H2HC MAGAZINE | 14ª EDIÇÃO | 2019

16ª EDIÇÃO | 2019

H2HC

CONFERENCE

O ESPÍRITO DO HACKING CONTINUA
VIVO EM CADA UM DE NÓS


H2HC
hacker
0xt313gr4m
/*4r4r4qu4r4
H4ck1ngT34m*/
l3ak
5m5
0daY
CYBER
ID-X2
3xpl01t
START
hackeia
HACKER AOU
黑客
0wn3d

CARTA DO EDITOR

Prezado(a) leitor(a),

É com grande satisfação que apresentamos a 14ª edição da H2HC Magazine! A premiação para autores de artigos foi atualizada. A partir desta edição, todos os autores ganharão ingresso gratuito para a H2HC do ano em que seu(s) artigo(s) for(em) publicado(s).

Gostaríamos de nos desculpar com os(as) leitores(as) pois, diferentemente do que diz a Carta do Editor da edição anterior desta revista, não foi possível publicar o artigo da coluna Fundamentos para Computação Ofensiva na edição online.

Não foi possível colocar o artigo da coluna "O exploit que eu vi" nessa edição da revista. No entanto, uma nova edição (não impressa, somente online) será publicada em breve e tal artigo estará presente. Fique ligado(a)! Site da revista: https://www.h2hc.com.br/revista

A H2HC Magazine é totalmente comprometida com a qualidade das informações aqui publicadas. Se você encontrou algum erro ou gostaria de agregar alguma informação, por favor, entre em contato! Mensagens de apreciação ou críticas são também muito bem vindas e servem de estímulo ao nosso trabalho.

Nosso e-mail é revista@h2hc.com.br.

Boa leitura!

H2HC

HACKERS TO HACKERS CONFERENCE

MAGAZINE

H2HC MAGAZINE

14ª Edição | Outubro 2019

DIREÇÃO GERAL

Rodrigo Rubira Branco

Filipe Balestra

DIREÇÃO DE ARTE / CRIAÇÃO

Letícia Rolim

IMPRESSÃO

Full Quality Gráfica e Editora

REDAÇÃO / REVISÃO TÉCNICA

Gabriel Negreira Barbosa

Ygor da Rocha Parreira

Raphael Campos Silva

João Guilherme Victorino

AGRADECIMENTOS

Fernando Mercês

Julio Moreira

Rafael Oliveira dos Santos

H2HC CONFERENCE

16ª Edição | Outubro 2019

ORGANIZAÇÃO

Firewalls security

patrocinadores PLATINUM

intel

tempest SECURITY INTELLIGENCE

CROWDFENSE VULNERABILITY RESEARCH HUB

patrocinadores GOLD

El Pescador a KnowBe4 company

Blue Frost Security

BYOS

Itaú

HACKING

ZERO NIGHTS 2018

APOIO

THREATINTELLIGENCE

SUMÁRIO

AGENDA

DIA 1

	H2HC	H2HC | University
08:20	**Credenciamento e entrega dos crachás H2HC**	
08:50	**ABERTURA - Filipe Balestra & Rodrigo Branco**	
09:10	**Keynote** *Dino Dai Zovi*	**Keynote** *Stefano Zanero*
10:10	**Machete: 9 Years of Cyber Espionage Operations in Latin America** *Veronica Valeros*	**Dissecting a linux kernel exploit - Part I** *Thais & Gustavo*
11:10	**Trashing like it is 1999 Unsolicited forensics on GPS trackers** *Matias Soler*	**Dissecting a linux kernel exploit - Part II** *Thais & Gustavo*
12:10	**LUNCH / ALMOÇO**	
14:10	**Bypassing a hardware based trusted boot through x86 CPU microcode downgrade -** *Alexander Ermolov*	**Analyzing phishing campaigns targeting Gmail users** *Elie Bursztein & Daniela Oliveira*
15:10	**Fertilising SnapDragons? Microsoft Windows on ARM** *Nikita Tarakanov*	**Como detectar uma bomba atomica?** *Lucas Ferreira*
16:10	**BREAK / INTERVALO**	
16:40	**Crash Analyzing with Reverse Tainting (Powered by Taintgrind)** *Marek Zmyslowski*	**Windows Objects Exploitation** *Bruno Oliveira*
17:40	**Under Pressure: Real world damage with TPMS spoofing** *Inbar & Raziel*	**NIDAVELLIR: Building Powerful Tools (Weapons?)** *Nelson Brito*

AGENDA

DIA 2

	H2HC	H2HC | University
10:00	**Keynote:** *Fermin Serna*	**Gamification Will Lead to Better Medical Device Resilience -** *Nina Alli*
11:00	**Oh memset, where did you go?** *Marion Marschalek*	**Raspy on Aircraft** *Bordini & Diego*
12:00	**LUNCH / ALMOÇO**	
14:00	**Modern Heap Exploitation: The poison NULL-byte** *Daniel Velazquez*	**Anti Tamper em Maquinas de Cartao** *Julio Della Flora*
15:00	**Embedded Research & Automation** *Brian Butterly*	**Network Reconnaissance Adventures in IPv6-Land** *Fernando Gont*
16:00	**BREAK / INTERVALO**	
16:30	**Using Binary Ninja to find format tring vulns in Binary Ninja** *Vasco Franco*	**Android kernel exploitation on Samsung Galaxy S7** *f0rb1dd3n & Alejo*
17:30	**Fuzzing TrustZone TEE to break Full Disk Encryption in Android** *Andrey Akimov*	
18:30	**ENCERRAMENTO**	

PALESTRAS E PALESTRANTES H2HC

Analyzing phishing campaigns targeting Gmail users

Elie Bursztein & Daniela Oliveira

Abstract: With over 1.4 billion active users and millions of companies entrusting it to handle their email, Gmail has a unique vantage point on how phishing groups operate. In this talk we Gmail telemetry to illuminate the differences between phishing groups in terms of tactics and targets. Then, leveraging insights from the cognitive and neuro-science fields on user's susceptibility and decision-making, we discuss how users from different demographic groups fall for phishing differently and how those insights can be used to improve phishing protections.

BIO: Elie Bursztein leads Google's security & anti-abuse research, which helps protect users against Internet threats. His research focuses on advancing the state of applied-cryptography, machine learning for fraud and abuse, at risk user protections, and web security. He is the author of 60+ scholarly publications for which he received 6 best papers awards. Elie gave over 20 talks at leading industry conferences and received multiple industry awards including the Back Hat Pwnie award. He was invited to give over 20 guest lectures to numerous universities including Stanford, Berkely and Tsing Hua. Elie's work is regularly covered by major news outlets including the Wall Street Journal, CBS, Forbes, Wired, the Huffington Post and CNN. Elie is a beret aficionado, tweets at @elie, and performs magic tricks in his spare time. Born in Paris, he received a Ph.D from ENS-cachan in 2008 before working at Stanford University and ultimately joining Google in 2011. He now lives with his wife in Mountain View, California.

BIO: Dr. Daniela Oliveira is the IoT Term Associate Professor in the Department of Electrical and Computer Engineering at the University of Florida. She received her PhD in Computer Science from the University of California at Davis. Her current research interests include understanding and addressing cyber deception and phishing in an interdisciplinary fashion. She received a National Science Foundation CAREER Award in 2012, a Presidential Early Career Award for Scientists and Engineers (PECASE) from President Obama, and the 2017 Google Security, Privacy and Anti-Abuse Award. Daniela is an experienced public speaker, having given talks at National Academy of Sciences (Distinctive Voices Program), USENIX ENIGMA Conference, TEDx, and in many universities. She is a National Academy of Sciences Kavli Fellow and a National Academy of Engineers Frontiers of Engineering Symposium Alumni. Her research has been sponsored by the National Science Foundation (NSF), the Defense Advanced Research Projects Agency (DARPA), the National Institutes of Health (NIH), the MIT Lincoln Laboratory, and Google. She was born and raised in Brazil and on her spare time she loves going to Disney World with her husband Marcio and her 10-old daughter Brooke. She is a dog lover and has a two-year-old German Shepherd, Wagner.

Android kernel exploitation on Samsung Galaxy S7

H2HC University: Ighor (f0rb1dd3n)

Abstract: Essa palestra irá mostrar a exploração de um heap overflow (0day) em um device driver do Galaxy S7, que permite privilege escalation.

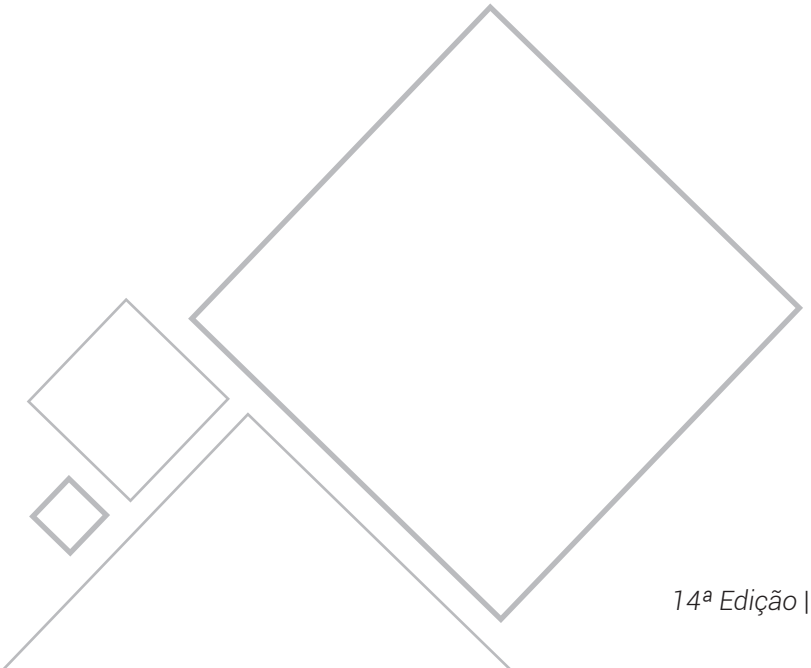
BIO: Security Researcher

Anti Tamper em Maquinas de Cartao

H2HC University: Julio Della Flora

Abstract: Com a supracitada palestra visamos apresentar técnicas de anti tamper em máquinas de cartão de crédito, bem como tentativas de bypass dessas tecnologias (quando aplicável).
Diversas máquinas de cartão serão abordadas, comumente modelos básicos e com baixo custo.
Técnicas de aquisição de dados em memórias flash trarão exemplos de obtenção de informações acerca do hardware do equipamento. Senhas de acesso padrão e outros dados importantes serão passíveis de descoberta explorando técnicas de extração de dados.Quando possível, mecanismos de comunicação como jtag, uart entre outros serão explorados para obtenção de informações.
Em resumo, a pesquisa propõe a apresentação de técnicas de exploração de hardware (nem sempre bem-sucedidas) em dispositivos embarcados pra transações financeiras.

