Independent Study: Modern Windows Vulnerability Analysis & Exploit Development

Markus Gaasedelen, Spring 2014

Abstract: As of January 2014, the Microsoft Windows operating system series maintains over a 90% market share in the global market of computing[1]. This fact alone helps explain why Windows is the most commonly targeted platform for malicious exploitation by hackers, organizations, and nation states alike. With years of relentless exploitation, great strides have been made by Microsoft in securing their operating system through numerous exploit mitigation techniques from the Windows XP era onwards. This course will explore the tools, a number of mitigations, and their associated bypass techniques that are utilized in most modern exploits on the Windows platform. The outcome of this course will leave one with the ability to analyze real world vulnerabilities and develop reliable exploits from end to end for Windows XP – Windows 7 systems.

Prerequisite: This study requires one to have a deep understanding of low level architecture, specifically a firm grasp on reading & writing x86 assembly. Prior experience in at least basic Linux binary exploitation, reverse engineering, and assembly level debugging is assumed. Trivial & dated forms of exploitation (such as execstack) will not be covered in favor of more modern, relevant topics.

[1] http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0 – Desktop Operating System Market Share

Course Outline

Section One

A) Tools of the Trade – Establishing the Windows Toolkit
B) Writing Windows Shellcode
C) Windows Mitigations by Generation

Section Two

A) Defeating DEP & Writing ROP
B) Defeating ASLR
C) Deliverable – Real world DEP & ASLR bypass

Section Three

A) Exploiting Structured Exception Handlers (SEH)
B) Defeating SafeSEH and SEHOP
C) Deliverable – Real world SEH based exploit

Section Four

A) Use After Free / Pointer Issues
B) Browser Exploitation
C) Deliverable – Real World Browser Exploit

Section Five

A) Closed Source Vulnerability Discovery
B) Source Auditing / Open Source Vulnerability Discovery
C) Deliverable – Unique Bugs & Crashes

Course Timeline & Grading

Timeline

The tentative course timeline is as follows

Jan. 23	
Jan. 30	Tools of the Trade – Establishing the Windows Toolkit
Feb. 6	Writing Windows Shellcode & Windows Mitigations by Gen.
Feb. 13	Defeating DEP & Writing ROP
Feb. 20	Defeating ASLR
Feb. 27	**Deliverable #1 Demo – Real world DEP & ASLR Bypass**
Mar. 6	Exploiting SEH & Defeating SafeSEH and SEHOP
Mar. 13	**********Spring Break**********
Mar. 20	**Deliverable #2 Demo – Real World SEH Based Exploit**
Mar. 27	Use After Free / Pointer Issues
Apr. 3	Browser Exploitation
Apr. 10	Browser Exploitation / Additional deliverable work time
Apr. 17	**Deliverable #3 Demo – Real World Browser Exploit**
Apr. 24	Closed Source Vulnerability Discovery
May. 1	Source Auditing / Open Source Vulnerability Discovery
May. 8	**Deliverable #4 Demo – Unique Bugs & Crashes**

Grading

Grades for the course are based on the deliverables outlined throughout this document. There are four expected deliverables, each with a set overall grade weight of 25% each. The deliverables are expected to be demoed in real time, with the exploit generation scripts and weaponized exploit submitted with a write-up on the same day as the presentation / demo.

Section One

Tools of the Trade – Establishing the Windows Toolkit

Having the proper tools and development environment is an important part of being an effective exploit writer. A number of customized debuggers, scripts, disassemblers, and various binary/file editors make up most of the tools found in a typical exploit writing environment.

1. WinDBG
    a. Getting Started with WinDBG
        i. http://t.co/Bc8niTBP9r
        ii. http://t.co/zHMGbAC8ou
        iii. http://t.co/kSbmLsuuC9
    b. Common WinDBG Commands
        i. http://www.windbg.info/doc/1-common-cmds.html
    c. WinDBG Cheat Sheets
        i. http://theartofdev.wordpress.com/windbg-cheat-sheet/
        ii. http://labs.snort.org/awbo/windbg.txt
    d. Mona.py v2 with WinDBG
        i. https://www.corelan.be/index.php/2012/12/31/jingle-bofs-jingle-rops-sploiting-all-the-things-with-mona-v2/
    e. Misc WinDBG Resources
        i. http://stackoverflow.com/questions/4946685/good-tutorial-for-windbg
        ii. https://www.corelan.be/index.php/2013/01/18/heap-layout-visualization-with-mona-py-and-windbg/

