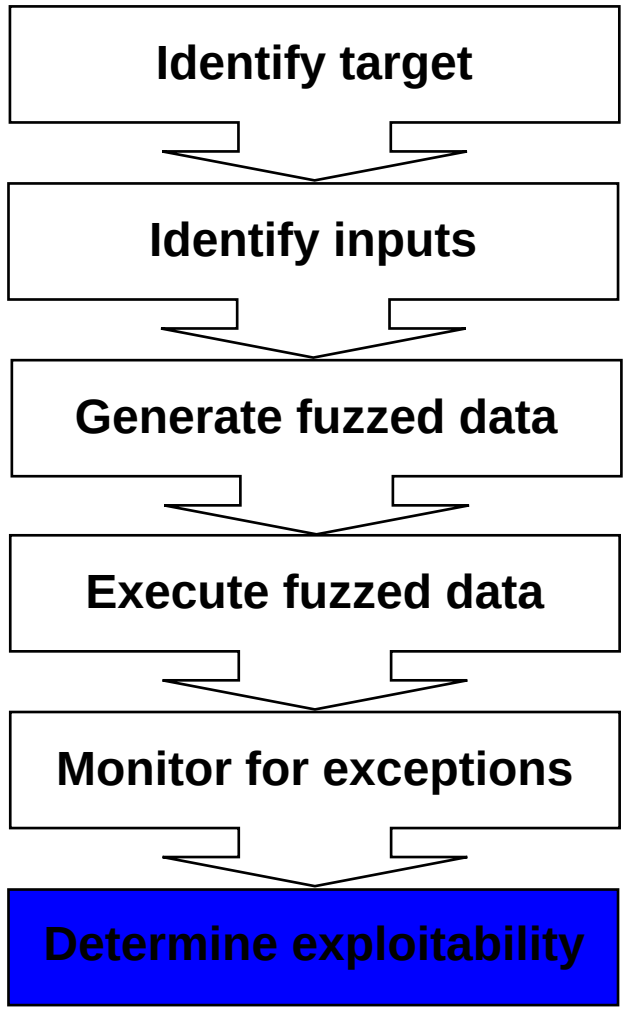
- Execute
  - Automated and repeated
- Monitor
  - HTML response
    - Error messages
  - Raw response
    - User input
  - Status codes
- Kill
  - Set timeout