BIO: Certificação Microsoft em servidores Windows, bacharel em Ciência da Computação, especialista em Redes de Computadores e Segurança de Dados, Mestre em Ciência da Informação, Especialista em Docência para o Ensino Superior. Atuou como professor de graduação e pós graduação em diversas faculdades como: Unopar, Unifil e FAG, ministrou cursos em parceria com o Senai-Londrina, Atuou como professor de Eletrônica Digital e Instrumentos de Medida na Fundação de Ensino Técnico de Londrina, Coordenador da Pós-graduação em Segurança da Informação e Ethical Hacking, Coordenador da Pós-graduação em Ethical Hacking e Cybersecurity EAD. Analista de segurança da informação e entusiasta em hardware hacking. Palestrou em conferências de segurança como H2HC, YSTS, RoadSec, entre outras. Possui 8 anos de experiência com tecnologia e segurança da informação. CVE's publicadas: CVE-2018-20823, CVE-2019-12762.



Bypassing a hardware based trusted boot through x86 CPU microcode downgrade

Alexander Ermolov

Abstract: It is widely known Intel CPU microcode is hardcoded into CPU ROM, and for security reasons, should be updated every time CPU is powered on, including situations like waking up from Sleep/Hibernation states. This is usually done by a microcode loader in UEFI BIOS. I've discovered a vulnerability in this loader, which allows tricking it to downgrade the CPU microcode.
One of the obvious consequences of this attack vector is removing fixes (implemented in microcode) for vulnerabilities like Spectre var2. However, I've found out the older versions of microcode allows to load the older versions of Intel ACMs (Authenticated Code Modules).

ACMs are a special code modules developed (and signed) by Intel to support some Intel security technologies, like Intel Boot Guard, Intel BIOS Guard, Intel TXT, Intel SGX. "Supporting" means serving as a Root of Trust. These modules are loaded into CPU L3 cache (sometimes called AC RAM) and executed from there. Like the other code, ACMs can be updated/fixed, and for security reasons running a downgraded version of an ACM is deprecated. This is maintained by a microcode and, like mentioned above, the old version of a microcode loads an old (associated) version of an ACM.
This opens up an opportunity to exploit patched vulnerabilities in ACMs influencing on the technologies they support. Which in turn leads to bypassing the trusted/measured boot (hardware-based).
In this talk I'm going to show how exactly this could be done on a real Intel TXT & Intel BIOS Guard protected platform.

BIO: Independent Security Researcher

Como detectar uma bomba atômica?

H2HC University: Lucas Ferreira

Abstract: Esta palestra irá apresentar o CTBT (Tratado de Proibição completa de Testes Nucleares) com foco nas tecnologias usadas para implementar o seu Sistema Internacional de Monitoramento, que coleta dados para detectar explosões de armas nucleares ao redor do mundo. O foco da apresentação será nas tecnologias usadas nos sensores e na análise dos dados. Também serão abordadas outros usos (monitoramento de mamíferos marinhos, previsões meteorológicas, alertas de tsunami, acompanhamento de vazamentos radioativos, etc.) para os dados coletados.

BIO: Lucas C. Ferreira é um administrador de sistemas com alguns (poucos :) cabelos brancos, que já trabalhou em grandes e pequenas empresas, sempre com foco em administração de sistemas e segurança da informação. Hoje é o líder da equipe de administradores de sistemas Unix e Linux do CTBTO, cuja missão é manter os sistemas de monitoramento de explosões nucleares "up-and-running". Lucas é também um membro da OWASP onde ocupa a posição de Líder do Capítulo de Viena (Áustria).

Crash Analyzing with Reverse Tainting (Powered By Taintgrind)

Marek Zmyslowski

Abstract: In recent years, fuzzing has become a popular and powerful method for vulnerability research. There are dozens of free and open frameworks available, with new ones arriving each month, but fuzzing itself is only part of the equation. Another part comes with triaging; or how to find only the relevant crashes when a fuzzer might find them in hundreds or even thousands. Often, these are sorted and binned based on the artefacts around the crash itself, but this is both naïve and superficial. In this talk, we will cover the use of taint analysis via 'reverse tainting' as a potent alternative.
This presentation will show how reverse tainting can be used as a part of the crash analysis. The audience will see how it can give an easy solution as to the reasons for a crash based on its inputs rather than its effects. File formats are structured containing different fields. With reverse taint, it is easy to find the connection between that crash and the particular field in the structure. It can be very helpful for the future analysis of why the crash occurred.

BIO: Currently Security Researcher @ Cycura where he is responsible for a different aspect of fuzzing services and vulnerability research. In the security industry for more than 12 years. Experience in the area of penetration testing, reverse engineering or vulnerability finding.

Dissecting a linux kernel exploit

H2HC University: Thais Hamasaki & Gustavo Scotti

Abstract: In this talk we will give an insight into Linux kernel exploitation, starting from a CVE report already discussed on https://blog.lexfo.fr/cve-2017-11176-linux-kernel-exploitation-part1.html.
We encourage you to read the blog post prior to the talk. Don't miss the opportunity to ask direct questions about this vulnerability and the techniques to write kernel mode exploits. We are going to give examples of writing bypasses and debugging techniques to help you on the development of your own exploit.
Since examples of exploit code are interesting but not really show the complexity of the actual development process, our focus will be to walk through the complexity of taking advantage of a vulnerability report to write a real-life (and working!) exploit.

BIO: Thaís Moreira Hamasak is a security researcher at Intel STORM (STrategic Offensive Research & Mitigations Team). Previous to that, she worked as a malware researcher @F-Secure, with focus on static analy sis, reverse engineering and logical programming. Thaís started her career within the anti-virus industry working on data and malware analysis, where se developed her knowledge on threat protection systems. She won the "best rookie speaker" award from BSides London for her very first talk about "Using SMT solvers to deobfuscate malware binaries". Recent research topics include binary deobfuscation, generic unpacking and static analysis automation. She is an active member of the Düsseldorf Hackerspace, where she also leads the groups for Reverse Engineering and x86 Assembly. In her free time, you can find Thaís buiding tools, cooking or climbing somewhere offline.

BIO: Gustavo Scotti, a.k.a. csh is one of those guys who curiosity drives his life. If I am not learning new stuff, experimenting with dangerous things, or living life at its fullest, csh is a dull boy. I am an enthusiast of mechanical engineer, electrical engineer, computer engineer, physics engineer, and music engineer. To fund all my hyperactive mind, I work as a Security Researcher at Intel Corporation, hacking cool stuff, at the lowest level you could imagine. Known by some exploits, axur05 e-zine, reversed engineered the PS2, wrote some rootkits, sniffers, and some other stuff.

Embedded Research & Automation | Brian Butterly

Abstract: There are a lot of differences between testing software and hardware / embedded devices, two of the big ones being feedback and input. While there are often many options (i.e. virtualization, log files, debugging) when testing a defined piece of software, an embedded device might just be a black box with a button and an LED. Especially when wanting to automate testing or perform fuzzing a lack of feedback or even automated input can be a back breaking issue.

Luckily there are a lot of both simple and professional solutions for solving these challenges which range from cheap Arduino based bread-boards over well-chosen sensors to Logic Analyzers. The talk will give an overview how security researchers can utilize knowledge from the maker scene to quickly put together custom test rigs which will enable them to perform full or at least semi-automated testing.

First, we will have a closer look at typical outputs of embedded devices (LEDs, sound, vibration and other actors) and see how these can be converted into digital information which can be collected at any chosen point in time. We will then utilize the collected information as simple feedback for fuzzing scripts. While doing so we will also try a few options to identify the reboot / crash of a device.

Afterwards we will have a look at the other side, input. By utilizing the same tool set we will have a look how a researcher is able to automate the pressing of a physical button or a click on a touch screen and various other inputs a device might accept. Here we will also cover options to crash and restart (reset) a device at will.

Eventually we will use the shown skillset to create a brute-forcer for an electronic safe, which has very limited input and output options.

BIO: Brian currently works in incident response in a very large and crazily diverse environment for a German company. There he aims at developing new methods for protecting even the strangest control systems and the overall surrounding networks. Still, at heart, he is an open minded security researcher and into breaking everything he can get his finger onto. Having worked in the areas of embedded-, hardware-, mobile- and telecommunications-security he has a lot of war stories and experience at hand and is always happy to share.

Fertilising SnapDragons? Microsoft Windows on ARM

Nikita Tarakanov

Abstract: Microsoft Windows has been out for quite a while. But WoA (Windows on ARM64) is quite a fresh beast. Several laptop makers have made laptops that have SnapDragon SoC and run on Windows ARM64 Edition. The first generation of those laptops is based on SnapDragon 835 SoC. The second (and the current one) generation is based on SnapDragon 850 SoC. This talk is about vulnerability discovery and exploitation of various components: of Windows on ARM64 itself and different parts of SnapDragon SoC.\

BIO: Nikita Tarakanov is an independent information security researcher. He has worked as an IS researcher in Intel corporation, Positive Technologies, Vupen Security, CISS. He likes writing exploits, especially for Windows NT Kernel. He won the PHDays Hack2Own contest in 2011 and 2012. Nikita has published several papers about kernel mode drivers and their exploitation. he is currently engaged in reverse engineering research and vulnerability search automation.

Fuzzing TrustZone TEE to break Full Disk Encryption in Android

H2HC University: Andrei Akimov

Abstract: ARM TrustZone is now utilized in all modern ARM-driven smartphones. This technology provides hardware isolation for secure processing of sensitive data. The idea of the technology is to divide digital world into two: Normal World and Secure World. While Normal World is normally a traditional Android or Linux with all its userspace and kernelspace, the Secure World is something mythical, not widely known and often without any public accessible documentation and source code.

Even Android kernel doesn't have access to the data processed in TrustZone. And besides this sensitive data, breach to TrustZone can lead to other amazing things like compromising Root Of Trust and achieving rootkit persistence.