2. Immunity Debugger
    a. IMMUNITY DEBUGGER—THE BEST OF BOTH WORLDS
        i. Chapter 5, Gray Hat Python
    b. Win32 Exploitation with Mona.py
        i. http://wmsmartt.wordpress.com/2011/11/08/win32-exploitation-with-mona-py-getting-set-up-part-i/
        ii. http://wmsmartt.wordpress.com/2011/11/09/win32-exploitation-with-mona-py-part-ii-configuration-and-basics/
    c. Mona.py Manual
        i. https://www.corelan.be/index.php/2011/07/14/mona-py-the-manual/
    d. Misc Immunity Resources
        i. http://www.exploit-db.com/download_pdf/16124/
        ii. https://www.corelan.be/index.php/2010/01/26/starting-to-write-immunity-debugger-pycommands-my-cheatsheet/
        iii. http://tuts4you.com/download.php?list.72

3. Metasploit / Kali
    a. Writing Metasploit Modules
        i. http://www.corelan.be:8800/index.php/2009/08/12/exploit-writing-tutorials-part-4-from-exploit-to-metasploit-the-basics/
    b. **More can be added in future iterations if necessary

4. OllyDbg
    a. Debugging Fundamentals for Exploit Development
        i. http://resources.infosecinstitute.com/debugging-fundamentals-for-exploit-development/
    b. OllyDbg Tricks for Exploit Development
        i. http://resources.infosecinstitute.com/in-depth-seh-exploit-writing-tutorial-using-ollydbg/
    c. **Waived ~ enough experience

5. IDA Pro
    a. **Waived ~ enough experience
6. Additional Suggested Tools:
    a. Python 2.7.x
    b. A Hex Editor
    c. 010 Editor
    d. Cygwin
    e. Notepad++
    f. NASM
    g. VMWare Workstation / Player
    h. **A more extensive list can be made

Additional applicable Windows toolkit resources and articles:

https://www.corelan.be/index.php/2009/09/05/exploit-writing-tutorial-part-5-how-debugger-modules-plugins-can-speed-up-basic-exploit-development/

http://x9090.blogspot.com/2010/03/tutorial-exploit-writting-tutorial-from.html

Shellcoding is an important part of exploitation, and shellcoding on windows tends to differ a little bit from its Linux counterpart. The resources below provide a good foundation for developing Windows based shellcode.

Introduction to Win32 shellcoding

https://www.corelan.be/index.php/2010/02/25/exploit-writing-tutorial-part-9-introduction-to-win32-shellcoding/

Writing Optimized Windows Shellcode in C

http://www.exploit-monday.com/2013/08/writing-optimized-windows-shellcode-in-c.html

The Wild World of Windows

The Shellcoder's Handbook, Chapter 6

Windows Shellcode

The Shellcoder's Handbook, Chapter 7

The Art of Win32 Shellcoding

http://www.codeproject.com/Articles/325776/The-Art-of-Win32-Shellcoding

Win32 Egg Hunting

https://www.corelan.be/index.php/2010/01/09/exploit-writing-tutorial-part-8-win32-egg-hunting/

Windows Mitigations by Generation

Understanding the main mitigations implemented for the Windows platform is an important part of being able to identify some of pitfalls one may encounter while developing exploits for Windows. The timeline below outlines a number of the core security elements added to Windows on a per generation/service pack basis.

2001 – Windows XP

- Security for mitigating binary exploitation on Windows is virtually non-existent at this point

2004 – Windows XP Service Pack 2

- DEP
- SafeSEH,
- GS Cookies
- Stack & Heap marked non-executable

2006 – Windows Vista

- ASLR is implemented, applies to stack/heap/images.
- Hardened Heap
- SEHOP

2009 – Windows 7

- DEP & ASLR further improved
- Security wise, not too different from Vista, but widespread adoption.

