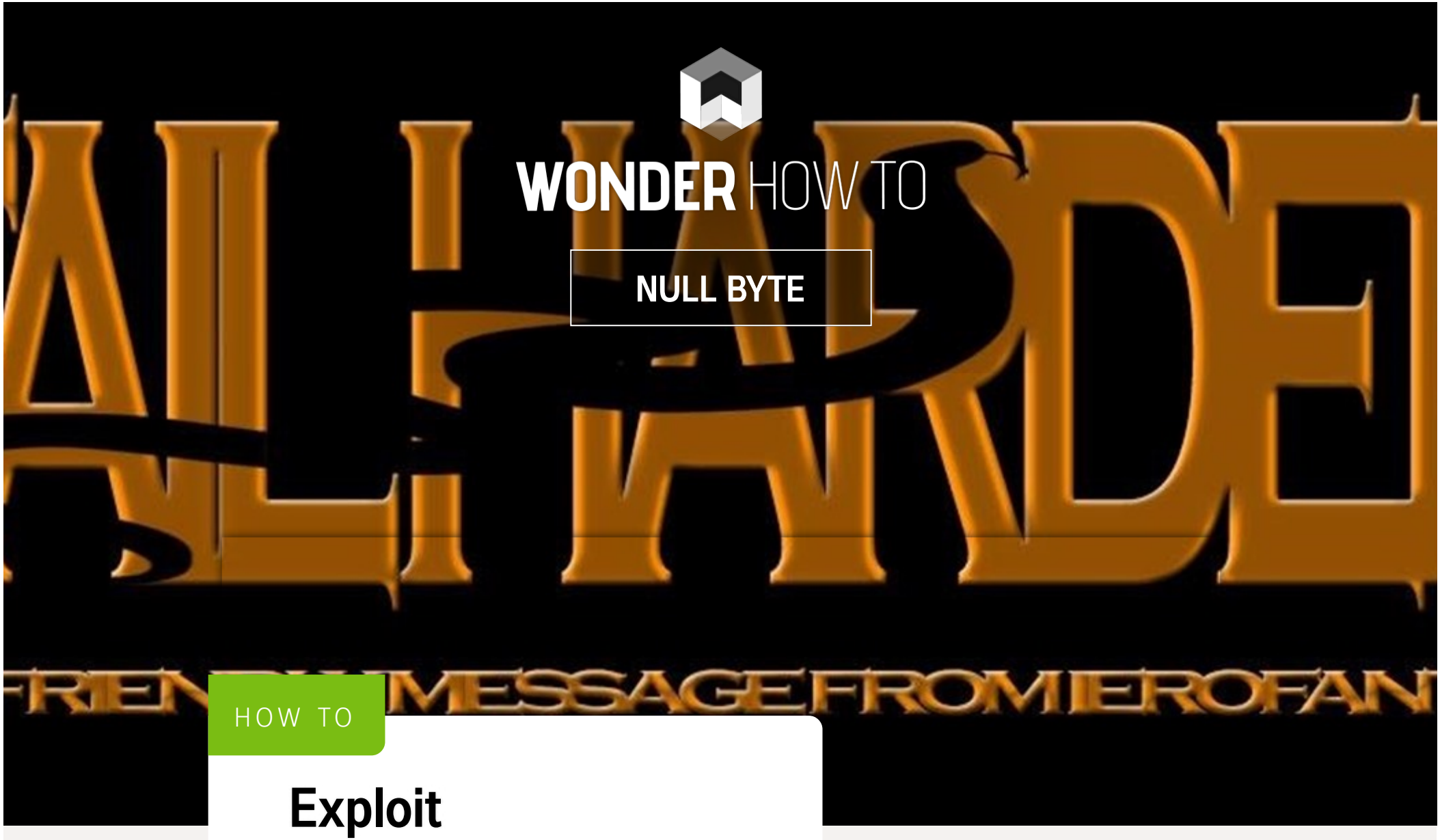


Everything You Need to Know

BY **IEROFANTIS MAX** 01/14/2016 4:36 PM

Step 1 What Exploit Development Is and Why Should I Be Interested on About This Topic

An exploit is a piece of software, a chunk of data, or a sequence of commands that takes advantage of a bug or vulnerability in order to cause unintended or unanticipated behavior to occur on computer software, hardware, or something electronic (usually computerized). Such behavior frequently includes things like gaining control of a computer system, allowing privilege escalation, or a denial-of-service attack.