We will focus on getting into TrustZone from Android userspace in smartphones of Samsung Galaxy series and its Trustonic implementation of Trusted Execution Environment (TEE). Trusted applications, or trustlets, executed there, is one of the windows to TEE, and they turned out to expose vast attack surface. While they are custom format binaries, designed to run in a special environment, it is still possible to run AFL on them. We will show you our approach to automatically discover vulnerabilities in Trustonic trustlets with such a cool way as having proved itself feedback-driven fuzzing.

BIO: Security researcher from Russia. Likes new technologies and cunning tricks that could be done with them. Focusing mostly on security analysis of binaries, dived into different CPU architectures, operating systems and technology stacks. So looking for general ways and universal approaches for exploring and hunting for bugs in them.

Gamification Will Lead to Better Medical Device Resilience

H2HC University: Nina Alli

Abstract: Gamers don't usually start their games by reading the manufacturers directions (amirite?) for the how-to's on a game, we just get in there and figure it out. Hackers use the same practice when getting into a system, learn and break by doing. This talk will go through what the medical device manufacturer community can learn from gaming to better secure their devices.

BIO: Nina is the Executive Director of the Biohacking Village: bringing together security researchers, to integrating Medical Device Manufacturers, Citizen Scientists, and Hands-On lab together to share findings, discover vulnerabilities, and existing solutions, unmet needs, opportunities, market and path to commercialization. Nina is currently working on a multi-industry cybersecurity resilience model that includes operating model, plan impacts, linkages to industry frameworks to implement best practices and integration for an improved operating and defensive alignment with increased literacy for patients, medical device manufacturers, legal and federal governance, and sustainability. Nina is a guru of trivial knowledge and RPG fan, especially the ass kicking games. For exercise, she run from rabid dogs, wrestle alligators while simultaneously participating in eating contests, and running for public office. For fun, she drives with her eyes open, plays hopscotch in the rain, race big wheels, has staring contests with wolverines, and passes out band aids to gunshot victims. Her favorite word is "interesting", since it has multiple meanings and all appear positive on the surface.

Keynote | Dino Dai Zovi

BIO: Dino Dai Zovi is an information security industry veteran and entrepreneur. Dino is also a regular speaker at information security conferences having presented his independent research at conferences around the world including DEFCON, BlackHat, and CanSecWest. He is a co-author of the books "The iOS Hacker's Handbook" (Wiley, 2012), "The Mac Hacker's Handbook" (Wiley, 2009) and "The Art of Software Security Testing" (Addison-Wesley, 2006). He is best known in the information security community for winning the first PWN2OWN contest at CanSecWest 2007.

Keynote | Fermin Serna

BIO: Fermin J. Serna is a Computer Science Engineer graduated at the prestigious Madrid's UCM currently working as Chief Security Officer at Semmle responsible of protecting corporate assets as well as running the security research team focused on open source security.
Previously to Semmle he served as Head of Product Security at Google for almost 8 years where he build, run and oversaw the application security program for Google products. Fermin has also worked at Microsoft at the MSRC Engineering team where he envisioned and built the industry recognized EMET tool. Fermin also served as CTO and co-founder of NGSEC and S21SEC in Spain.
Fermin has found, been credited and published multiple security vulnerabilities on software developed by Microsoft, Google, Adobe, Oracle and open source (dnsmasq, glibc, ...). Because of this Fermin has

been recognized with multiple awards including a RootedCon lifetime achievement award and two nominations, one winner, of a Pwnie award for Best client side bug in 2016. Fermin is also a regular speaker at security conferences such as BlackHat, Syscan, Bluehat, H2HC, Rootecon, DeepSec, Source, Summercon, ...

Machete: 9 Years of Cyber Espionage Operations in Latin America

Veronica Valeros

Abstract: Since early 2011, a threat actor has been conducting espionage operations in Latin America using an espionage tool known as Machete or Ragua. In this talk we will present the analysis of Machete based on the collection, reverse engineering, and analysis of more than a hundred Machete samples from 2011 to 2019. The large corpus of samples allowed us to study changes in its features and to map the gradual evolution of Machete from its creation until today. Our talk will focus on the technical aspects of this malware and the analysis of the decoy documents used in the spear phishing campaigns. Finally we will discuss how Machete managed to stay operational to this day.

BIO: Veronica is a researcher and intelligence analyst from Argentina. Her research has a strong focus on helping people and involves different areas from wireless and bluetooth privacy issues to malware, botnets and intrusion analysis. She has presented her research on international conferences such as BlackHat, EkoParty, Botconf and others. She is the co-founder of the MatesLab hackerspace based in Argentina, and co-founder of the Independent Fund for Women in Tech. She is currently the director of the CivilSphere project at the Czech Technical University, dedicated to protect civil organizations and individuals from targeted attacks. / @verovaleros

Modern Heap Exploitation: The poison NULL byte

Daniel Medina Velazquez

Abstract: In this talk we will go through the journey of heap exploitation by showing how a simple NULL-byte off-by-one vulnerability can be used to obtain arbitrary code execution within a target using one of the latest versions of Glibc. Glibc has been hardened over time to prevent heap exploitation, however, as we will show, it is still possible to bypass some of these checks. The presentation will cover the techniques and tricks used to successfully exploit the vulnerability in a realistic target with constrained heap manipulation.

BIO: Daniel Medina Velazquez is a security researcher at Intel's Offensive Security Research group. His main interests are embedded systems security, exploit development, and micro-architectural attacks.

Network Reconnaissance Adventures in IPv6 Land

H2HC University: Fernando Gont

Abstract: As a result of the exhaustion of the IPv4 free address space, more and more sites are becoming dual-stacked -- that is, available over both IPv4 and IPv6. Due to the vast size of the IPv6 address space, traditional network reconnaissance techniques (such as "ping sweeps") are no longer effective, and thus alteranative ones need to be devised and evaluated. In this presentation, we will explore the topic of IPv6 network reconnaissance from a practical perspective, with real-world examples that employ a number of open source IPv6 tools we have developed. Additionally, we will provide insights about the most effective IPv6 network reconnaissance techniques, and some key findings resulting from our own exploration of the IPv6 Internet.

BIO: Fernando Gont specializes in the field of communications protocols security, working for private and governmental organizations from around the world.

Gont has worked on a number of projects for the UK National Infrastructure Security Co-ordination Centre (NISCC) and the UK Centre for the Protection of National Infrastructure (CPNI) in the field of communications protocols security. As part of his work for these organizations, he has written a series of documents with recommendations for network engineers and implementers of the TCP/IP protocol suite, and has performed the first thorough security assessment of the IPv6 protocol suite.

Gont is currently working as a security consultant and researcher for SI6 Networks (). As part of his work, he is active in several working groups of the Internet Engineering Task Force (IETF), and has published 30 IETF RFCs (Request For Comments) and more than a dozen IETF Internet-Drafts. Gont has also developed the SI6 Network's IPv6 Toolkit () -- a portable and comprehensive security asessment toolkit for the IPv6 protocol suite, and the IoT-toolkit (-- a security assessment toolkit for IoT devices.

Gont runs the IPv6 Hackers and the IoT Hackers mailing-lists (), and has been a speaker at a number of conferences and technical meetings about information security, operating systems, and Internet engineering, including: CanSecWest 2005, Midnight Sun Vulnerability and Security Workshop/Retreat 2005, FIRST Technical Colloquium 2005, Kernel Conference Australia 2009, DEEPSEC 2009, HACK.LU 2011, DEEPSEC 2011, LACSEC 2012, Hackito Ergo Sum 2012, German IPv6 Kongress 2014, H2HC 2017, Positive Hack Days 8, Hack In Paris 2018, and Troopers 2018. Additionally, he is a regular attendee of the Internet Engineering Task Force (IETF) meetings.

More information about Fernando Gont is available at his personal web site: .

NIDAVELLIR: Building Powerful Tools (Weapons?)

H2HC University: Nelson Brito

Abstract: Ten years ago a prototype tool has been built, which was suppose to automate the penetration tests tasks... This prototype tool served as the basis for other projects that came along these years, such as: POP, ESF, T50 and Inception. Originally named Exploit Next Generation, the tool had a modular approach with 71 files and 13,552 lines of C code and Assembly, using a variety of techniques to successful exploit a vulnerability. This presentation will talk about how to create and build a working prototype tool, covering some basic and advanced concepts to illustrate the strategy adopted (fingerprinting, assembly components, shellcode, polymorphism, etc.) -- half of the presentation will be code demonstration.

BIO: I'm merely another cybersecurity thinker and philosopher, occasionally researcher and enthusiast, addicted to computer and network (in)security, being the creator of the T50 and the only Brazilian to speak at the extinct PH-Neutral (invite-only) in Berlin. As a sought-after speaker I've presented in some of the most notorious conferences in Brazil: BSides, H2HC, YSTS, ROADSEC, etc. My researches are: POP, ESF, T50 and Inception — highlight to T50, a tool widely used by companies validating their infrastructures, incorporated by ArchAssault, BackTrack, BlackArch, Debian, Kali and Ubuntu.

Oh memset, where did you go?

Marion Marschalek

Abstract: A compiler's code optimization is a scary beast. It tends to take over the thinking for a developer, kneads the input source into much smaller, faster and elegant output code, and in general happens to be very good at that too. Big help in this undertaking are so called compiler built-ins and intrinsics, which, as it turns out, are essential to study should one be interested in how compilers "disappear" function calls.

A common favorite to study is libc's memset function, which is known to occasionally fall victim to compiler optmization. Dead store elimination tends to think erasing content from memory is rather useless; we security folks disagree. By looking closer at how the compiler uses built-in functions, applies code inlining and chooses between call or inliner, we can learn a lot about its impact on security and potential ways for attackers to abuse this compiler behavior.

BIO: Marion Marschalek is a former Malware Analyst and Reverse Engineer who recently started work at Intel in order to conquer the field of low level security research, where she nowadays spends an unusual amount of time looking at compiler source code. She has spoken at all the conferences and such, and seen all the things, and is one of the happiest hackers out there. Also, she runs a free reverse engineering bootcamp for women, because the world needs more researcherettes.

Raspy on Aircraft

H2HC University: Bordini & Diego

Abstract: A cada dia as aeronaves se tornam mais conectadas, ofertando serviços de acesso a internet a bordo para os passageiros, mas quais os riscos deste tipo de acesso? A pesquisa é resultado de um pentest realizado em um Airbus A320 onde foram explorados diversos vetores de ataque tendo como pivot um Raspberry a bordo de uma aeronave. O que seria possível fazer com um device tão minimalista? Quais os riscos para os tripulantes? Estas e outras perguntas serão respondidas durante a apresentação.

