

Using Next Generation Fuzzing Tools

AND THUS SITTING ON A PILE OF ODAY



Dr. Jared DeMott
CHIEF HACKING OFFICER
@jareddemott



1

Jared

Founder - VDA Labs

Former NSA

Built Next-Gen Tech

Teacher

- On-site Trainings
 - Secure Coding, etc.
- Pluralsight
 - AppSec, Malware, Fuzzing, Reversing
 - <https://www.pluralsight.com/courses/security-hackers-developers>
- Cons and University
 - Black Hat, DerbyCon, HITB, etc.

Often in the News

- <https://www.vdalabs.com/2017/12/15/thoughts-on-snowden/>



2

End-to-end Security for Makers

- Training
- Consulting
- Testing
- Response
- Special Projects

End-to-end Security for Business

- Training
- GRC
- Delivery
- Testing
- 24/7 Managed Support
- Incidents / Resilience

3

Why Fuzzing?

Why not DevSecOps?
Or CloudSec?
Or API, Web, or Mobile code security topics?

4

Product Security Engineering and Testing

- SDL assessment
 - Developer Training
 - Architecture and Design Review
- Domain ramp-up
- Code Creation and Testing
 - Component
 - Static
 - Dynamic
 - Manual
- Triage, Remediation, and re-engineering as needed
 - Repeat all steps
- Reporting