Reasons on why should I learn about Exploit Development

1) Personal: The feeling of making an exploit or finding a vulnerability is incomparable. ?hat is the moment when you begin to feel really someone who is thinking outside of the box and begins to create his own "art" (what hacking really is).

2) Objectives: In many cases there is a need to build your own exploit and if you have this knowledge you are a step above ahead from the plethora of "professionals" with the many meaningless certifications.

HOW TO

4 Ways to Crack a Facebook Password & How to Protect Yourself from Them

HOW TO

Top 10 Things to Do After Installing Kali Linux

3) Financial: Most serious (pay attention to the serious) vendors **NOWADAYS** will pay you if you email them and tell them that you found a vulnerability in their program or their systems.

Step 2 How to Start

The holy grail of tutorials goes like :

The only reliable sources are those of Corelan Team (https://www.corelan.be/) and fuzzy security (http://www.fuzzysecurity.com/) .Also other websites or blogs are ALWAYS mentioning these both sources as references.

I will begin from analyzing the problem of learning from Corelan Team and fuzzy security and later I will explain the main mistake of the rest of the blogs or websites when they talk about this subject.

There Is no way to understand these articles If you are someone who is beginning this great journey .Even if you partly understand what It is going on it is impossible to fully realize what these articles are saying. Also these articles are outdated and I will explain thoroughly later.

HOW TO

Buy the Best Wireless Network Adapter for Wi-Fi Hacking in 2019

HOW TO

Hack Android Using Kali (Remotely)

HOW TO HACK WI-FI

Cracking WPA2 Passwords Using the New PMKID Hashcat Attack

The same applies to fuzzy security, although it is simplier and more noob friendly .However simplicity does not always mean that the researcher understands what It really happens.

This is happening for many reasons

Someone new to the field of exploit development will believe that If he wants to make an exploit he should load a program with random bits to occur buffer overflow and after that he is ready to load the payload.

HOW TO

Crack Any Master Combination Lock in 8 Tries or Less Using This Calculator

HOW TO HACK WI-FI

Cracking WPA2-PSK Passwords Using Aircrack-Ng

HOW TO

Get Unlimited Free Trials Using a "Real" Fake Credit Card Number

the whole of your time on the exploit-db reading exploits and understand what is happening, because exploit-db is not the most reliable source, nor all the exploits there are working (I have tried thousands)

Also you will not learn about white / black / gray box fuzzing, reverse engineering, different kind of bugs etc. things that I will analyze below.

The other blogs and websites unfortunately they follow the same pattern.

What should I learn

It is great to judge others but we should suggest solutions. My solution is in the following lines to guide anyone who wants to start his journey to exploit development from the beginning to the end trying to be as clear as possible.

Step 3 A-Operating Systems / Memory Management (Windows)

Why should you know about the operating systems and especially Windows?

Because the overwhelming percentage of the users and programs aimed for Windows, so the exploits should

be made for the Windows and not for XP but the most recent ... We are not living in 2001, but in 2016

Also you have to deal with operating systems because they contain very important concepts for exploit development:

1) scheduler (time allocation to programs),
2) memory manager (space allocation to programs)
3) synchronizer (avoid conflicts for a resource at the same time)
4) protection (avoid conflicts for memory)

As you see I have underlined the memory manager because Memory is a finite resource required by most non trivial processes. It is the operating systems obligation to ensure that memory is correctly managed in the sense that processes that request memory are given it and that processes memory is protected from other processes. The specific roles of the operating system include tracking the amount of free and allocated memory and also handling the transfers between memory and disk.

I have also emphasized on number four because it's necessary when you want to break something apart to understand how it works (memory manager) and how It is protected.