BIO: *Thiago Bordini*
Diretor de Inteligência Cibernética e Pesquisa na New Space Prevention Inteligência Cibernética. Executivo com mais de 20 anos de experiência no mercado de inteligência cibernética, atuando com análise e prevenção de ameaças e fraudes cibernéticas e disseminação de conteúdo educativo sobre o assunto para profissionais e empresas. MBA em Gestão Estratégica de TI e Segurança da Informação. Palestrante em diversos eventos nacionais e internacionais como YSTS, EkoParty, H2HC, CIAB, SecurityBSides, SANS, dentre outros. Membro da HTCIA (High Technology Crime Investigation Association) Organizador da Security BSides São Paulo

BIO: *Diego*
A tech guy. Pentester e Analista de Threat Intelligence, Diego vem dedicando os últimos 6 anos na área de TI em estudos voltados para a área de Segurança Ofensiva. Entusiasta de tecnologias Wireless (802.11, 802.15.4, 802.15.1, RFID, SDR), também tem grande interesse em pesquisas e estudos nas áreas de OSINT, Hardware Hacking, Inteligência/Contrainteligência e Pentest de forma geral.

Securing Cyber-Physical Systems: Moving Beyond Fear

H2HC University Keynote | Stefano Zanero

Abstract: Cyber-physical systems are attracting a lot of attention: attacks on connected cars received a lot of media exposure, as did attacks on industrial control systems, medical devices, and more generally on IoT devices. A lot of this interest is driven by vulnerability research (often in the form of "stunt hacking"). While useful and frankly engaging and attractive, this research does not really help answer the fundamental question of how to embed security analysis in design. In this talk, we will use automotive security as a case study to try to outline a risk-based design methodology that can be used to deal with our hyper-connected future.

BIO: Stefano Zanero received a PhD in Computer Engineering from Politecnico di Milano, where he is currently an associate professor with the Dipartimento di Elettronica, Informazione e Bioingegneria.

His research focuses on malware analysis, cyberphysical security, and cybersecurity in general. Besides teaching "Computer Security" and "Computer Forensics" at Politecnico, he has an extensive speaking and training experience in Italy and abroad. He co-authored over 90 scientific papers and books. He is a Senior Member of the IEEE, the IEEE Computer Society, and a lifetime senior member of the ACM. Stefano co-founded the Italian chapter of ISSA (*Information System Security Association*). He has been named a Fellow of ISSA. A long time op-ed writer for magazines, Stefano is also a co-founder and chairman of Secure Network, a leading information security consulting firm; a co-founder of 18Months, a cloud-based ticketing solutions provider; and a co-founder of BankSealer, a startup in the FinTech sector that addresses fraud detection through machine learning techniques.

Trashing like it is 1999: Unsolicited forensics on GPS trackers

Matias Soler

Abstract: Hidden in a dark corner, in the bottom shelf of a huge rack full of old industrial cooking equipment, luminaires, and other weird objects, a bucket full of secrets was awaiting to be found. Who would have imagined that what was once destined to die as landfill, would finally end up revealing the secrets of one of the biggest food distribution networks in Argentina.

Join me in this journey of discovery, guided by the will to unveil the secrets hidden in these devices, that will make you think: what else are we ignoring that leaves scary details of our lives/companies dumped in the trash?

During this talk I will walk you through the process I took, from doing an initial assessment, analyzing the potential threat vectors, trying and failing multiple times, then failing again, until in the end simplicity was the key. While we travel this path together, I will talk about embedded MCU protections, bypasses, dumping flashes on corroded devices, and how to interpret data.

BIO: Matias Soler is a Senior Security Researcher at Intel STORM team. Prior to that he worked for nine years at Immunity Inc where he has performed different tasks such as exploit development, reverse engineering, security research, and consulting. He has also taught trainings on binary and web exploitation. Matias has experience in both offensive and defensive areas within the information security field. He has previously presented at several international conferences such as Ekoparty, BlackHat Briefings and Infiltrate.

Under Pressure: Real world damage with TPMS spoofing

Inbar Raz & Raziel Einhorn

Abstract: Modern vehicles are equipped with Tire Pressure Monitoring System (TPMS) - a system that alerts the driver when the tire pressure is inappropriate. TPMS broadcasts an unencrypted data stream at known frequencies and has already attracted the attention of security researchers, who demonstrated the ability to spoof the transmission and cause an alert.

However, while previous research concluded that the worst case scenario would be forcing the driver to pull over for inspecting the vehicle - and by that facilitating some other illicit activities such as robbing or kidnapping - we will demonstrate a specific scenario which, using already-published research, could cause driver distraction, theoretically resulting in an unsafe driving scenario.

In this talk we will quickly go over the TPMS, show how to research it using Software Defined Radio and reach spoofing capabilities, and end by showing a proof of concept for our attack scenario. We will also include a "Fuckup" section where we will show you how we failed during the research and what we've learned.

BIO: Inbar has been teaching and lecturing about Internet Security and Reverse Engineering for nearly as long as he has been doing that himself, since the age of 9 on his Dragon 64. He spent most of his career in the Internet and Data Security field, and the only reason he's not in jail right now is because he chose the right side of the law at an early age.

Inbar specializes in outside-the-box approach to analyzing security and finding vulnerabilities, using his extensive experience of over 25 years at the IDF, Check Point, PerimeterX, and nowadays Argus Cyber Security, protecting the automotive domain from hackers.

BIO: Raziel has been working in the areas of Wireless Communication and Electronic Warfare for more than 10 years. Being fascinated by the possibilities that lie in the Cyberspace, he's looking for ways to use tools and knowledge from the RF world, in order to discover new attack surfaces and vectors. Nowadays Raziel is working at Argus Cyber Security, protecting the automotive domain from hackers.

Using Binary Ninja to find format string vulns in Binary Ninja

Vasco Franco

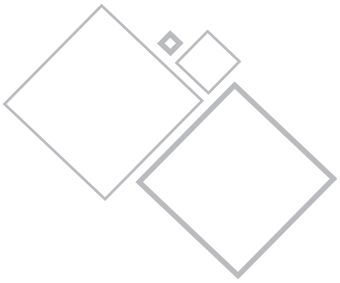
Abstract: This talk is based on the article chosen by the reviewers of the Paged Out Ezine as the best article: https://pagedout.institute/download/PagedOut_001_beta1.pdf

Windows Objects Exploitation

H2HC University: Bruno Oliveira

Abstract: This preso will drive the audience to know and understand some of the Windows objects that can be utilized during a kernel exploitation. Since the kernel holds every information regarding all aspects from the userspace, it is interesting to observe attack possibilities that could be done while on this privileged area.

BIO: Bruno Oliveira is MSc, computer engineer and Principal Security Consultant at Trustwave's SpiderLabs. During his career, always kept focus in offensive security, nowadays works full-time in penetration testing at Trustwave and still spends some extra (good) time on RE and exploit development. Spoken previously in many conferences around the globe as H2HC, YSTS, SHA2017, BlackHat, SOURCE, HackInTheBox, Ekoparty, THOTCON, AppSec USA, etc.



Explorando SEH através de um stack overflow - Parte 1

por Rafael Oliveira dos Santos

Nota do Editor: *Os procedimentos descritos neste artigo foram testados pela equipe da revista utilizando 2 máquinas virtuais rodando sobre o KVM (Ubuntu 18.04.1 LTS): Windows XP Professional SP2 32-bit VL (em inglês) e Kali 2019.3 64-bit. Na máquina virtual Windows foi instalado o Immunity Debugger v1.85; os arquivos vulnserver.exe e essfunc.dll copiados para essa mesma máquina virtual foram os baixados de [5] em 13/09/2019. Na máquina virtual Kali foi utilizado o SPIKE que já veio instalado (pacote versão 2.9-1kali6). Alguns valores presentes neste artigo (como, por exemplo, alguns endereços de memória) diferiram dos obtidos através dos testes realizados, mas tais diferenças são mínimas e logo não invalidam o artigo.*

Introdução

Este texto tem por objetivo explicar as etapas necessárias para a criação de um *exploit* para o serviço "*vulnserver.exe*", um binário desenvolvido por Steven Bradshaw [1] para aprendizagem na área de desenvolvimento de *exploits*. Outras publicações já abordaram formas de explorar esse serviço como, por exemplo: *Sh3llc0d3r* apresentou em seu blog [2] exemplos de exploração para os comandos "TRUN", "GMON", "HTER" e "KSTET", Marco Lugo apresentou, em [3], a exploração dos comandos "LTER", "TRUN", "HTER" e "GMON", e o próprio Stephen Bradshaw, em [4], abordou a exploração do comando "GMON".

Seguiremos um caminho semelhante ao que já foi abordado nas referências citadas e exploraremos o comando "LTER" através de um *stack-based buffer overflow*, ganhando execução de código pela corrupção do ponteiro para um *Structured Exception Handler* (**SE Handler**). Vale ressaltar que as técnicas e conceitos discutidos neste artigo dizem respeito ao Windows XP SP2.

Esse artigo foi dividido em duas partes. Nessa primeira parte do artigo, nos preocuparemos em expor os detalhes sobre o método utilizado para testar o comando, e sobre o estado inesperado que o serviço atingiu a partir de um pacote recebido pela rede. A segunda parte, que será publicada na 17ª edição desta revista, abordará a exploração desta falha.

Antes de prosseguir, é importante citar que vários fundamentos necessários para o que será abordado aqui foram baseados nos artigos publicados pelo Corelan Team [9] e em edições anteriores desta revista.

Ambiente e o binário

Para replicar todas as etapas, é recomendado utilizar duas máquinas virtuais que possam se comunicar via rede: Windows XP SP2 (servindo como vítima) e Kali Linux (atuando como atacante). O *debugger* adotado por este artigo foi o Immunity Debugger [10], que deve ser instalado na máquina virtual com Windows; os arquivos "vulnserver.exe" e "essfunc.dll", disponíveis em [5], devem ser copiados para essa mesma máquina virtual (ambos devem ficar na mesma pasta). O processo de *fuzzing* adotado utiliza o SPIKE, que deve ser instalado na máquina virtual com o Kali Linux (pode-se instalar com o comando **apt-get install spike**).