2012/2013 – Windows 8/8.1

- Windows 8 came with a huge focus on beefing up security
- The presentation below explains most of these additions and upgrades by a few of the very engineers that developed them
- http://media.blackhat.com/bh-us-12/Briefings/M_Miller/BH_US_12_Miller_Exploit_Mitigation_Slides.pdf

Section Two

Defeating DEP & Writing ROP

Hardware Data Execution Prevention is a mitigation technique that was implemented in Windows XP SP2. DEP is used to mark segments of memory as non-executable so that if control is ever gained by an attacker, they cannot simply jump directly to injected shellcode and start executing it.

There are a number of ways to disable DEP once execution flow is achieved, and that's generally done through a wonderful technique known as Return Oriented Programming (ROP).

Defeating DEP

http://bernardodamele.blogspot.com/2009/12/dep-bypass-with-setprocessdeppolicy.html

http://seclists.org/fulldisclosure/2010/Mar/att-553/Windows-DEP-WPM.txt

http://www.exploit-db.com/wp-content/themes/exploit/docs/17914.pdf

Chaining DEP with ROP

http://www.fuzzysecurity.com/tutorials/expDev/7.html

https://www.corelan.be/index.php/2010/06/16/exploit-writing-tutorial-part-10-chaining-dep-with-rop-the-rubikstm-cube/

http://blog.harmonysecurity.com/2010/04/little-return-oriented-exploitation-on.html

http://blog.harmonysecurity.com/2010/04/little-return-oriented-exploitation-on_16.html

http://www.exploit-monday.com/2011/11/man-vs-rop-overcoming-adversity-one.html

ASLR is short for Address Space Layout Randomization. It typically goes hand in hand with DEP, as one without the other proves to be completely worthless. ASLR randomizes the location of allocated image sections, so that an exploit cannot simply use hardcoded addresses in its payload.

Using non-ASLR modules

http://exploitresearch.wordpress.com/2012/06/23/abusing-non-aslrd-modules-on-windows-7/

http://nakedsecurity.sophos.com/2013/10/11/anatomy-of-an-exploit-ie-zero-day-part-1/

Heap Spraying

http://www.fuzzysecurity.com/tutorials/expDev/8.html

http://www.fuzzysecurity.com/tutorials/expDev/11.html

http://www.exploit-db.com/wp-content/themes/exploit/docs/17914.pdf

https://www.corelan.be/index.php/2013/02/19/deps-precise-heap-spray-on-firefox-and-ie10/

https://www.youtube.com/watch?v=GOgIaK7RZ4o

http://ecee.colorado.edu/~ekeller/classes/fall2013_advsec/papers/heap-sprays-to-sandbox-escapes_issa0113.pdf

http://www.thegreycorner.com/2010/01/heap-spray-exploit-tutorial-internet.html

Memory Disclosure

http://media.blackhat.com/bh-us-12/Briefings/Serna/BH_US_12_Serna_Leak_Era_Slides.pdf

http://vreugdenhilresearch.nl/Pwn2Own-2010-Windows7-InternetExplorer8.pdf

http://security.stackexchange.com/questions/22989/how-leaking-pointers-to-bypass-dep-aslr-works

Additional ASLR Bypass related resources:

http://www.fireeye.com/blog/technical/cyber-exploits/2013/10/aslr-bypass-apocalypse-in-lately-zero-day-exploits.html

https://cyvera.com/brute-forcing-aslr-on-windows/

http://www.securityweek.com/aslr-bypass-techniques-appearing-more-frequently-attacks

http://www.slideshare.net/ffri/mr201308-understanding-bypassing-aslr-by-a-pointer-at-a-fixed-address-eng

Deliverable – Real world DEP & ASLR bypass

This exercise will challenge one's ability to write a reliable exploit that defeats DEP & ASLR for a pre-existing bug (or you can try to find your own!) in a piece of real world software. In an attempt to maximize the impact of a single exploit, see if you can construct such a payload for the vulnerable software that is functional on both a Windows XP (SP3) and Windows 7 machine.

Suggested Application Candidates:

Adobe XFA - CVE-2013-0640
Additional CVEs/Applications TBD

The deliverable should be in the form of a 2.7.x Python Script or Metasploit Module used to generate the payload, along with a sample payload capable of popping calc.exe on requested systems and a brief CTF-esque write-up describing the vulnerability & exploit.

Live demo on Windows XP SP3 & Windows 7 VMs required

Section Three

Exploiting Structured Exception Handlers (SEH)

Structured Exception Handlers are a form of catching exceptions thrown by a section of code. They're essentially a road map of where to send the execution flow if something goes horribly wrong. One can use this to their advantage by gaining control through manipulation of the SEH chain.