Sources for learning about these concepts:

http://www.osdever.net/tutorials/index
http://www.tutorialspoint.com/operating_system/index.htm
http://elsoc.wikia.com/wiki/Memory_Management
https://web.stanford.edu/class/cs140/projects/pintos/pintos.html
https://learn.saylor.org/course/cs401
http://pages.cs.wisc.edu/~remzi/OSTEP/

And of course you have to download the book:

Modern Operating Systems- Andrew S.Tanenbaum

Step 4 B-Programming Language

You have to know even the basics of programming, no matter what language you will start to learn because all work with the same way. All languages have statements, operators, functions, exceptions etc and what changes is the syntax of the language.

Now that you know the basics of operating systems and how memory functions and pointers work I reccomend you to learn C, because C has to do with:

loops, pointers/recursion, Data structures (linked lists, binary trees etc.) concepts very important to exploit Development.

Places to learn C:

Books that you can find on the internet

- The C book
- Essential C
- The new C standard - an annotated reference
- Object Oriented Programming in C (PDF)
- Programming in C (3rd Edition) - Stephen Kochan
- C Primer Plus - Stephen Prata
- C Programming: A Modern Approach - K. N. King
- A Book on C - Al Kelley / Ira Pohl
- The C book - Mike Banahan, Declan Brady and Mark Doran
- Practical C Programming, 3rd Edition - Steve Oualline
- C: How to Program (6th Edition) - Paul Deitel & Harvey M. Deitel
- Head First C - David & Dawn Griffiths

Websites

www.cprogramming.com
www.zentut.com/c-tutorial
www.geeksforgeeks.org
www.thenewboston.org (videos)
www.wikipedia.com
www.stackoverflow.com

After it'll be easy to learn and read Python or Perl, the majority of exploits are written on these languages

Step 5 C- Learn Assembly

Now that you have learned about how a program operates within a system (A, B) it's time to learn Assembly.

Why should I learn Assembly

Assembly language is one of the closest forms of communication between humans and computers.

With assembly, the programmer can precisely track the flow of data and execution in a program in a mostly human-readable form. Once a program has been compiled, it is difficult (and at times, nearly impossible) to reverse-engineer the code into its

original form. As a result, if you wish to examine a program that is already compiled but would rather not stare at hexadecimal or binary, you will need to examine it in assembly language.

Since debuggers will frequently only show program code in assembly language, this provides one of many benefits for learning the language.

NOT OPTIONAL! Without the knowledge of Assembly you will not be able to analyze correctly a program that is already compiled. Basically you will not be able to understand anything from the processes the debugger performs.

What Assembly language should I learn

You have to learn how to program in x86 assembly, as well as the history and basic architecture of x86 processor family.

When referring to x86 we address the complete range of x86-based processors (since the original Intel 8086 in 1978). This includes:

• IA-32 assembly, also commonly referred to as x86-32 assembly (Intel architecture 32-bit, since the Intel