Para analisar o binário, precisamos abri-lo no *debugger* com: "*File -> Open* -> vulnserver.exe", conforme destaca a Figura 1 e, em seguida, apertar F9 (run) para permitir sua execução.

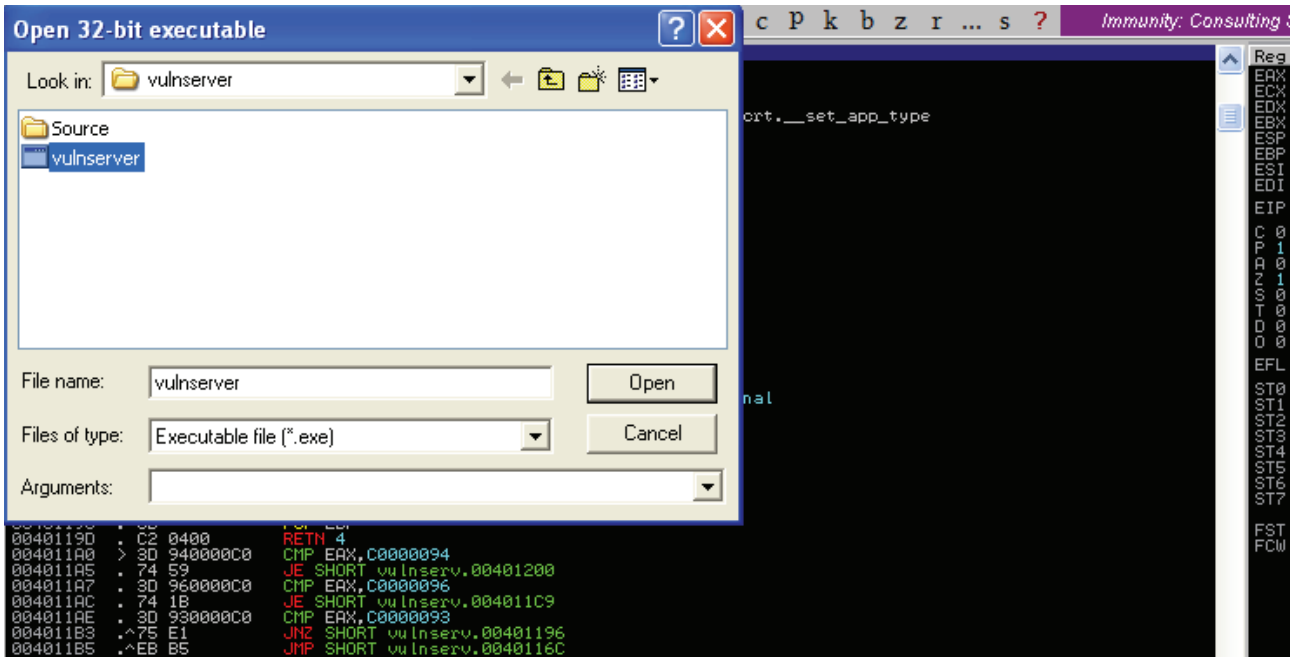


Figura 1: Vinculando o binário para análise no Immunity Debugger.

O serviço "vulnserver.exe", por padrão, fica escutando a porta 9999 e, para este estudo, o comando **LTER** será avaliado. Para testarmos seu funcionamento, podemos (através do Kali) estabelecer uma conexão com o comando nc [11], passando como primeiro parâmetro o endereço IP do Windows e como segundo parâmetro a porta do vulnserver.exe, conforme destaca a Figura 2.

Figura 2: Comandos disponíveis no vulnserver.exe.

Encontrando um estado inesperado

O SPIKE é basicamente uma API e um conjunto de ferramentas que permite a criação de testes de *stress* para protocolos de rede, conforme descreveu em detalhes Dave Aitel em sua apresentação na Blackhat EUA de 2002 [6]. Conforme mencionado anteriormente, utilizaremos tal ferramenta na máquina virtual atacante (Kali Linux). Para isso, criaremos um *template* para o SPIKE com o conteúdo da Listagem 1 (salvaremos em um arquivo com o nome fuzz.spk).

```
s_string("LTER");
s_string(" ");
s_string_variable("rafael");
s_string("\r\n");
```

Listagem 1: Conteúdo de fuzz.spk.

As *strings* utilizadas em "s_string" não serão modificadas a cada requisição enviada ao serviço. Já "s_string_variable" identifica um campo variável, onde, no nosso caso, "rafael" será o valor utilizado na primeira requisição mas, a partir da segunda, o SPIKE escolherá o valor a ser utilizado. Utilizando nosso *template*, o SPIKE se conectará ao serviço e enviará uma série de requisições como as do exemplo (meramente ilustrativo) presente na Listagem 2.

```
LTER rafael
LTER 1111111111
LTER AAAAAAAAAA
LTER “””AAAAAAAAAAAAAAAAAAAAAAAAAAAAA
LTER >)(!@#&!(@#*&)BBBBBBBBBBBCCCCCCCCCCCCCCC
...
```

Listagem 2: Exemplo (meramente ilustrativo) de requisições utilizando fuzz.spk.

Podemos iniciar o SPIKE através do comando “generic_send_tcp”, onde o primeiro parâmetro recebido representa o endereço IP da vítima (Windows XP no nosso caso), o segundo a porta de destino, o terceiro o arquivo que contém o *template* do SPIKE, o quarto o número de variáveis do *template* para serem “puladas” (no nosso caso, nenhuma) e, por fim, o quinto parâmetro representa o número de iterações a serem “puladas” (no nosso caso, nenhuma também). Mais informações podem ser obtidas em [14]. O comando a ser executado a partir do Kali Linux para iniciar o fuzzing está presente na Listagem 3.

```
# generic_send_tcp 10.0.0.35 9999 fuzz.spk 0 0
```

Listagem 3: Comando para iniciar o fuzzing.

Após alguns instantes, ao receber uma determinada requisição (“LTER /.:/AAAAA...” - 5000 A's, sem as aspas), o serviço ficou “congelado” no *debugger* e a seguinte mensagem foi exibida (na extremidade de baixo): *“Access violation when writing to [00C10000] – use Shift +F7/F8/F9 to pass exception to program”*. Em outras palavras, uma instrução tentou escrever no endereço 0x00C10000 disparando um “*access violation*” que foi capturado pelo *debugger*.

Nota do Editor: *Fica como desafio ao leitor: (1) Descobrir quais requisições estão sendo enviadas pelo comando generic_send_tcp mencionado anteriormente; e (2) Identificar qual foi a requisição recebida pelo “vulnserver.exe” que gerou o crash.*

A partir desse ponto, há três lugares que merecem atenção:

1) A instrução que causou a exceção, que no nosso caso foi **MOV [EDI], EDX** (instrução referente ao endereço presente no registrador **EIP**);
2) O estado de alguns registradores, no nosso caso, **EDI=0x00C10000**, **EDX=0x41414141** e **ESP=0x00C0F214** – Figura 3; e
3) O *call stack*, onde podemos descobrir qual função estava sendo executada (msvcrt.strcpy)[1] e os argumentos enviados para a função (ponteiros dos buffers de destino e origem).

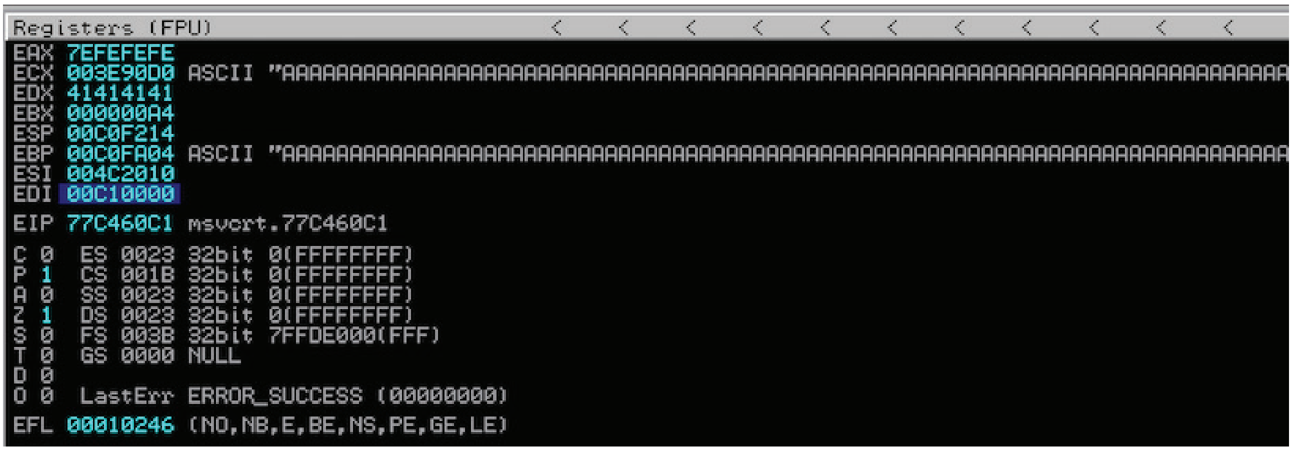


Figura 3: Estado dos registradores no momento da exceção.

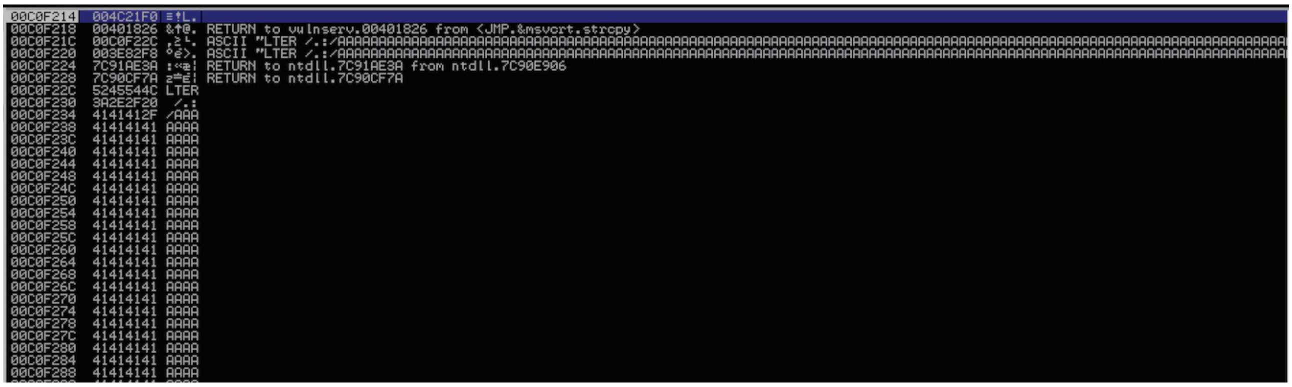


Figura 4: *Stack* no momento da exceção.