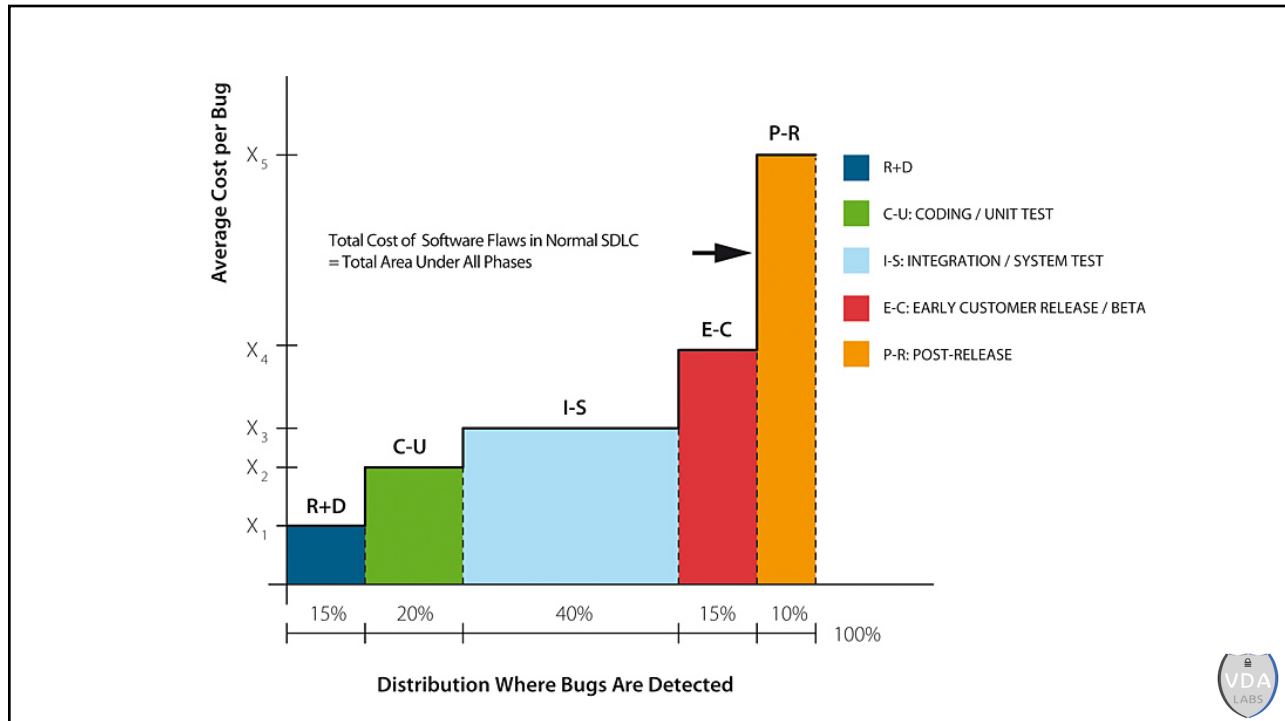
5

Bugs Show up for Lots of Reasons

```

    graph TD
      BF[Business Functions] --> G[Governance]
      BF --> C[Construction]
      BF --> V[Verification]
      BF --> O[Operations]
      
      G --> SM[STRATEGY & METRICS]
      G --> EG[EDUCATION & GUIDANCE]
      G --> PC[POLICY & COMPLIANCE]
      
      C --> SR[SECURITY REQUIREMENTS]
      C --> SA[SECURE ARCHITECTURE]
      C --> TA[THREAT ASSESSMENT]
      
      V --> DR[DESIGN REVIEW]
      V --> SRV[SECURITY TESTING]
      V --> IR[IMPLEMENTATION REVIEW]
      
      O --> EH[ENVIRONMENT HARDENING]
      O --> OE[OPERATIONAL ENABLEMENT]
      O --> IM[ISSUE MANAGEMENT]
      
      SM --- B1[Bug]
      SA --- B2[Bug]
      DR --- B3[Bug]
      OE --- B4[Bug]
    
```

6



7

What Technical Bugs to Look For?

Unmanaged (native) Code

- C and C++

Scripting Languages

- Python, Perl, Go etc.

Managed Code

- .NET (C#), Ruby

Web Application Code


- PHP, Java, ASP, etc.

Mobile Code

- React, native, etc.

- Memory corruption bugs
 - Buffer overflows and the like
- Insecurity design
 - Authentication bypass
 - Cmd injection
 - Weak crypto
 - Native calls to vuln C
- Cloud insecurities
- App flaws
 - OWASP top 10
 - Cmd Injection, IDOR, Permissions, SQLi, XSS, etc.

8




DAST


- Dynamic Application Security Testing
 - Fuzzing native code
 - 70% of patched/CVE Windows bugs are still memory corruption based
 - Application scanning
 - Web, mobile, Cloud

Test the Attack Surface

- APIs
- Pages
- Input fields (network, file data)
- Settings
- Etc.



9




Pros

- Less FP compared to static analysis

Cons

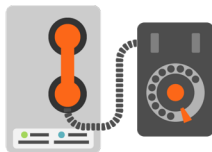
- Only attack surface based
 - In the past
 - Less likely to find deeper bugs
 - Not scanning all the code
 - Harder to integrate into normal dev build processes



10

Traditional Fuzzing Tools vs. Feedback Fuzzers

11



Traditional Fuzzing Methods

Mutation

- Mutations of samples
 - Long run times

Generation

- Define data format, and create mutations based on heuristics
 - Laborious to create

- Suite

- Predefined set of mutations
 - Purchase build your own monitoring



12

Example of Older Fuzzing Tools

- GPF
- Spike
- TAOF
- MiniFuzz
- BFF
- Radamsa
- Sulley
 - BooFuzz
- KernelFuzz
- Peach Fuzzer
- Defensics



Defensics Fuzz Testing

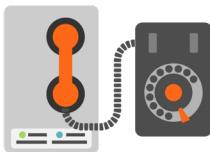
Fuzz Smarter. Remediate Faster. Release Safer.



13

New Fuzzing Techniques

- Scale
- Feedback/Guided
 - Create new test cases based on constraint solving or code coverage/GAs
 - Early research
 - CyberFuzz, FuzzPark, etc.
 - Two tools emerged
 - Sage and AFL
- Even a newer category spoken about at BH - guided + structured



14

The screenshot shows the fuzzbuzz website with the heading "Fuzzing on autopilot" and a sub-heading "FUZZIT". Below this is a terminal window displaying the output of an American Fuzzy Lop (AFL) fuzzer. The terminal output is as follows:

```

american fuzzy lop 1.86b (test)

process timing
  run time      : 0 days, 0 hrs, 0 min, 2 sec
  last new path : none seen yet
  last uniq crash : 0 days, 0 hrs, 0 min, 2 sec
  last uniq hang  : none seen yet

cycle progress
  now processing : 0 (0.00%)
  paths timed out : 0 (0.00%)

stage progress
  now trying : havoc
  stage execs : 1464/5000 (29.28%)
  total execs : 1697
  exec speed  : 626.5/sec

fuzzing strategy yields
  bit flips : 0/16, 1/15, 0/13
  byte flips : 0/2, 0/1, 0/0
  arithmetics : 0/112, 0/25, 0/0
  known ints : 0/10, 0/28, 0/0
  dictionary : 0/0, 0/0, 0/0
  havoc      : 0/0, 0/0
  trim       : n/a, 0.00%

map coverage
  count coverage : 1.00 bits/tuple
  findings in depth
  favored paths : 1 (100.00%)
  new edges on : 1 (100.00%)
  total crashes : 39 (1 unique)
  total hangs   : 0 (0 unique)

overall results
  cycles done : 0
  total paths : 1
  uniq crashes : 1
  uniq hangs  : 0

path geometry
  levels : 1
  pending : 1
  pend fav : 1
  own finds : 0
  imported : n/a
  variable : 0

[cpu: 92%]
    
```

Below the terminal window is a flowchart titled "Overview" showing the workflow: Upload builds → Build bucket Cloud Storage → Fuzzing → Find crash → File bug → Test if fixed (daily) → Close bug. The flowchart also includes steps for "De-duplicate", "Minimize", and "Calculate".