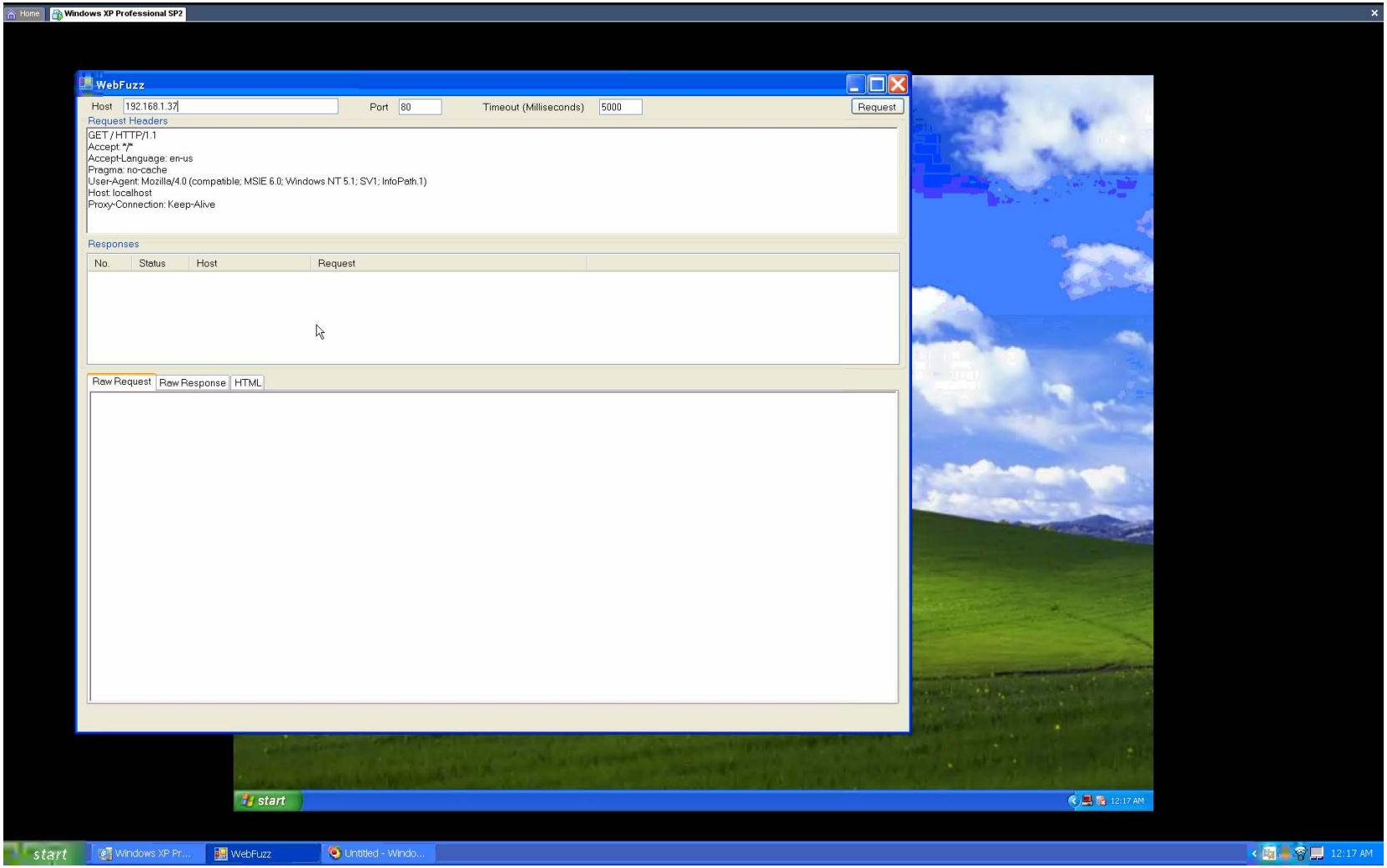
# WebFuzz – Determine Exploitability

+ Skills
  - HTTP
  - HTML
  - Client side scripting
  - SQL

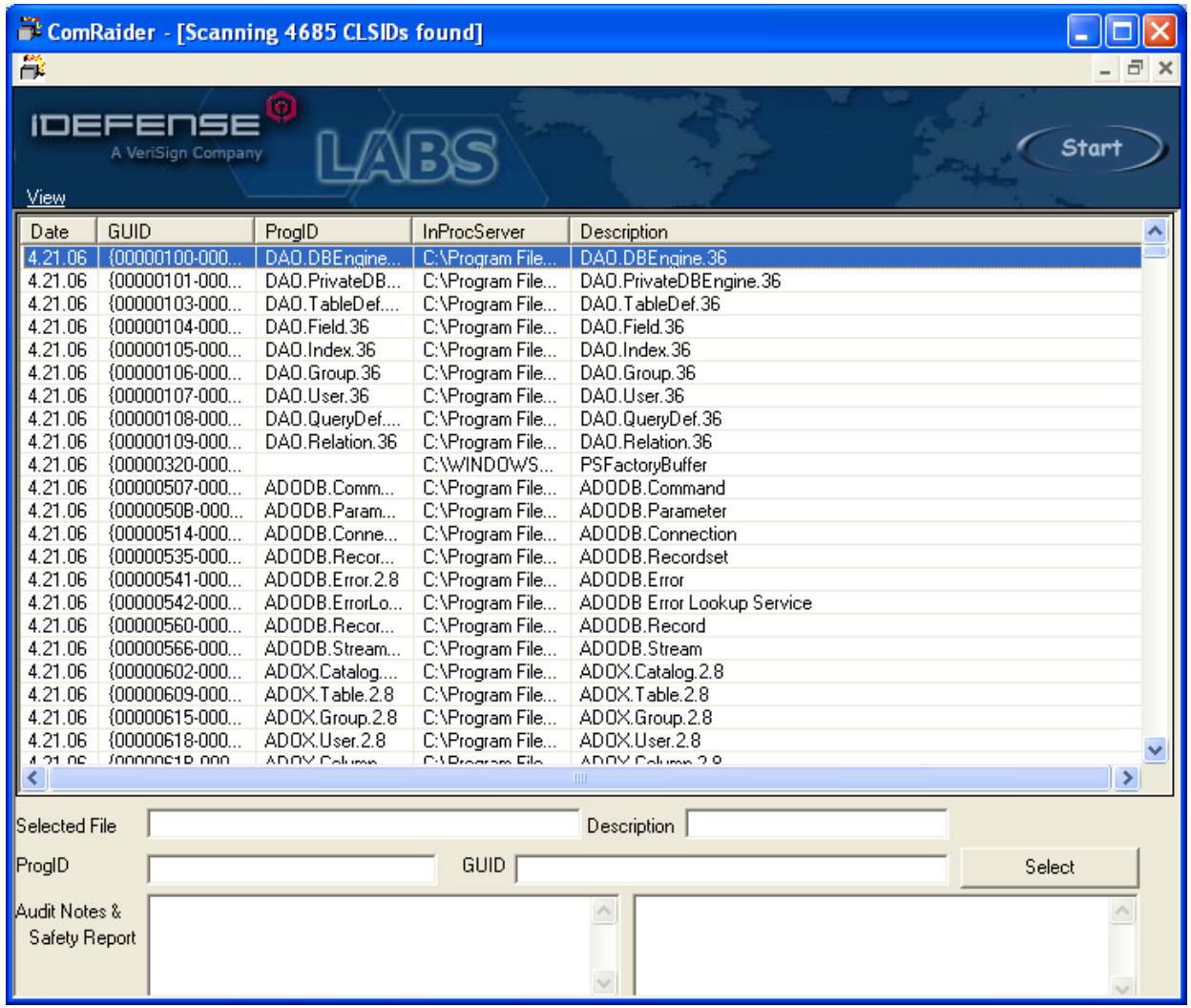
+ Vulnerability types
  - Denial of service
  - Cross site scripting (XSS)
  - SQL injection
  - Directory traversal/Weak access control
  - Weak authentication
  - Weak session management (cookies)
  - Buffer overflow
  - Improperly supported HTTP methods
  - Remote Command Execution
  - Remote Code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