1 Essa informação pode ser verificada pressionando Alt+k (*Call stack*).

Observando o *dump* da *stack* (Figura 4), o Immunity Debugger nos fornece alguns detalhes adicionais na quarta coluna como, por exemplo, o uso da palavra RETURN. Com base nisso, temos o endereço de retorno da função strcpy (segunda linha identificada na Figura 4) e, por consequência, os parâmetros que foram passados para tal função [2]. Dessa forma, temos:

- [00C0F218] → Endereço de retorno: 00401826
- [00C0F21C] → Primeiro argumento (*buffer* de destino): 00C0F22C
- [00C0F220] → Segundo argumento (*buffer* de origem): 003E82F8

Olhando os argumentos recebidos pela função, percebemos que uma *string* grande com o conteúdo "LTER /.:/AAAAAAAAAA..." estava sendo copiada para o *buffer* de destino (que está presente na *stack*, mas em outro *stack frame*). Em um determinado momento, essa cópia disparou um "access violation". Veja que no momento da exceção, o endereço de memória apontado por **EDI** (00C10000) iria receber 0x41414141 (uma sequência de 4 letras "A" em ASCII).

Olhando as regiões de memória mapeadas para o processo, ilustradas na Figura 5 (o que pode ser feito com Alt+m), percebemos que a região de memória onde os dados estavam sendo copiados abrange o range 0x00C0F000 – 0x00C0FFFF e que o endereço 00C10000 não encontra-se mapeado. Logo, a exceção aconteceu pela tentativa de acessar (no caso escrever) um endereço de memória que não estava mapeado para o processo.

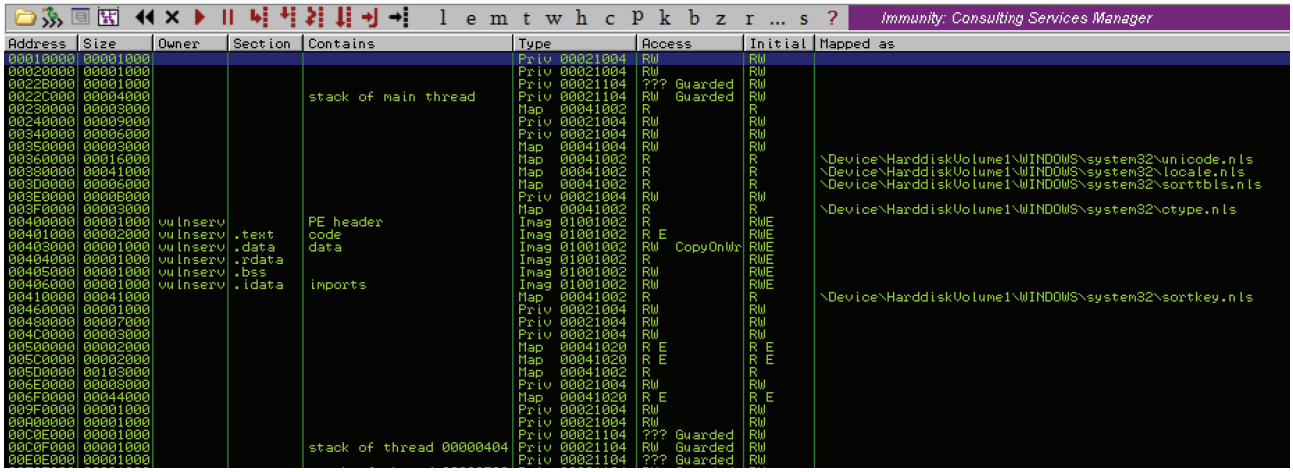


Figura 5: Regiões de memória no momento da exceção.

2 - Essa característica depende da *calling convention* utilizada. Para maiores informações, consultar a coluna "Fundamentos para computação ofensiva" das edições 7 e 8 dessa revista.

Como temos o código fonte de "vulnserver.exe" disponível, podemos verificar que, ao receber o comando LTER [7], a requisição recebida será tratada e, no nosso caso, a função **Function3** será chamada pois o caractere '.' está presente na string enviada ao servidor após "LTER ". Essa função (Function3) [8] reserva um *buffer* de 2000 bytes na pilha e depois tenta copiar para esse *buffer* (utilizando strcpy) a requisição recebida da rede depois de tratada; devido ao tamanho da requisição, um *stack-based buffer overflow* ocorre durante a cópia onde, ao acessar um endereço de memória não mapeado para o processo, acarretará na exceção analisada por nós.

Bom, até aqui já sabemos como causar um "access violation" no serviço, mas como poderíamos evoluir isso para execução de código? A resposta para essa pergunta se inicia com uma olhada no estado do "SEH chain" dessa *thread* (que pode ser feito com Alt+s), conforme ilustrado na Figura 6.

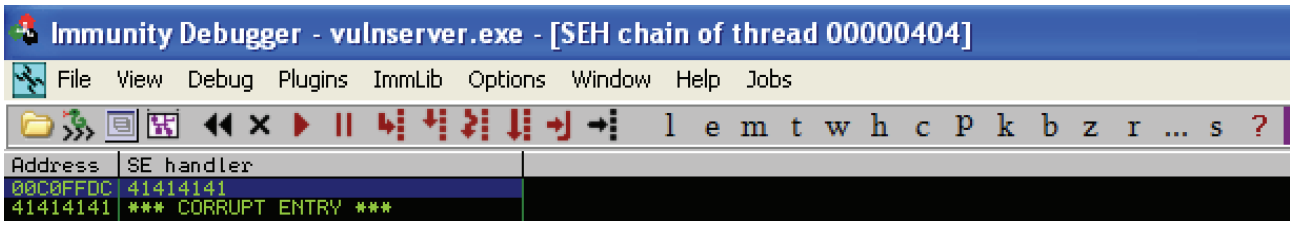


Figura 6: SEH *chain* no momento da exceção.

A Figura 6 mostra que o *overflow* na *stack* conseguiu corromper o campo denominado *SE handler*. Para entender melhor o que significa isso, vamos agora introduzir o SEH.

Entendendo o SEH

Uma exceção pode ser vista como um evento que acontece durante a execução de um programa e requer um tratamento fora do fluxo normal do mesmo [12]. Um exemplo de como isso pode ser implementado em uma aplicação está presente no Código 1.

```
try {
      // executar alguma coisa!
}
catch {
      // executa outra coisa com objetivo
      // de tratar a exceção (caso ocorra)
}
```

Código 1: Exemplo de tratamento de exceção.

SEH (*Structured Exception Handling*) [12] é um mecanismo do Windows utilizado para tratar exceções, podendo estas ser tanto de *hardware* quanto de *software*. Tal tratamento ocorre com base em uma lista encadeada denominada SEH *chain* (Figura 7). Cada elemento dessa lista possui uma estrutura chamada *_EXCEPTION_REGISTRATION_RECORD,* definida como mostra a Listagem 4.

```
typedef struct _EXCEPTION_REGISTRATION_RECORD {
    struct _EXCEPTION_REGISTRATION_RECORD *Next;
    PEXCEPTION_ROUTINE Handler;
} EXCEPTION_REGISTRATION_RECORD, *PEXCEPTION_REGISTRATION_RECORD;
```

Listagem 4: Estrutura _EXCEPTION_REGISTRATION_RECORD.

Conforme mostra a Listagem 4, a estrutura *_EXCEPTION_REGISTRATION_RECORD* é composta basicamente por 2 elementos:

- Ponteiro para a próxima estrutura de tratamento de exceções (chamaremos aqui de **Next SEH**), criando assim uma cadeia (*chain*) de estruturas. O último elemento desse SEH *chain* possui o valor 0xFFFFFFFF neste campo;
- Ponteiro para a rotina de tratamento de exceção (chamaremos aqui de **SE Handler**).

Quando uma exceção ocorre, o SEH *chain* é percorrido: os SE Handlers são as entidades executadas a fim de tratar a exceção.

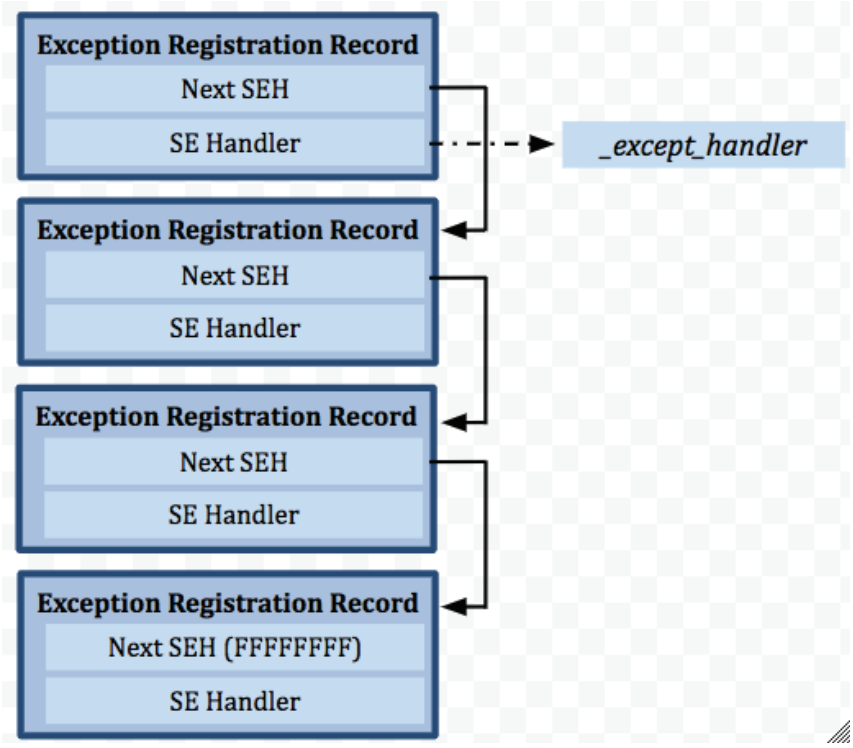


Figura 7: Exemplo de um SEH chain. Figura adaptada de [13].