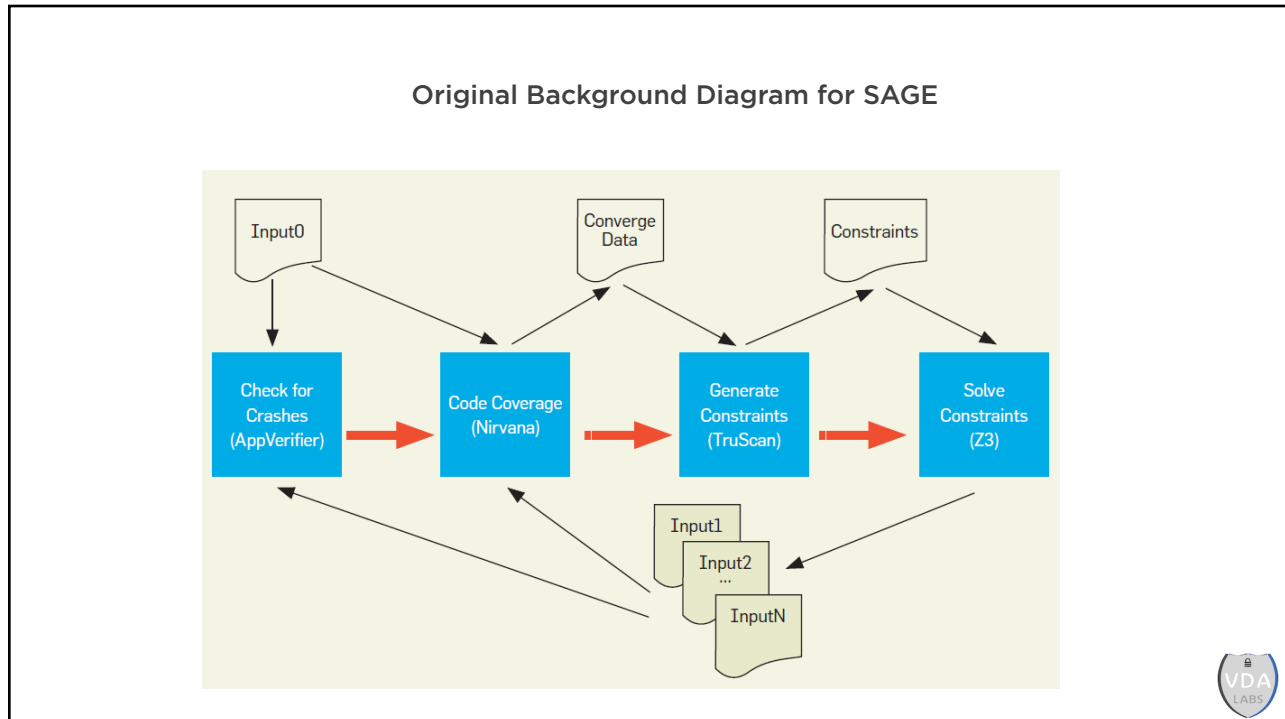
15

```

void f(int x, int y) {
    int z = 2 * y;
    if (x == 100000) {
        if (x < z) {
            assert(0); // bug!
        }
    }
}
    
```

Randomly reaching this code would be unlikely
A fuzzing sample that was close would be required
x = 100000, y = 50001

16



17

SAGE

Operates on Native Binaries

Found Past “Hard Bugs”

- ANI-format bug that blackbox and whitebox missed
- MS07-017
- Found 1/3 of all File Fuzzing Bugs during the security testing of Windows 7
 - SAGE is typically run last, meaning those bugs were missed by other approaches

MSRD has matured A LOT since original SAGE

18

Possible Limitations of Guided Fuzzers

Path Explosion

If Program Exhibits Nondeterministic Behavior

- This can lead to non-termination of the search and poor coverage

Even if Deterministic

- Imprecise symbolic representations
 - Pointers, floating points numbers, etc.