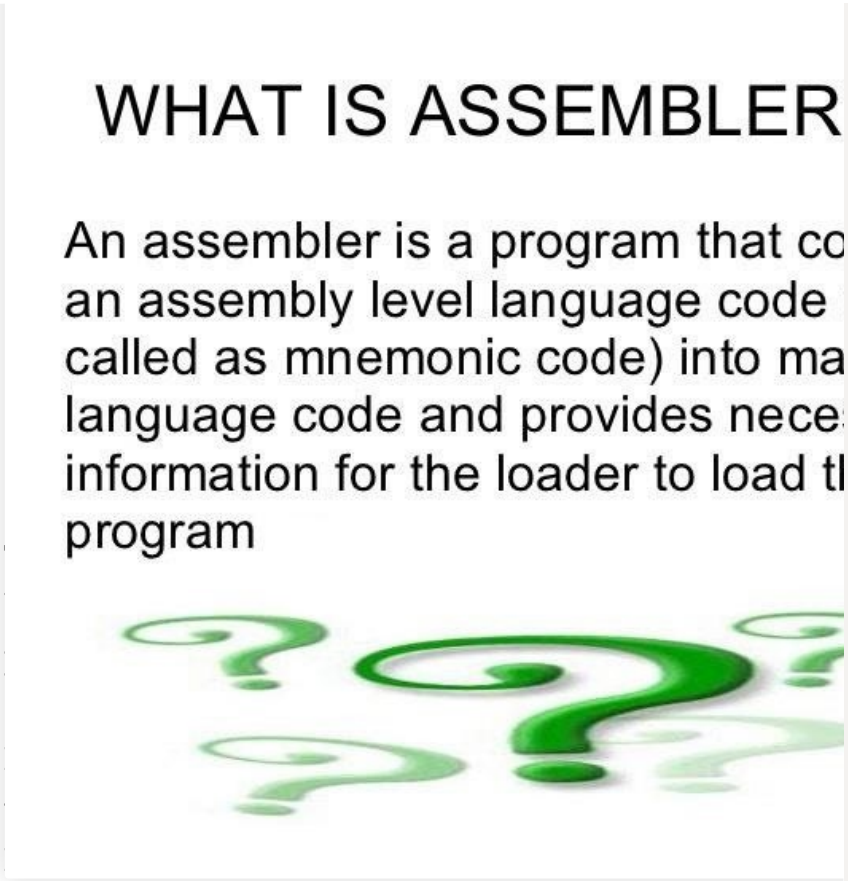
80386), a 32-bit extension of the original 16-bit Intel x86 processor architecture (used in Intel 8086 - 80286 CPUs). IA-32 has full backwards compatibility with the 16-bit x86.

• x86-64, also called the AMD64 or AMD 64-bit extension, backwards compatible with 32-bit code without performance loss.

• Intel 64, previously named IA-32e or EM64T, almost identical to x86-64.

The term "x86" can refer both to an instruction set architecture and to microprocessors which implement it. The name x86 is derived from the fact that many of Intel's early processors had names ending in "86".

What Assembler should I use



My personal preference is NASM and I would advise you to use NASM too.

Source to learn Assembly:

https://courses.engr.illinois.edu/ece390/books/artofasm/artofasm.html

Step 6 D-Reverse Engineering

Before we go to the last part of fuzzing now that you have the necessary supplies you should collect/learn the tools to do reverse engineering.

Why Reverse Engineering is nessecary?

a) Finding malicious code. Many virus and malware detection techniques use reverse engineering to understand how abhorrent code is structured and functions.

b) Discovering unexpected flaws and faults. Even the most well-designed system can have holes.

c) Finding the use of others' code. Reverse engineering techniques
can be used to detect the presence or absence of software elements
of concern.

d) Finding the use of shareware and open source code where it was not
intended to be used. Reverse engineering enables the detection of code replication issues.

What tools I will need

You will need System Monitoring Tools, Disassemblers, Debuggers and Decompilers.

System monitoring is an important part of the reversing process. In some cases you can actually get your questions answered using system-monitoring tools and without ever actually looking at code.

Here is a brief overview of their most interesting tools:

- FileMon This tool monitors all file-system-level traffic between programs and the operating system
- Nagios - Network Monitoring Software
- Cacti - Network Monitoring Software
- TCPView This tool monitors all active TCP and UDP network connections on every process.
- PortMon PortMon is a physical port monitor that monitors all serial

and parallel I / O traffic on the system

- Process Explorer Process Explorer is like a turbo-charged version of the built-in Windows Task Manager, and was actually designed to replace it.Process Explorer can show processes, DLLs loaded within their address spaces, handles to objects within each process, detailed information on open network connections, CPU and memory usage graphs, and the list just goes on and on.

Disassemblers

In essence, a disassembler is the exact opposite of an assembler. Where an assembler converts code written in an assembly language into binary machine code, a disassembler reverses the process and attempts to recreate the assembly code from the binary machine code.