Observando o estado do SEH *chain* (Figura 7) durante o momento que ocorreu a exceção, temos o cenário ilustrado pela Figura 8.

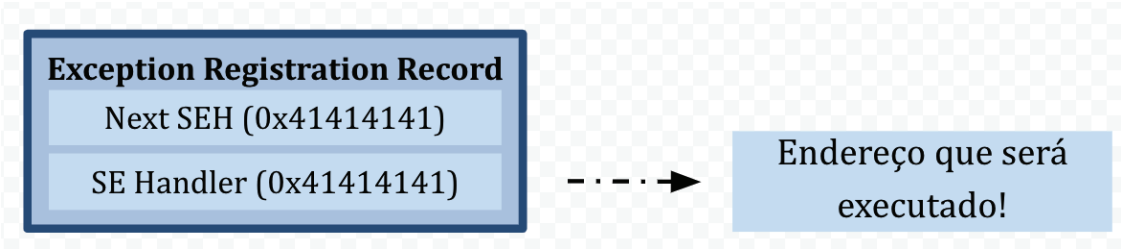


Figura 8: Ilustração dos valores de **Next SEH** e **SE Handler** após o *overflow*.

Tal cenário se mostra interessante uma vez que os valores sobrescritos (Figura 8) são valores que podemos controlar – incluindo o endereço do SE Handler, que potencialmente será executado para tratar exceções. Ambos os campos (**Next SEH** e **SE Handler)** da estrutura foram sobrescritos por "A" (0x41 em hexadecimal), lembrando que a origem desse "A" é a *string* gerada pelo SPIKE e enviada ao servidor.

Com isso, temos o controle do **SE Handler** e também uma forma de gerar uma exceção (como visto no tópico anterior). Portanto, podemos potencialmente desviar a execução de "vulnserver.exe" para um endereço controlado!

No próximo artigo vamos abordar desde a criação até os desafios encontrados na escrita de um *exploit* para essa falha. Até a próxima, pessoal!

Referências

[1] "The Grey Corner: Introducing Vulnserver." http://www.thegreycorner.com/2010/12/introducing-vulnserver.html

[2] "The sh3llc0d3r's blog." http://sh3llc0d3r.com/

[3] "GitHub - MarcoLugo/vulnserver: Different exploits created for Vulnserver." https://github.com/MarcoLugo/vulnserver

[4] https://resources.infosecinstitute.com/seh-exploit/

[5] https://github.com/stephenbradshaw/vulnserver

[6] "An Introduction to SPIKE, the Fuzzer Creation Kit - Black Hat" https://www.blackhat.com/presentations/bh-usa-02/bh-us-02-aitel-spike.ppt

[7] https://github.com/stephenbradshaw/vulnserver/blob/master/vulnserver.c#L260

[8] https://github.com/stephenbradshaw/vulnserver/blob/master/vulnserver.c#L149

[9] "Articles | Corelan Team." https://www.corelan.be/index.php/articles/

[10] "Immunity Debugger - Immunity Inc.." http://www.immunityinc.com/products/debugger/

[11] "nc(1): arbitrary TCP/UDP connections/listens - Linux man page." https://linux.die.net/man/1/nc

[12] "Structured Exception Handling (Windows) - MSDN - Microsoft." https://msdn.microsoft.com/en-us/library/windows/desktop/ms680657(v=vs.85).aspx

[13] https://www.securitysift.com/windows-exploit-development-part-6-seh-exploits/

[14] https://resources.infosecinstitute.com/intro-to-fuzzing/

Rafael Oliveira dos Santos

Formado em Ciência da Computação pela UFRJ, trabalho hoje no time de Segurança da Globo.com, mais especificamente na parte de segurança de aplicações. Minha paixão por segurança começou quando entrei para o CSIRT-UFRJ em 2010, grupo de extensão voltado para estudos na temática dentro da universidade. Tenho grande interesse na área de segurança ofensiva, desenvolvimento de exploits, e na criação de ferramentas que auxiliam desenvolvedores a tornarem seus códigos mais seguros.

Uma visão holística dos frameworks de Segurança da Informação

por Julio Moreira

Nota do Editor: A equipe da H2HC Magazine não possui muita experiência no assunto abordado por esse artigo. Logo, a revisão técnica não foi muito aprofundada. No entanto, o autor (Julio Moreira) é um excelente profissional com muita experiência nessa área.

Com a crescente adoção da tecnologia através da transformação digital, é fato que nosso cotidiano (e o das organizações) está cada vez mais dependente deste mundo virtual. Hoje, é praticamente "impossível" realizar tarefas simples como solicitar um transporte (táxi ou qualquer outro meio de locomoção), pedir alimentação ou mesmo atitudes mais simples, como conversar com os familiares sem estar conectado à Internet. Alguém aí se lembra de quando fez/recebeu uma ligação no telefone celular (*too old*, agora é smartphone...)? Provavelmente há mais tempo do que ter mandado e/ou recebido áudio via WhatsApp, certo!?!?

Se por um lado há a constatação desta dependência, por outro as consequências são, no mínimo, preocupantes. O número de vazamentos de dados, de acordo com relatório do *Ponemon Institute* [1], têm crescido de forma alarmante: no relatório *Cost of Data Breach Report de 2019* [1], a quantidade média de registros vazados foi de **25.575** registros, cerca de 3,9% maior que o ano de 2018. O tempo de detecção e contenção também aumentou, de **265,7** dias em 2018 para **279** dias em 2019 (crescimento de **4,3**%), causando uma perda financeira média de **US$ 3,92M** por vazamento versus US$ 3,86M de 2018 (aumento de 1,6%). A Tabela 1 sumariza esses dados.

	2019	2018	%
Quantidade média de registros vazados, por vazamento	25.575	24.615	3,9%
Tempo médio de descoberta e contenção do vazamento (em dias)	279	267,5	4,3%
Custo total médio de um vazamento de dados (em milhões de dólares)	3,92	3,86	1,6%

Tabela 1 - Comparativo do vazamento de dados de 2019 versus 2018.

Essa grande janela de oportunidade (dependência dos meios digitais e o crescimento constante do vazamento de dados) é o principal motivador deste artigo: um convite para você mergulhar numa jornada de exploração dos principais padrões (daqui por diante usaremos o termo *framework*) de segurança do mercado utilizados nas organizações e como eles se propõem a evitar que os vazamentos e outros problemas de segurança aconteçam. Esse texto é baseado nas referências listadas no fim do artigo.

Gestão de Segurança da Informação

Uma frase muito comum para os administradores é que *"Não se gerencia o que não se mede, não se mede o que não se define, não se define o que não se entende, e não há sucesso no que não se gerencia"* - William Edwards Deming. Logo, só podemos gerenciar a segurança da informação se fizermos a definição e medição dos processos e procedimentos de manipulação da informação. Para isso, a uma importante ferramenta de gestão de segurança da informação são os *frameworks.*

De acordo com Jason Wild, em artigo escrito para o site OriginIT [2], um *framework* de Segurança da Informação pode ser descrito como um conjunto de políticas, processos e procedimentos avaliados, aprovados e documentados, que são utilizados para definir como as organizações devem (ou deveriam) gerir suas informações, com o propósito de reduzir os riscos oriundos de vulnerabilidades existentes nos processos e nas tecnologias que processam, armazenam e transmitem as informações no dia-a-dia. Ainda de acordo com Wild, atualmente existem cerca de 250 *frameworks* de segurança diferentes, desenvolvidos e/ou adaptados para atender uma ampla variedade de empresas, setores, interesses (que vão desde a proteção da informação e do negócio em si até a regulamentação do mercado) e que são usados globalmente.

Vale ressaltar que os setores governamental e militar são os maiores desenvolvedores/incentivadores da criação desses modelos, já que a proteção das informações para esses setores é fator determinante de sucesso, desde os primórdios da sociedade: alguém aí lembrou da Cifra de César (Figura 1)?

Figura 1 - Cifra de Cesar [3]

Principais frameworks de Segurança

Cada *framework* foi desenvolvido com propósito e foco específicos. Por isso, muitas vezes o gerenciamento da segurança da informação nas organizações necessita que mais de um padrão seja utilizado, criando assim um *framework* sob medida. Os padrões mais usados são os seguintes:

ISO/IEC 27.001 e **ISO/IEC 27.002** – São as normas mais utilizadas e referenciadas (algo como o *Santo Graal*) quando o assunto é Segurança da Informação. Elas são complementares, já que a 27.001 é uma norma de gestão (define como se administrar um sistema), enquanto a 27.002 é um código de prática (um guia que diz como implementar os controles referenciados na norma de gestão).

NIST SP 800-53 – *Security and Privacy Controls for Federal Information Systems and Organizations* - Desenvolvida pelo *National Institute of Standards and Technology* (NIST) dos EUA, esta publicação é um catálogo de controles de segurança e privacidade de informações para sistemas e organizações federais. Também define um processo para selecionar controles para proteger operações organizacionais (incluindo missão, funções, imagem e reputação), ativos organizacionais, indivíduos, outras organizações e a Nação, de um grupo diversificado de ameaças que incluem ataques cibernéticos, desastres naturais, falhas estruturais e erros humanos (intencionais e não intencionais).

NIST CSF (*Cyber Security Framework*) – Resultado de um decreto do presidente dos EUA Barack Obama em 2013, o *Cyber Security Framework* foi criado em uma ação voluntária para reduzir os riscos cibernéticos à infraestrutura crítica[1]. Consiste em um conjunto de padrões, diretrizes e práticas para promover a proteção da infraestrutura crítica. A abordagem priorizada, flexível, repetível e econômica do padrão ajuda proprietários e operadores de infraestrutura crítica a gerenciar riscos relacionados à segurança cibernética. Apesar de suas origens, esse *framework* pode ser utilizado para aplicações diversas (não somente infraestrutura crítica).

NIST SP 800-37 - *Risk Management Framework for Information Systems and Organizations: A System Life Cycle Approach for Security and Privacy* – Esta publicação descreve o *Risk Management Framework* (RMF) e fornece diretrizes para a aplicação do RMF a sistemas de informação e organizações. O RMF fornece um processo disciplinado, estruturado e flexível para gerenciar riscos de segurança e privacidade que inclui a categorização de segurança das informações; seleção, implementação e avaliação de controles; controle padronizado de autorizações de sistema; e monitoramento contínuo. Passou por uma recente revisão para adequar-se ao **NIST CSF**.