Programs that Thoroughly Mix the State of Variables

- "if (md5_hash(input) == 0xdeadbeef)"

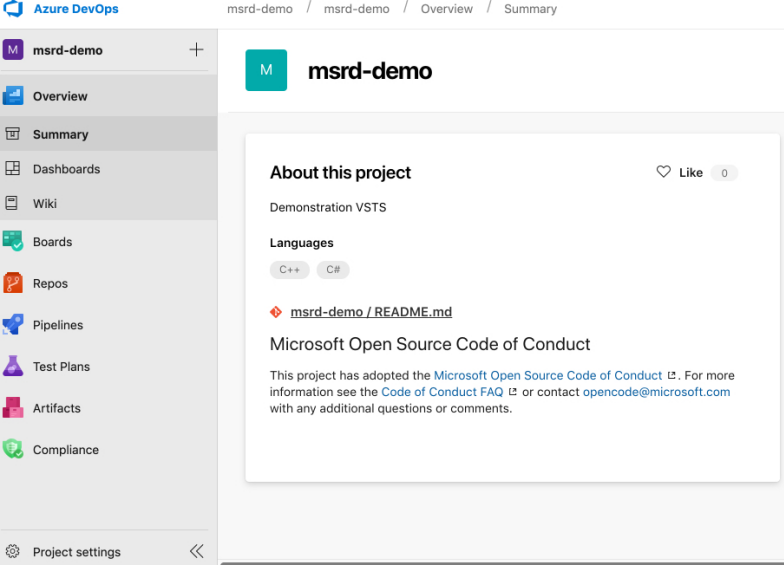


19

But Really, Fuzzing Not
Getting Integrated Into The
CI/CD was/is the Biggest
Limitation

20

Azure DevOps with MSRD Extension



The screenshot displays the Azure DevOps interface for a project named 'msrd-demo'. The left sidebar contains navigation options: Overview, Summary, Dashboards, Wiki, Boards, Repos, Pipelines, Test Plans, Artifacts, Compliance, and Project settings. The main content area shows the 'About this project' section with the following details:

- About this project:** Demonstration VSTS. Includes a 'Like' button with a count of 0.
- Languages:** C++, C#
- msrd-demo / README.md**
- Microsoft Open Source Code of Conduct**
- Text:** This project has adopted the [Microsoft Open Source Code of Conduct](#). For more information see the [Code of Conduct FAQ](#) or contact opencode@microsoft.com with any additional questions or comments.

21

DevOps friendly
This is new for the
fuzzing community

21



Microsoft Security Risk Detection

22

VDA Uses MSRD

<https://microsoftsecurityriskdetection.com>

- Scalable and Cloud Based
- Easy setup of targeted application
 - Includes little tweaks in the VM that make bug hunter more productive
- Nice Reporting and Debugging
 - Triaging is simplified
- Finds security bugs in heavily tested applications
 - Oday machine
 - For certain types of apps anyway



23

Web Scan Results - VDA Labs - WebGoat

Expand All | Site Structure View | Issue View

SQLInjection.aspx

- ASPNET ViewState security
- Blind SQL
- Collecting Sensitive Personal Infor...
- Cross-Site Request Forgery (CSRF)
- HTTPS Everywhere
- JavaScript Memory Leaks
- Reflection
- Reflection
- SQL Injection
- Sensitive Data Exposure
- Sensitive data over an insecure c...
- X-Content-Type-Options
- X-Frame-Options
- X-Powered-By

Attack Module	Error Id
Blind SQL	WS-0011

URL	Severity
http://msrd-webgoat.westus2.cloudapp.azure.com/Content/SQLInjection.aspx	High

Parameter	Attack value
Logical Or with single quote and hash comment	

Summary
Logical Or with single quote and hash comment

Detail
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTA

More Information

Attack 1 | Attack 2 | Attack 3 | Attack 4 | Attack 5 | Attack 6 | Attack 7 | Attack 8 | Attack 9

Request
POST /Content/SQLInjection.aspx HTTP/1.1
Accept: text/html,application/xhtml+xml,application/x
Accept-Encoding: gzip, deflate
Accept-Language: en-US
User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/
Host: msrd-webgoat.westus2.cloudapp.azure.com
Content-Length: 500

Web Scanning In the Same Tool

Maybe Add A Mobile scanner In the Future?