Online Disassembler:
https://www.onlinedisassembler.com/odaweb/

Commercial Freeware / Shareware Windows Disassemblers

OllyDbg
http://www.ollydbg.de/ (official website)
http://www.openrce.org/downloads/browse/OllyDbg_Plugins (plugins)
http://www.ollydbg.de/odbg64.html (64 bit version)

Decompilers

A decompiler is a computer program that takes as input an executable file, and attempts to create a high level, compilable source file that does the same thing. It is therefore the opposite of a compiler, which takes a source file and makes an executable. Decompilers usually do not perfectly reconstruct the original source code, and can vary widely in the intelligibility of their outputs.

C4Decompiler: A new decompiler under development. Windows only, has a slick user interface inspired to Visual Studio 2010 with many useful interactions, that unfortunately are not always obvious.

Dcc: DOS to C decompiler. One of the first decompilers. It shows its age, but it's still referenced by many other decompilers for its structuring abilities. Only supports 8086 (16 bits) programs.

- Jad (commercial, no fee for noncommercial use, no source code)
- JADO (Free, GPL, not actively maintained)
- DJ Java Decompiler ("freeware", no source code)
- HomeBrew Decompiler (Free, GPL)
- JODE decompiler and optimizer (Free, GPL)
- JReversePro (Free, GPL)
- SourceTec Java Decompiler (commercial, patch to Mocha)

A must-read book about reverse engineering is: Reversing-Secrets of Reverse Enginering, Eldad Eilam

Step 7 E-Software Auditing / Fuzzing

Software Auditing

There are many ways to find a vulnerability, but in very general terms we can say there are three main ways: white / black / gray-box testing.

White-Box Testing: It refers to the testing of a system with full knowledge and access to all source code and other architecture documents. This testing enables to reveal bugs and vulnerabilities quickly in comparison with trial and error method. More complete testing coverage is ensured by exactly knowing what to test

Some Freeware programs for software auditing and white-box testing are:

• Google CodeSearchDiggity - Utilizes Google Code Search to identifies vulnerabilities in open source code projects hosted by Google Code, MS CodePlex, SourceForge, Github, and more. The tool comes with over 130 default searches that identify SQL injection, cross-site scripting (XSS), insecure remote and local file includes, hard-coded passwords, and much more. Essentially, Google CodeSearchDiggity provides a source code security analysis of nearly every single open source code project in existence - simultaneously.

• FindBugs - Find Bugs (including some security flaws) in Java Programs

• FxCop (Microsoft) - FxCop is an application that analyzes managed code assemblies (code that targets the .NET Framework common language runtime) and reports information about the assemblies, such as possible design, localization, performance, and security improvements.

• PMD - PMD scans Java source code and looks for potential code problems (this is a code quality tool that does not focus on security issues)

• PreFast (Microsoft) - PREfast is a static analysis tool that identifies defects in C / C ++ programs

• RATS (Fortify) - Scans C, C ++, Perl, PHP and Python source code for security problems like buffer overflows and TOCTOU (Time Of Check, Time Of Use) race conditions

• OWASP SWAAT Project - Simplistic Beta Tool - Languages: Java, JSP, ASP .Net, and PHP

• Flawfinder Flawfinder - Scans C and C ++

• RIPS - RIPS is a static source code analyzer for vulnerabilities in PHP web applications

• Brakeman - Brakeman is an open source vulnerability scanner specifically designed for Ruby on Rails applications

• Codesake Dawn - Codesake Dawn is an open source security source code analyzer designed for Sinatra, Padrino and Ruby on Rails applications. It can work also for non web application wrote in Ruby programming language

• VCG - Scans C / C ++, Java, C # and PL / SQL for security issues and for comments which may indicate defective code. The config files can be used to carry out additional checks for banned functions or functions which commonly cause security issues.

Black Box : Testing refers to testing a system without knowledge of specification to the internal workings of the system, access to the source code, and knowledge of the architecture. Essentially this approach mimics in a close approach, how an attacker typically follows approach to the application. However, the uncovering of issues or vulnerabilities could be further longer, because of lacking internal application knowledge.