Structured Exception Handling

http://msdn.microsoft.com/en-us/library/windows/desktop/ms680657(v=vs.85).aspx

SEH Based Exploits

https://www.corelan.be/index.php/2009/07/25/writing-buffer-overflow-exploits-a-quick-and-basic-tutorial-part-3-seh/

https://www.corelan.be/index.php/2009/07/28/seh-based-exploit-writing-tutorial-continued-just-another-example-part-3b/

SEH Exploitation

http://fuzzysecurity.com/tutorials/expDev/3.html

SEH Based Overflow Exploit Tutorial

http://resources.infosecinstitute.com/seh-exploit/

SEH Stack Based Windows Buffer Overflow Tutorial

http://www.thegreycorner.com/2010/01/seh-stack-based-windows-buffer-overflow.html

General Technique for SEH Exploits

https://www.youtube.com/watch?v=ls_lfZdurHM

Additional SEH Exploitation Tutorials/Resources:

http://www.exploit-db.com/wp-content/themes/exploit/docs/17505.pdf

http://www.exploit-db.com/wp-content/themes/exploit/docs/17971.pdf

http://blog.pusheax.com/2013/05/exploit-writing-seh-based.html

After SEH overwrites & manipulation became a common vector for exploits to gain control of execution flow, some additional elements of security were added to the SEH paradigm starting from Windows XP SP2 onwards. Methods of defeating SEHOP and SafeSEH were developed soon after, and are detailed in a few forms below.

Bypassing SafeSEH + SEHOP

http://dl.packetstormsecurity.net/papers/bypass/SafeSEH_SEHOP_principles.pdf

Bypassing SEHOP

http://mirror7.meh.or.id/Windows/sehop_en.pdf

Bypassing Stack Cookies, SafeSEH, SEHOP, HW DEP and ASLR

https://www.corelan.be/index.php/2009/09/21/exploit-writing-tutorial-part-6-bypassing-stack-cookies-safeseh-hw-dep-and-aslr/

Exploiting IDM - Windows 7 x64 SafeSEH Bypass

https://www.youtube.com/watch?v=2CGbejqG-QM

Deliverable – Real world SEH based Exploit

This exercise will challenge one's ability to write a reliable exploit that defeats SEH/SafeSEH/SEHOP for a pre-existing bug (or you can try to find your own!) in a piece of real world software. In an attempt to maximize the impact of a single exploit, see if you can construct such a payload for the vulnerable software that is functional on both a Windows XP (SP3) and Windows 7 machine.

Suggested Application Candidates:

BlazeDVD 6.2 - CVE: 2006-6199
Kingsoft Writer 2012 8.1.0.3030 - CVE: 2013-3934
Easy LAN Folder Share Version 3.2.0.100 - CVE: 2013-6079
Total Video Player 1.3.1 - CVE: N/A
Watermark Master 2.2.23 - CVE: 2013-6935

The deliverable should be in the form of a 2.7.x Python Script or Metasploit Module used to generate the payload, along with a sample payload capable of popping calc.exe on requested systems and a brief CTF-esque write-up describing the vulnerability & exploit.

Live demo on Windows XP SP3 & Windows 7 VMs required

Section Four

Use After Free / Pointer Issues

Use after free and general pointer mishandling bugs can be some of the most powerful bugs one can find. Properly crafted data can almost always get control of execution flow. With OOP as common as it is, there's bound to be mishandled objects or memory allocations in a given program. UAF and general pointer issues tend to be at the heart of browser exploits, so this segment plays as what I feel is a necessary precursor to browser exploitation

Beginners Guide to "Use after free Exploits #IE 0-day Exploit Development"

http://0xicf.wordpress.com/2012/11/18/beginners-guide-to-use-after-free-exploits-ie-0-day-exploit-development/

Spraying the Heap [Chapter 2: Use-After-Free] – Finding a needle in a Haystack

http://www.fuzzysecurity.com/tutorials/expDev/11.html

The Difference Between Heap Overflow and Use After Free Vulnerabilities

http://grey-corner.blogspot.com/2010/03/difference-between-heap-overflow-and.html