24


Microsoft

Security Risk Detection Fuzzing Jobs Web Scanning Learn More

Fuzzing Jobs - VDA Labs

Id	Name	OS Image	Created	Status	Results
3f78b9aa	XNview KDC	Windows Server 2008 R2 x64	7/24/19 10:02 PM	Fuzzing (Day 6 of 14) Started on: 7/25/19 9:51 AM	10
87f1ec7	XNview FPX	Windows Server 2008 R2 x64	7/24/19 9:54 PM	Fuzzing (Day 6 of 14) Started on: 7/25/19 9:27 AM	103
7e4c2ab2	WZ ISO (Clone)	Windows Server 2008 R2 x64	7/24/19 3:07 PM	Fuzzing (Day 6 of 14) Started on: 7/25/19 11:02 AM	6
d57ce293	WZ RAR	Windows Server 2008 R2 x64	7/20/19 3:31 PM	Fuzzing (Day 10 of 14) Started on: 7/21/19 12:43 PM	5
d266297b	WZ ISO	Windows Server 2008 R2 x64	7/20/19 3:31 PM	Fuzzing (Day 10 of 14) Started on: 7/21/19 1:06 PM	9
cc05452a	XNview JNG	Windows Server 2008 R2 x64	7/16/19 9:44 AM	Fuzzing (Day 6 of 14) Started on: 7/24/19 10:19 PM	1
ecfd7009	XNview PAX	Windows Server 2008 R2 x64	7/14/19 12:35 AM	Completed	24

Native



25



Linux Fuzzing Results

Job Results - VDA Labs - isoinfo ISO Fuz... 4 results

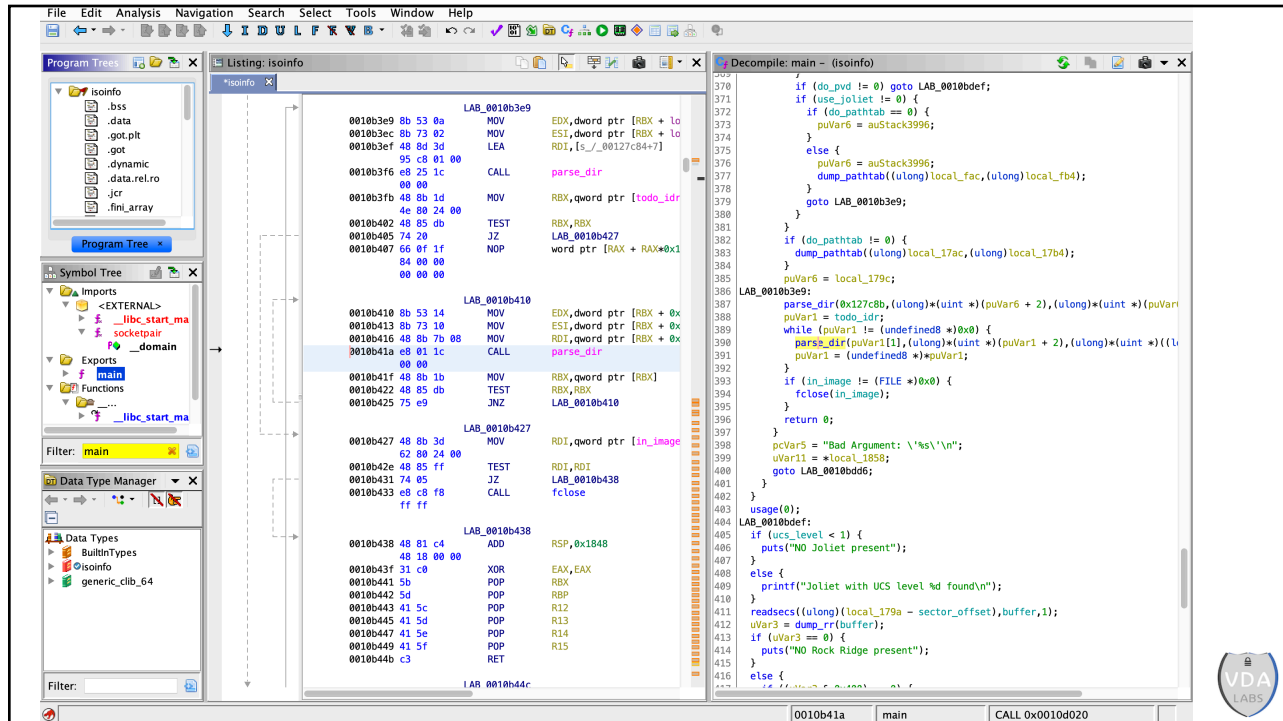
Filters [Clear filters](#) [Export selected](#) [Download seed files](#) [Download selected](#)

Severity	Created	Severity	Type	Hash
Crashes				
Type				
AbortSignal	9/5/19 3:51:59 PM	Crashes	AbortSignal	7235277...
StackBufferOverflow	9/5/19 3:47:52 PM	Crashes	StackBufferOverflow	3381760...
StackBufferOverflow	9/5/19 6:15:03 PM	Crashes	StackBufferOverflow	4582305...
Unknown	9/5/19 3:44:55 PM	Crashes	Unknown	255

Exploitable!