Black box testing implies that you only have knowledge of what you can observe. You as the end user control the input that goes into the black box and you can observe the ouput that emerges from the other end, but you do not have knowledge of the inner workings of the target. The situation is most commonly found when remotely accessing Web Applications and Web Services.You can craft inputs in the form of Hypertext Markup Language (HTML) or Extensible Markup Language (XML) requests and observe the generated Web page or return value,

respectively, but have no idea what is going on under the hood.

Gray-box testing: Is the combination of black-box and white-box testing

What is Fuzzing

Fuzz testing or fuzzing is a software testing technique, often automated or semi-automated, that involves providing invalid, unexpected, or random data to the inputs of acomputer program. The program is then monitored for exceptions such as crashes, or failing built-in code assertions or for finding potential memory leaks. Fuzzing is commonly used to test for security problems in software or computer systems. It is a form of random testing which has been used for testing hardware or software.

The field of fuzzing originated with Barton Miller at the University of Wisconsin in 1988

Smart and dumb fuzzing

A fuzzer that generates completely random input is known as a "dumb" fuzzer, as it has no built-in intelligence about the program it is fuzzing. A dumb

fuzzer requires the smallest amount of work to produce (it could be as simplistic as piping / dev / random into a program). This small amount of work can produce results for very little cost - one of fuzzing's big advantages.

However, sometimes a program will only perform certain processing if particular aspects of the input are present. For example, a program may accept a "name" field in its input, and this field may have a "name length" associated with it. If these fields are not present in a form that is valid enough for the program to identify, it may never attempt to read the name. However, if these fields are present in a valid form, but the length value is set to the incorrect value, the program may read beyond the buffer containing the name and trigger a crash. Without input that is at least partly valid, this is very unlikely to happen. In these cases, "smart" fuzzers can be used.

Fuzzing Phases

1) Identify target. It is not possible to select a fuzzing tool or technique until we have a target in mind.

2) identify input. Virtually all exploitable vulnerabilities are caused by applications accepting

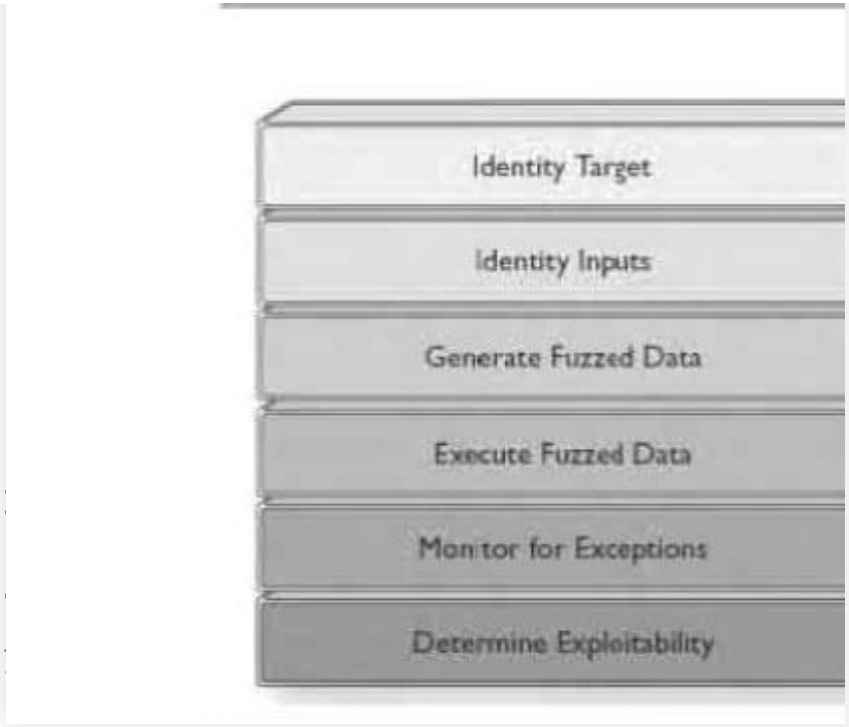
user input and processing that data wirhout first sanitizing it or applying validation routines.

3) Generate fuzzed data. Once input vectors have been identified, fuzz data must be generated.