# WebFuzz - Demo

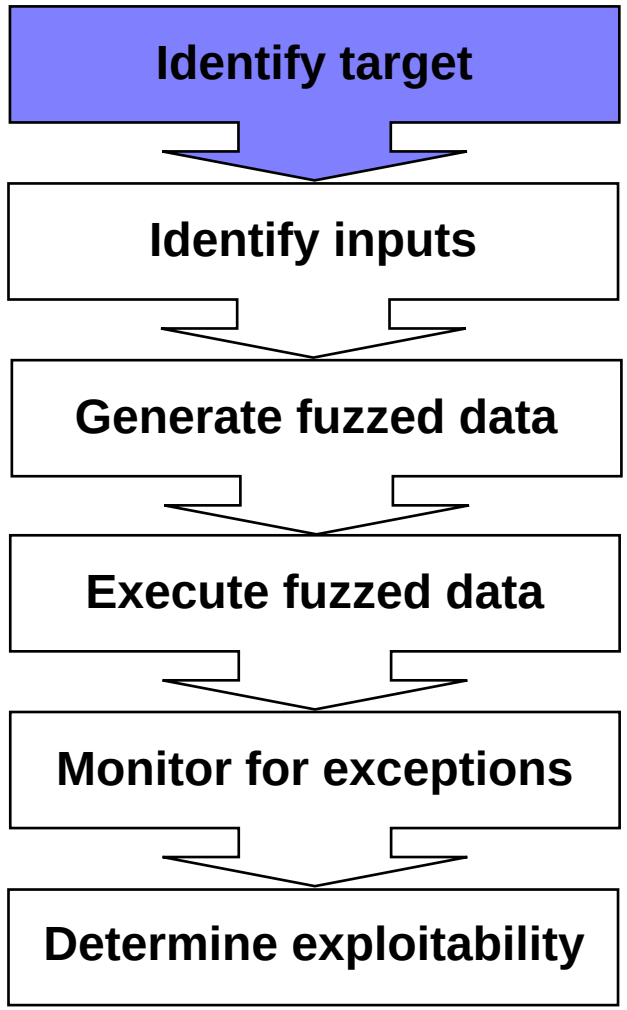


# COMRaider



# COMRaider – Identify Target

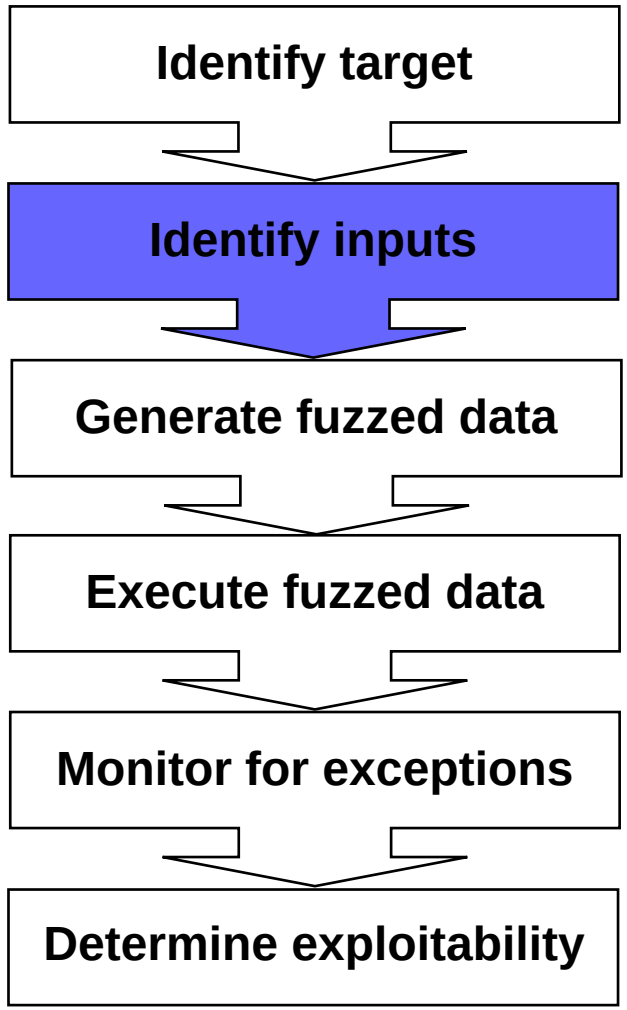
- Client side attacks
- Vendor history
  - Past vulnerabilities
- High risk targets
  - Popular applications
- Identify ActiveX controls
  - Choose Active DLL or OCX file directly
  - Scan a directory for registered COM servers
  - Manually enter a GUID
  - Choose from controls that should be loadable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