26



27

How Does MSRD Work?

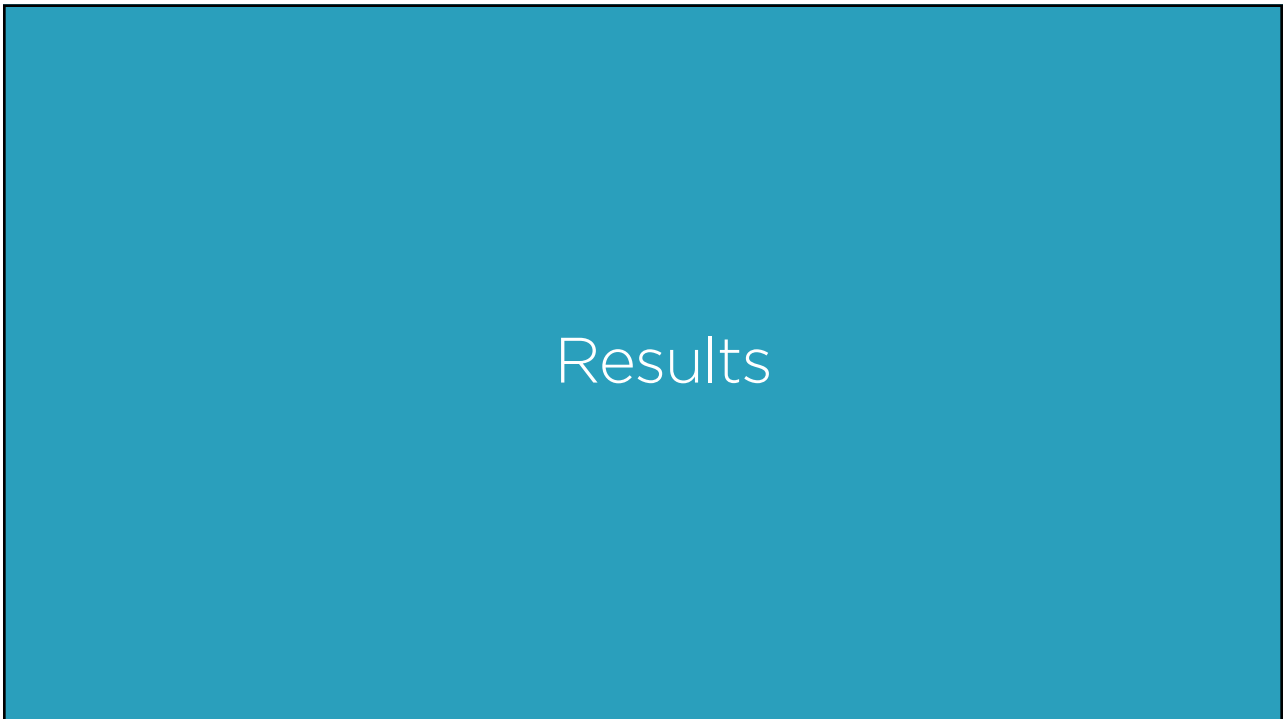
- DevOps or Manually Upload your binary
 - Fuzzing setup/analysis help is available through VDA Labs
 - We are fuzzing experts
- Run Multiple fuzzing jobs as needed
- Fix the security bugs and repeat

DEMO

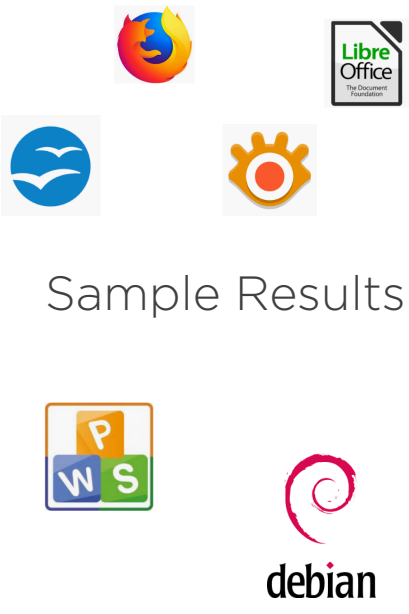
28



29









30




Sample Results

Sample of Applications Tested

- Foxit PDF Reader
- WinZip
- WinRAR
- Chrome
- Firefox
- OpenOffice
- LibreOffice
- Office Plugin
- WPS
- IrfanView
- XnView
- Isoinfo
- Lots more...