4) Execute fuzzed data.

5) Monitor for Exceptions.A vital but often overlooked step during fuzzing is the exception or fault monitoring process.Transmitting 10.000 fuzz packets to a target Web Server, for example, and ultimately causing that server to crash is a useless endeavor if we are unable to pinpoint the packet responsible for the crash.

6) Determine exploitability. Once a fault is identified depending on the goals of the audit, t might also be necessary to determine if the uncovered bug can be further exploited.



may still take a while to create a working fuzzer for your target; by contrast, others take a very simple approach. A selection of these frameworks and fuzzers is listed here for your reference:

- Radamsa

Radamsa is designed to be easy to use and flexible. It attempts to "just work" for a variety of input types and contains a number of different fuzzing algorithms for mutation.

• Sulley

Sulley provides a comprehensive generation framework, allowing structured data to be represented for generation based fuzzing. It also contains components to help with recording test cases and detecting crashes.

• Peach

The Peach framework can perform smart fuzzing for file formats and network protocols. It can perform both generation- and mutation-based fuzzing and it contains components to help with modelling and monitoring the target.

• SPIKE

SPIKE is a network protocol fuzzer. It requires good knowledge of C to use and is designed to run on Linux.

• Grinder

Grinder is a web browser fuzzer, which also has features to help in managing large numbers of crashes.

• NodeFuzz

NodeFuzz is a nodejs-based harness for web browsers, which includes instrumentation modules to gain further information from the client side

Learn About Fuzzing:

• Fuzzing: Brute Force Vulnerability Discovery (ISBN 0321446119)
• Fuzzing for Software Security Testing and Quality Assurance (ISBN 1596932147)

• Peer Reviews in Software: A Practical Guide (ISBN 0201734850)

• Secure Programming with Static Analysis (Brian Chess / Jacob West)

• A Bug Hunter's Diary: A Guided Tour Through the Wilds of Software Security

Step 8 Epilogue

I am confident that all of the above clarify what someone has to learn about exploit development, and I am glad that I add an article in the family of exploit

development articles, that I believe will help many people out there to have a great beginning on this learning process.

If anyone wants to see a demonstration of all the above take a look at this article that It has a video of mine where I find a vulnerability and I make an exploit.

https://null-byte.wonderhowto.com/how-to/exploit-development-stack-base-buffer-overflow-part-1-video-0167376/

Thank you and Remember!

References

Books

The *ShellCoder's* Handbook- Discovering *and* Exploiting *Security* Holes, 2nd *Edition*
The.IDA.Pro.Book.2nd.Edition.Jun.2011
The Mac Hacker's Handbook
The Art of Assembly Language, 2nd Edition
Reversing - Secrets Of Reverse Engineering (2005)
Hacking - The Art of Exploitation 2nd Ed
gray *hat* hacking *3rd* edition
Fuzzing Brute Force Vulnerability Discovery
Modern Operating Systems 3e (2007)

Websites / Links

https://www.mwrinfosecurity.com/knowledge-centre/15-minute-guide-to-fuzzing/
https://www.owasp.org/index.php/Source_Code_Analysis_Tools
http://www.careerride.com/Testing-white-box-black-box-gray-box.aspx
http://www.backerstreet.com/decompiler/decompilers.htm
http://www.thefreecountry.com/programming/disassemblers.shtml
https://en.wikibooks.org/wiki/X86_Disassembly/Disassemblers_and_Decompilers

https://en.wikipedia.org/wiki/List_of_debuggers
https://courses.engr.illinois.edu/ece390/books/artofasm/artofasm.html
https://stackoverflow.com/questions/214734/some-x86-asm-reference-tutorials
http://www.gcflearnfree.org/computerbasics/2/print
http://www.tutorialspoint.com/operating_system/
http://elsoc.wikia.com/wiki/Memory_Management
http://academicearth.org/
http://www.osdever.net/tutorials/index
https://learn.saylor.org/course/cs401
http://pages.cs.wisc.edu/~remzi/OSTEP/
http://www.tutorialspoint.com/cprogramming/
http://www.cprogramming.com/tutorial.html
http://einstein.drexel.edu/courses/Comp_Phys/General/C_basics/

- Follow Null Byte on Twitter, Flipboard, and YouTube
- Sign up for Null Byte's weekly newsletter
- Follow WonderHowTo on Facebook, Twitter, Pinterest, and Flipboard

Never Miss a Hacking or Security Guide

New Null Byte in your inbox, every week.