# COMRaider – Identify Inputs

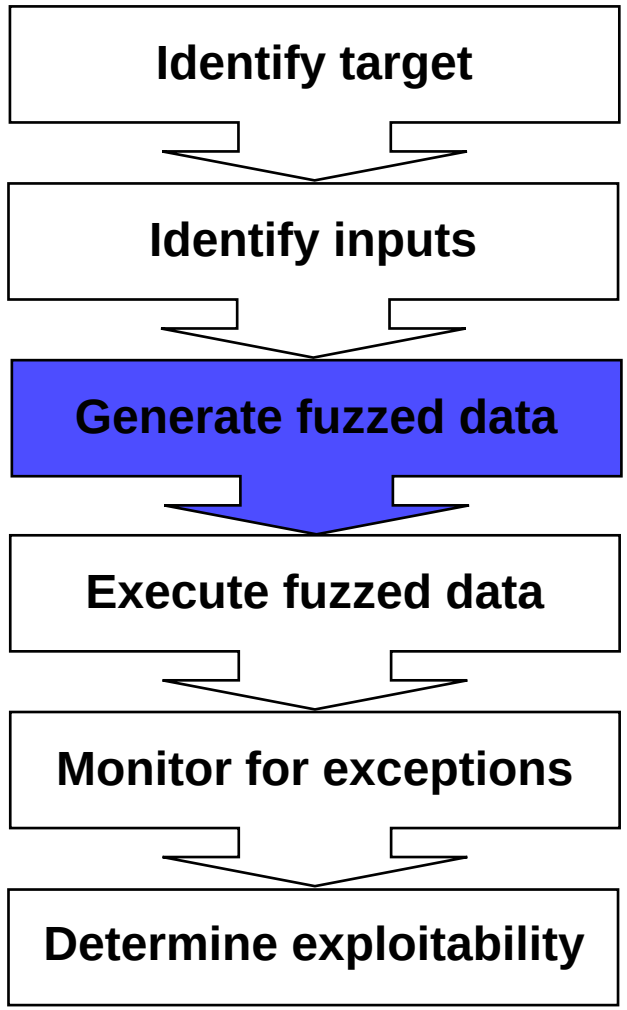
+ Indentify fuzzable ActiveX controls
  - Load and parse type library files (*.tlb) to enumerate interfaces