Use-after-frees: That pointer may be pointing to something bad

https://www-304.ibm.com/connections/blogs/xforce/entry/use_after_frees_that_pointer_may_be_pointing_to_something_bad?lang=en_us

Insecticides don't kill bugs, Patch Tuesdays do (use-after-free)

http://d0cs4vage.blogspot.com/2011/06/insecticides-dont-kill-bugs-patch.html

CMarkup Use After Free Vulnerability – CVE-2012-4782

http://www.vnsecurity.net/2013/01/cmarkup-use-after-free-vulnerability-cve-2012-4782/

Happy New Year Analysis of CVE-2012-4792

http://blog.exodusintel.com/2013/01/02/happy-new-year-analysis-of-cve-2012-4792/

Exploits developed for vulnerabilities found in web browsers are some of the most valuable exploits for rather obvious reasons. It's one of the most common vectors for getting code execution onto a remote machine. Simply navigating to a maliciously crafted webpage is enough for an attacker to pwn one's machine.

With that said, browser exploits are among the most advanced exploits in industry. Modern iterations of IE, Firefox, and Chrome have some extensive security systems in place which generally require the chaining of multiple bugs in order to gain privileged execution, outside of a sandbox.

Client Side Browser Exploits

Gray Hat Hacking, Chapter 23

Heap Sprays to Sandbox Escapes: A Brief History of Browser Exploitation

http://blogbromium.files.wordpress.com/2013/01/heap-sprays-to-sandbox-escapes_issa0113.pdf

Bypassing Browser Memory Protections

http://www.volkanrivera.com/esp/wp-content/uploads/2008/08/sotirov-dowd.pdf

Heap spraying in Internet Explorer with rop nops

http://www.greyhathacker.net/?p=549l

Advanced Exploitation of Mozilla Firefox UAF Vuln – (MFSA 2012-22)

http://www.vupen.com/blog/20120625.Advanced_Exploitation_of_Mozilla_Firefox_UaF_CVE-2012-0469.php

Post-mortem Analysis of a Use-After-Free Vulnerability (CVE-2011-1260)

http://www.exploit-monday.com/2011/07/post-mortem-analysis-of-use-after-free_07.html

Attacking the Windows 7/8 Address Space Randomization

http://kingcope.wordpress.com/2013/01/24/attacking-the-windows-78-address-space-randomization/

CVE-2012-0769, the case of the perfect info leak – Flash Exploitation

http://zhodiac.hispahack.com/my-stuff/security/Flash_ASLR_bypass.pdf

Flash JIT – Spraying info leak gadgets

http://zhodiac.hispahack.com/my-stuff/security/Flash_Jit_InfoLeak_Gadgets.pdf

Pwn2Own Challenges: Heapsprays are for the 99%

http://dvlabs.tippingpoint.com/blog/2012/03/15/pwn2own-2012-challenge-writeup

Exploiting CVE-2011-2371 (FF reduceRight) without non-ASLR modules

http://gdtr.wordpress.com/2012/02/22/exploiting-cve-2011-2371-without-non-aslr-modules/

Anatomy of an exploit - inside the CVE-2013-3893 Internet Explorer 0day

http://nakedsecurity.sophos.com/2013/10/11/anatomy-of-an-exploit-ie-zero-day-part-1/

Exploiting and Analyzing CVE-2013-3893

http://sgros-students.blogspot.com/2014/01/exploiting-and-analysing-cve-2013-3893.html

Metasploit releases CVE-2013-3893 (IE SetMouseCapture Use-After-Free)

https://community.rapid7.com/community/metasploit/blog/2013/09/30/metasploit-releases-cve-2013-3893-ie-setmousecapture-use-after-free

Another Day, SpiderLabs Discovers Another IE Zero-Day

http://blog.spiderlabs.com/2013/10/another-day-another-ie-zero-day.html
http://blog.spiderlabs.com/2013/10/ie-zero-day-cve-2013-3897-technical-aspects.html

Zero-Day Season Is Really Not Over Yet – (MS12-063)

http://eromang.zataz.com/2012/09/16/zero-day-season-is-really-not-over-yet/

When a DoS Isn't a DoS - (MS11-003)

http://blogs.ixiacom.com/ixia-blog/ie-vulnerability/

Exploiting Internet Explorer 11 64-bit on Windows 8.1 Preview

http://ifsec.blogspot.com/2013/11/exploiting-internet-explorer-11-64-bit.html

Deliverable – Real World Browser Exploit

This exercise will challenge one's ability to write a reliable exploit that gains remote code execution through a Windows web browser for a pre-existing bug. In an attempt to maximize the impact of a single exploit, see if you can construct such a payload for the vulnerable software that is functional on both a Windows XP (SP3) and Windows 7 machine.