Your Email

GET THE NEWSLETTER

Related

HOW TO

Exploit Development-Stack Base Buffer Overflow/Part 1(VIDEO)

HOW TO

Create a Metasploit Explo Minutes

15 Comments

THOGS TUTS
3 YEARS AGO

☆ 3

Impressive guide! I'm looking forward to hearing more from you in the next time.

REPLY

IEROFANTIS MAX
3 YEARS AGO

☆ 3

Thank you very much! I am working at this moment on reverse engineering,fuzzy techniques etc so I will definitely write similar tutorials.

REPLY

WASHU WASHU
3 YEARS AGO

☆ 4

One of the most complete and comprehensive guide on the topic. Extremely well done! I'l definitely be referring back to this when ever I need to do some exploit development.

Cheers,
Washu

REPLY

IEROFANTIS MAX
3 YEARS AGO

1

Thank you so much for your kind words!

REPLY

FEDERICO MOLTÓ

3 YEARS AGO

1

Thx for sharing your knowledge! I've been learning for some time and null byte has been incredible helpfull to me.

i'll share an idea...i'm a really soon too be lawyer and i'm working on a project that aims to use the same "hacking" mindset to our legal institutions, because i see in laws/sentences something like the "code" that rules our societies, so i think that the best way to optimize them is to find their vulnerabilities and exploit them (in a systemic way). If anyone is interested or could guide me a little please PM me! I truly think this deserves our attention.

Thanks again from Mendoza, Argentina!

REPLY

IEROFANTIS MAX

3 YEARS AGO

1

Welcome and thank you very much! I hope all the best for your project and don't hesitate to ask me about this.
Greetings from Greece

REPLY

GTOROSS
3 YEARS AGO

☆ 3

omg +kudos
i'll read this with more time, thx OP

REPLY

IEROFANTIS MAX
3 YEARS AGO

1

Thank you, glad you like It !

REPLY

MR_NAKUP3NDA
3 YEARS AGO

☆ 3

well done +1
Hacked by Mr_Nakup3nda

REPLY

IEROFANTIS MAX
3 YEARS AGO

1

I appreciate this!
Ierofantis

REPLY

1

UNH0LYS0DA
3 YEARS AGO

Amazing guides, really helps in giving an overview in what to to learn, which is a lot ofcourse, but that's good. KUDOS.

REPLY

IEROFANTIS MAX
3 YEARS AGO

1

Thank you Unholy!

REPLY

ELECTRONICENGINEER52
ELECTRONICENGINEER52
2 YEARS AGO

1

Thank you for your great and gainful article.In step 3 (O.S) can I start with Linux instead of Windows?

REPLY

PROJECT HACKER
1 YEAR AGO

1

Dear Ierofantis , Thanks so much for such a great and comprehensive reference regarding exploit development. I read a lot about this on the internet but never find a well-documented and written tech article like yours.

Best of luck and looking forward to read more from you - I already registered and would recommend your website to other friends interested in this subject.

The most thing I loved in this article is :

1) Personal: The feeling of making an exploit or finding a vulnerability is incomparable. ?hat is the moment when you begin to feel really someone who is thinking outside of the box and begins to create his own "art"(what hacking really is). this part really touched me as its my dream to find a vul and write a working exploit for it - hope this dream will become true one day. Thanks once again.

REPLY

NAEEM MALIK
9 MONTHS AGO

1

Assembly book link is broken and please update it I had to learn it

Step 1: Update Broken Link

Step 2:

Step 3:

REPLY

Share Your Thoughts

YOU
LOGIN TO COMMENT

Click to share your thoughts