31



Types of Crashes MSRD Finds

- Write Access Violations
- Read Access Violations
- Null Dereference
- Heap/Stack Corruptions
- Float Divides/Divide By Zero
- Exception Handler Misuses
- Kernel-Base C++ Exceptions ...



32

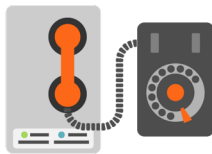
Example of Results

Foxit PDF Reader Update 95

Exploitable = Yes

+	<input type="checkbox"/>	6/7/19 2:13:59 AM	FirstChanceExceptions	Read Access Violation ▲	7249152...
+	<input type="checkbox"/>	6/8/19 7:47:55 AM	FirstChanceExceptions	Read Access Violation ▲	1199529...
+	<input type="checkbox"/>	6/7/19 2:42:12 AM	FirstChanceExceptions	Read Access Violation ▲	1365144...
+	<input type="checkbox"/>	6/7/19 10:42:07 AM	FirstChanceExceptions	Read Access Violation ▲	1522698...
+	<input type="checkbox"/>	6/7/19 10:03:50 AM	FirstChanceExceptions	Read Access Violation ▲	1745974...
+	<input type="checkbox"/>	6/10/19 5:35:15 AM	FirstChanceExceptions	Read Access Violation ▲	1911130...

33



Linux Fuzzing Supported

- RedHat Linux Virtual Machine
- Quick and Easy Setup!
 - SSH based connection to VM
- Helps support Multi-Cross platform applications
 - Lots of support for tons of applications
- Lots of bugs to find
 - Quick Results (in 25 Minutes!)



34

Example Crash

- Easier Triaging

- Bucket/Review crashes in cloud interface
- Seeds that have caused crashes are saved for future review
- Easily download seeds that caused crash and review
- Still manual to create exploits

35



!Exploitable



Some Stats

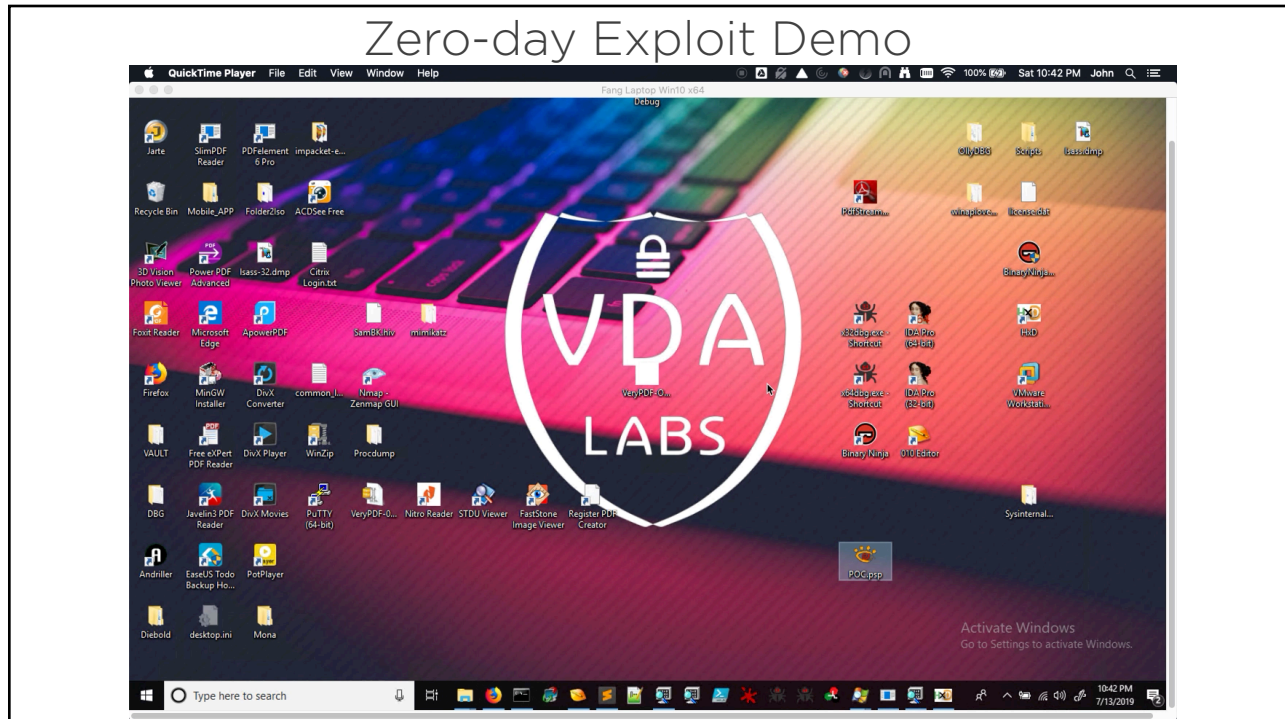
- 15-20% Exploitable
- 20-25% Probably Exploitable
- 10-15% Probably Not Exploitable
- 55-60% Unknown