    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



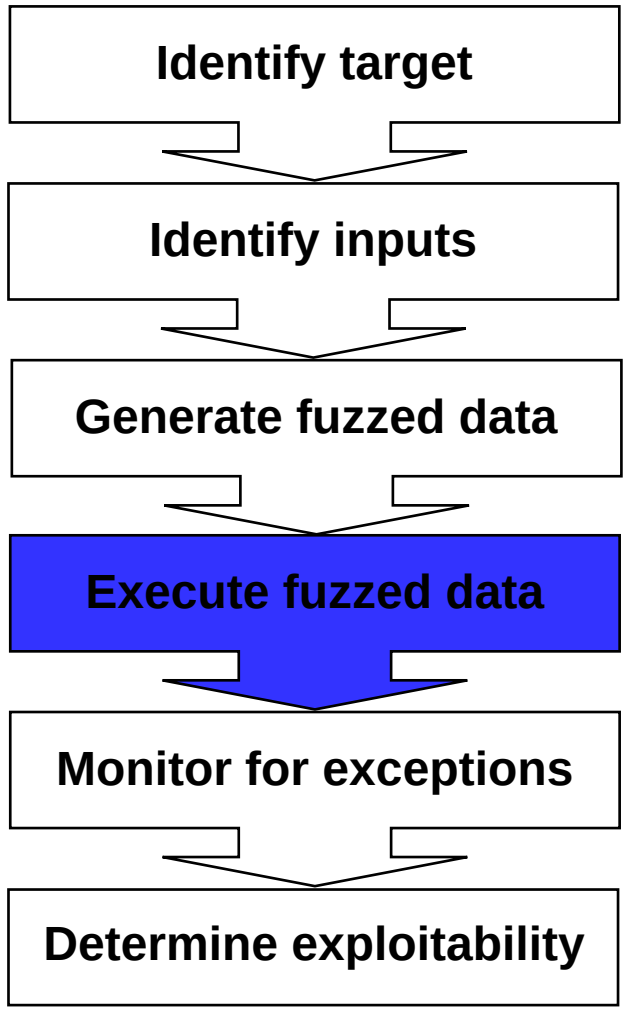
# COMRaider – Generate Fuzzed Data

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



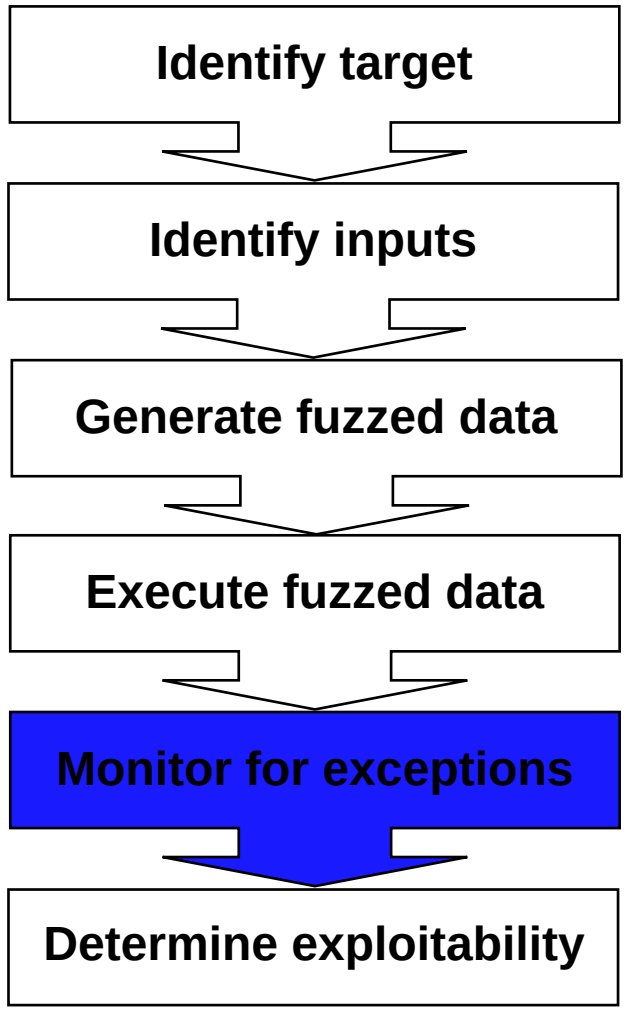
# COMRaider – Execute Fuzzed Data

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