Suggested Bugs:

IE 6, 7, 8, 9 - MS12-063
IE 6, 7, 8 - MS11-003
IE 6, 7, 8, 9, 10, 11 - CVE-2013-3893
Or find/select your own from 2008 onwards

The deliverable should be in the form of a 2.7.x Python Script or Metasploit Module used to generate the payload, along with a sample browser payload capable of popping calc.exe on requested systems and a brief CTF-esque write-up describing the vulnerability & exploit.

Live demo on Windows XP SP3 & Windows 7 VMs required

Section Five

Closed Source Vulnerability Discovery

Searching for bugs in proprietary software or given a binary that you don't have the source for can be a tedious process, especially when given a large codebase to work with. This section will cover some of the tools and techniques used when it comes to searching for bugs sans source.

Vulnerability Discovery

The Shellcoder's Handbook, Second Edition, Chapters 15-17, 19-21

Vulnerability Analysis

Gray Hat Hacking, Section IV, Chapters 20-22, 25 26

Approaches for Discovering Security Vulnerabilities in Software Applications

https://blogs.oracle.com/security/entry/approaches_for_discovering_sec

A Bug Hunter's Diary

http://www.amazon.com/Bug-Hunters-Diary-Software-Security/dp/1593273851

The de facto book on fuzzing for Vulnerabilities, albeit a bit dated –

Fuzzing: Brute Force Vulnerability Discovery

http://www.amazon.com/Fuzzing-Brute-Force-Vulnerability-Discovery/dp/0321446119

Detecting Software Vulnerabilities Static Taint Analysis

http://tanalysis.googlecode.com/files/DumitruCeara_BSc.pdf

In the scenario that you're provided the source code for a given application or library, you're not longer restricted to only dredging through what you can make of disassembly, RE, and fuzzers while looking for bugs in most proprietary software. Below are some techniques and pointers as to what to look for while auditing source code, and even some tools for automating the process.

Source Code Auditing

http://pentest.cryptocity.net/code-audits/code-audits-101.html

http://pentest.cryptocity.net/code-audits/code-audits-102.html

A fantastic book on secure coding practices & code audit among other things

The Art of Software Security Assessment (TAOSSA)

http://www.amazon.com/The-Software-Security-Assessment-Vulnerabilities/dp/0321444426

Attacking the Code: Source Code Auditing

http://www.exploit-db.com/wp-content/themes/exploit/docs/108.pdf

Five Most Overlooked Open Source Vulnerabilities Found By Audits

http://www.informationweek.com/five-most-overlooked-open-source-vulnerabilities-found-by-audits/d/d-id/1063598

Source Code Analysis Tools

https://www.owasp.org/index.php/Source_Code_Analysis_Tools

http://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis

http://samate.nist.gov/index.php/Source_Code_Security_Analyzers.html

Joxean Koret - Interactive Static Analysis Tools for Vulnerability Discovery

http://www.slideshare.net/rootedcon/joxean-koret-interactive-static-analysis-tools-for-vulnerability-discovery-rooted-con-2013

Deliverable – Unique Bugs & Crashes

This exercise will challenge one's ability to discover real world bugs in production software. The application should work on both Windows XP (SP3) and Windows 7. The goal is to find some real bugs that result in memory corruptions or various crashes. Time permitting, triage any crashes or vulnerabilities found and attempt to gain control of EIP.

Application Candidates:

Find a piece of shareware, or some other application that you feel should have some bugs that aren't too crazy to discover and see what you can find.

The deliverable should be in the form of a write-up with an overview of the bug(s) you found, and a more detailed rundown on what you feel may be the most critical bug. What can be gained from it - Information leak? Code execution?

If control of EIP can be achieved, additional kudos shall be given for developing a POC payload that sets EIP to 0x41414141 or even a fully weaponized exploit in the spirit of previous deliverables. This however is not required, but could be a fun challenge should something notable be found.