Tested Over 50 applications


- 1000+ solid bugs
 - Lots of Oday - at least 100



36




37

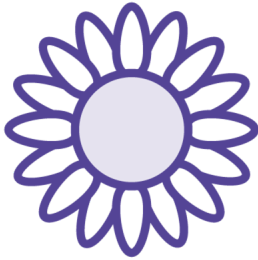


Professional Services

- Any complex domain, tool, and development practice will always require expert setup
 - Seed Discovery/Creation
 - Feedback tools generally work only on file consuming applications
 - Custom Harness creation for APIs, network applications, etc.
 - Software Fix or Exploit Dev
- VDA Labs is the MSRD partner



38



Seeds

- Good seeds to get the best code coverage
 - Must be in a file format type
 - MinSet is common
 - Though, a harness can be used with seeds for complex application testing
- Good seeds find more crashes!



39



Fuzzing Harnesses

- Take file inputs and use them in creative ways to test applications
- Can be used to test hard to reach code paths
 - Network, API, etc
 - Some setup time is required
 - Known to find 15 year old bugs in one case



40

Custom Fuzzing Harnesses

- Client/Server Setups
- GUI Applications
- API Testing
- Wrappers

```

18 {
19     // get length of file
20     inFile.seekg(0, inFile.end);
21     int length = inFile.tellg();
22     inFile.seekg(0, inFile.beg);
23
24
25     // read file into heap buffer
26     char *buffer = (char*) malloc(length);
27     inFile.read(buffer, length);
28
29     // make sure heap buffer was populated before sending to target library
30     if (buffer)
31     {
32         // call library function and pass input from heap/file buffer
33         VulnDLL::VulnDLLFunctions vulnfunks;
34         vulnfunks.VulnFunction(buffer);
35     }
36     printf("end\n");
37     // delete our heap buffer to prevent memory leaks
38     free(buffer);
39 }
40

```

41



Exploit Development

- PoCs can help expedite the fix
- Building Exploits with Crashes
- Proving the importance of the crash
- Providing Vendors the Crash along with the Exploit



42

Exploit Development

The screenshot displays the IDA Pro interface. The disassembly window shows a list of memory addresses from 41414141 to 41414152, with the value '???' for each. A red arrow points to the address 41414141. The register window shows the value of the 'eax' register as 41414141, also highlighted with a red arrow. The command window shows an exception report for an access violation at address 004010246, with the following details:

```

(3520.3700): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=00000000 ebx=00000000 ecx=41414141 edx=775ef1f0 esi=00000000 edi=00000000
eip=41414141 esp=00afd6f0 ebp=00afd710 iopl=0         nv up ei pl zr na pe nc
cs=0023  eax=002h  ds=002h  eax=002b  fs=0053  gs=002b             efl=00010246
41414141 ??             ???

```

43



Code Patches

- Typically work with vendors to have them patch their code directly
- But, could also
 - Building patches in IDA or Ghidra
 - Fixing Vulnerable Code



44

Identifying The Crash

Fixing Vulnerable Code

Applying Patches to Test Applications

45

VDA Gives Back! :-)

- Exploits and Crashes
 - Windows/Linux
- Will release some of our work once Vendors have confirmed patches
 - Will update as we find more and report
- <https://github.com/VDA-Labs>

46

Questions?



Secure Coding is Hard

- Your networks and products are at risk
- Takes passion, diligence, and commitment from all players to be safe
 - Best practices in
 - *Training, design, coding, testing, deployment, monitoring, and response*

Thanks!

- @vdalabs
- www.vdalabs.com
- info@vdalabs.com