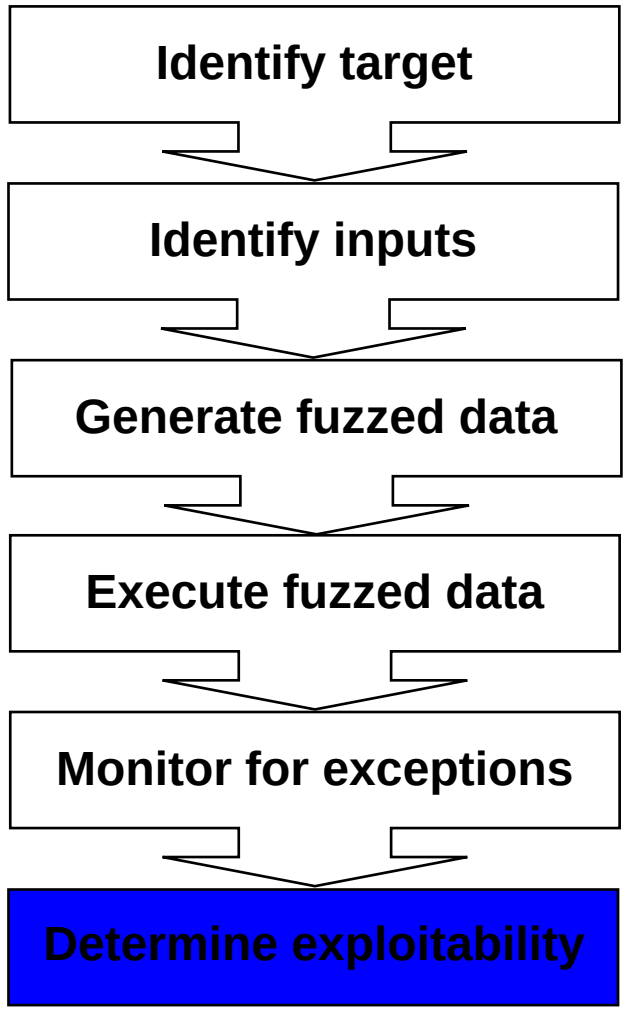
# COMRaider – Monitor for Exceptions

+ 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

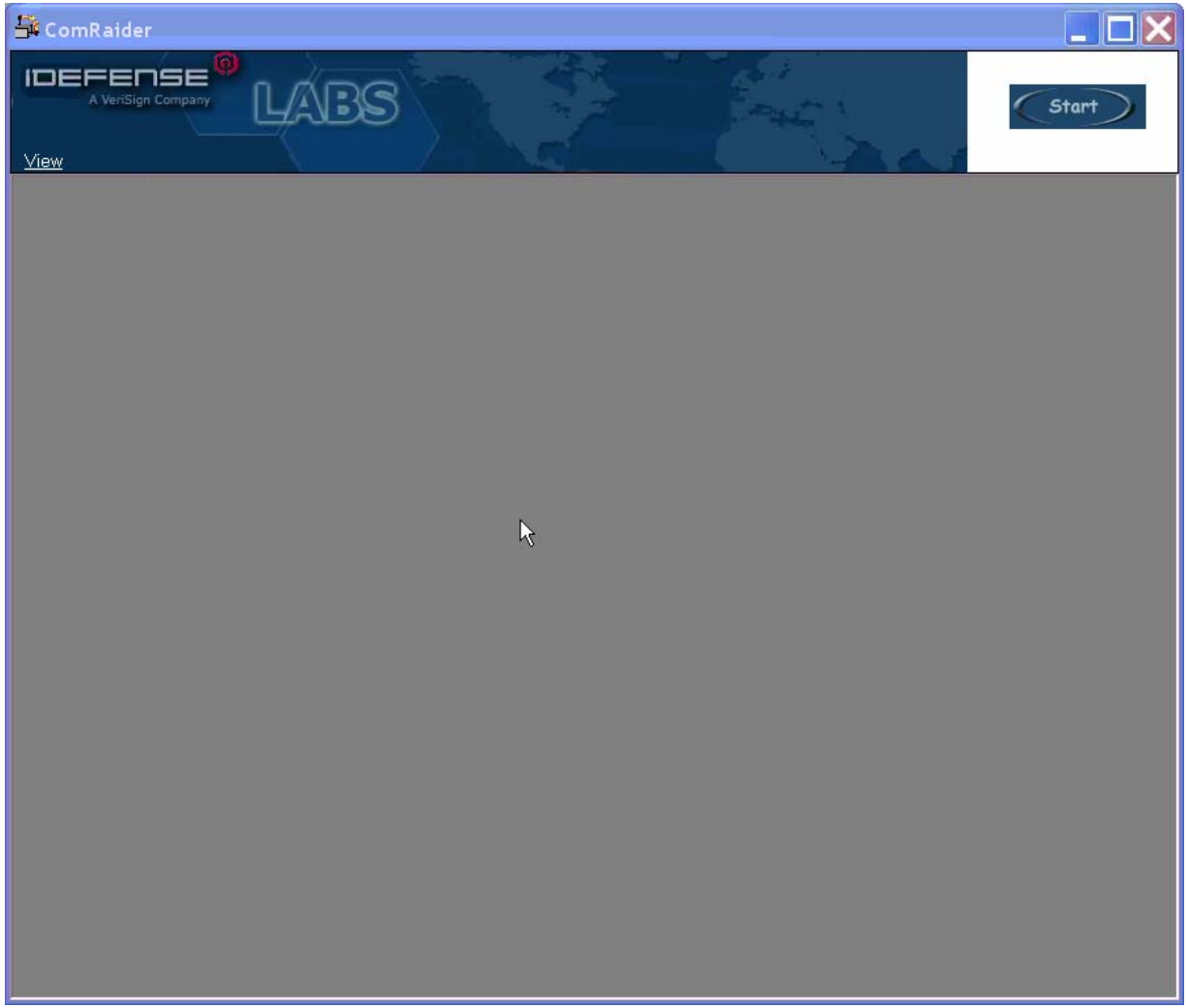


# COMRaider – Determine Exploitability

+ 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