1 - Infraestruturas Críticas são instalações, serviços e bens que, se forem interrompidos ou destruídos, provocarão sério impacto social, econômico, político, internacional ou à segurança nacional [5].

CIS CSC Top 20 – É uma publicação de guias de boas práticas para a segurança de computadores, orientada a controles mais técnicos (*hardening* e configurações de segurança). O projeto foi iniciado no início de 2008, como uma resposta a perdas massivas de dados sofridas por organizações que compunham a base da indústria de defesa dos EUA. A publicação foi desenvolvida inicialmente pelo Instituto SANS, a propriedade foi transferida para o *Council on Cyber Security* (CCS) em 2013 e depois para o *Center for Internet Security* (CIS) em 2015. Hoje esse framework conta com ferramentas automatizadas que fazem a verificação e apresentam quais controles não estão implementados.

COBIT (*Control Objectives for Information Technology*) – É um *framework* voltado para realizar a governança[2] de TI. Essa estrutura foi criada pela ISACA (*Information Systems Audit and Control Association*) e tem como objetivo principal o alinhamento entre os objetivos do negócio e os objetivos da TI, fazendo com que a TI atenda às necessidades de negócio (requisitos de negócios) da maneira mais eficiente possível. Colocamos o COBIT nessa lista porque ele possui um processo específico dedicado à Segurança da Informação (DSS05).

ISF Standard of Good Practice for Information Security (SoGP) – Criado pelo *Information Security Forum* (ISF), o *framework* apresenta tópicos de segurança da informação orientados para os negócios com orientação prática e confiável, ajudando os profissionais de segurança a fornecer boas práticas atualizadas que podem ser integradas a seus processos de negócios, políticas de segurança de informações, gerenciamento de riscos e acordos de conformidade. Uma característica muito interessante deste *framework* é o alinhamento existente entre o *SoGP* e outros *frameworks*: a partir da sua implementação, é possível ficar aderente com 6 outros *frameworks* citados neste artigo (COBIT, NIST-CSF, CIS-CSC 20, PCI-DSS e ISO/IEC 27001 e 27002). Isso porque o *SGoP* tem o maior número de controles e foi desenvolvido por profissionais de segurança que sentiam na própria pele as dificuldades de manter a conformidade a diversas leis e regulamentações, onde cada uma dessas leis/regulamentações se baseavam em um *framework* diferente.

2- Conjunto de processos, costumes, políticas, leis e regulamentos que regulam a maneira como uma empresa é dirigida, administrada ou controlada.

Outros Frameworks importantes

Aqui estão listados outros *frameworks* cujo conteúdo foi elaborado para garantir segurança a temas específicos como, por exemplo, computação na nuvem (*Cloud Computing*). Em geral, são estruturas cujos controles foram elaborados para padronizar a segurança da informação em um determinado setor do mercado ou tecnologia específica.

PCI-DSS *(Payment Card Industry-Data Security Standards)* – Criado pelo *PCI Standards Security Council*, este padrão "foi desenvolvido para incentivar e aprimorar a segurança dos dados do titular do cartão e promover a ampla adoção de medidas de segurança de dados consistentes no mundo todo." [5]. É considerado por muitos profissionais de segurança o framework cujos controles técnicos são descritos de forma bastante detalhada, podendo inclusive serem implementados em empresas de outros segmentos.

Cloud Security Alliance – Cloud Control Matrix* (CSA-CCM) –** Criada em 2008, a CSA surge com o objetivo de definir boas práticas de segurança face a emergente tendência de migração dos ambientes para computação na Nuvem. Opera o mais popular programa de certificação de provedores de segurança em nuvem, o **CSA *Security, Trust & Assurance Registry (STAR). O conjunto de controles definidos em sua matriz não só garantem a proteção dos dados, mas também a conformidade com 46 outros *frameworks*, de acordo com a versão de março de 2019.

***Federal Financial Institutions Examination Council's* (FFIEC) –** O FFIEC é um órgão interinstitucional, formal, encarregado de prescrever princípios únicos, padrões e formulários de relatório para o exame federal de instituições financeiras pelo *Board of Governors* do *Federal Reserve System (FRB)*, a *Federal Deposit Insurance Corporation (FDIC)*, o *National Credit Union Administration (NCUA)*, o *Office of the Comptroller of the Currency (OCC)* e o *Office of Thrift Supervision (OTS)*, além de fazer recomendações para promover a uniformidade na supervisão das instituições financeiras. Seu *framework* é composto por 494 controles, alguns relacionados diretamente ao NIST-CSF.

Financial Service Sector Coordinating Council (FSSCC) – Fundada em 2002 pelo setor financeiro, a FSSCC coordena atividades críticas de infraestrutura e segurança interna no setor de serviços financeiros. Seus 70 membros consistem em associações de comércio financeiro, serviços financeiros e as empresas financeiras mais críticas. Seu *framework* se baseia na gestão perfil de risco destas instituições e utiliza o *framework* do FFIEC como base para conformidade.

HITRUST CSF – É um *framework* desenhado para retomar o ponto em que o HIPAA [6] parou e melhorar a segurança para provedores de serviços de saúde e fornecedores de tecnologia. Combina requisitos de quase todos os regulamentos de conformidade existentes, incluindo o GDPR[3] da UE (União Europeia). Inclui tanto a análise de risco quanto as estruturas de gerenciamento de risco, juntamente com os requisitos operacionais para criar uma estrutura maciça e homogênea que possa ser aplicada a quase todas as organizações e não apenas às da área da saúde. Aqui no Brasil não se fala muito deste *framework*, exceto se a instituição médica possuir negócios com o setor de saúde norte americano.

Análise

Existem 2 pontos interessantes nesses *frameworks:* a possibilidade de separá-los em 3 classificações diferentes, de acordo com o propósito, e estabelecer cruzamentos a partir da sobreposição dos controles que existe entre eles.

Classificação

Quando nos deparamos com essa quantidade imensa de frameworks, uma pergunta pode vir a nossa mente: "qual *framework* devo utilizar e em qual circunstância?" A resposta pode ser a clássica de um consultor ("Depende..."); porém, podemos estabelecer uma classificação que nos permita ter maior direcionamento nessa decisão:

***Frameworks* de Controles**
Exemplo: NIST SP 800-53, CIS CSC Top 20, HITRUST CSF, FFIEC, PCI-DSS, ISO/IEC 27002, CSA-CCM

Este tipo de *framework* é utilizado para se construir um *baseline*, avaliar o estado das capacidades técnicas do ambiente, priorizar a implementação de controles e criar um *roadmap* para a organização.

***Frameworks* de Programa**
Exemplo: ISO/IEC 27.001, NIST CSF, COBIT (DSS05)

São *frameworks* utilizados para construir o sistema de gestão de segurança da informação (SGSI), permitindo planejamento, suporte, documentação, operação, avaliação e performance, e evolução dos processos e controles de segurança estabelecendo, assim, uma relação muito próxima da estratégia de segurança com os objetivos de negócio.

3- GDPR, ou *General Protection Data Regulation* (Regulamento Geral de Proteção de Dados, em tradução livre), é um conjunto de regras de proteção de dados para todas as empresas que operam na União Européia, não importa onde estejam fisicamente. Essas regras visam dar às pessoas mais controle sobre seus dados pessoais e criam condições equitativas para as empresas que passam a adotá-la. Mais informações em [7].

***Frameworks* de Risco**

Exemplo: NIST SP 800-37, FFIEC, FSSCC

Frameworks de risco proporcionam aos times de segurança da informação demonstrar a gestão correta do programa de segurança, pois define as principais etapas do processo para avaliar e gerenciar riscos, estabelece e estrutura o programa de gerenciamento de risco, identifica, mede e quantifica o risco, com o propósito de priorizar atividades de segurança. Com isso, estabelece-se um canal comum de comunicação com os *stakeholders* (partes interessadas) já que utiliza uma visão não-técnica da gestão da segurança na empresa. Não obstante, serve também como ferramenta para apoiar a gestão estratégica das organizações, por permitir visualizar os possíveis riscos inerentes a estratégia e, assim, facilitar a tomada de decisão de quais riscos assumir, quais controlar e quais mitigar (eliminar ou reduzir a um nível aceitável) – este processo é conhecido como estabelecer o **apetite de risco**.

Sobreposições de Frameworks

Outro aspecto interessante nos *frameworks* é a sobreposição entre eles, de modo que "cruzamentos" podem ser construídos para mostrar conformidade com padrões regulatórios diferentes. Assim, pode-se adotar um conjunto de *baseline frameworks* e, com um bom trabalho de pesquisa e análise, avaliar e definir a conformidade com tantos outros *frameworks,* regulamentações e legislações quanto forem necessários. Alguns Exemplos:

- A ISO 27002 define a política de segurança da informação na seção 5; O COBIT o define na seção "Planejar e Organizar" e o PCI DSS define como "Manter uma política de segurança de informações";
- Ao implementar e gerenciar corretamente o NIST CSF, você estabelece conformidade com 153 controles do FFIEC. O item *"ID.AM-1: Physical devices and systems within the organization are inventoried"* do NIST é equivalente ao controle *"D1.G.IT.B.1: An inventory of organizational assets (e.g., hardware, software, data, and systems hosted externally) is maintained"* do FFIEC;
- Na mesma linha, o controle *BA.01.01 – Business Application Register* do ISF – SoGP é equivalente ao controle *"ID.AM-2: Software platforms and applications within the organization are inventoried"* do NIST CSF.

Frameworks em números

A Tabela 2 demonstra, em números, como os *frameworks* foram concebidos pensando em controles, objetivos de controle e atividades, de forma a permitir sua verificação por um terceiro independente (a.k.a. auditoria).

Framework	Categoria de Controles	Objetivos de Controle	Atividades
ISO27001 2013	Cláusulas (5)	Objetivos de controle (35)	Controles (114)
ISO27002 2013	Seções (14)	Objetivos de controle (35)	Controles (114)
NIST CSF (2018)	Funções (5)	Categorias (22)	Subcategorias (98)
ISF - SoGP 2018	Categorias (17)	Áreas (34)	Tópicos (131)
COBIT 2019	Domínios (5)	Processos (40)	Práticas (231)
PCI-DSS v3.2	Objetivos (6)	Requerimentos (12)	Requerimentos (170)
CIS-CSC TOP 20	Categorias (3)	Controles (20)	Sub-Controles (171)
CSA-CCM	Domínios (16)	Domínios (16)	Controles (116)
FFIEC	Domínios (5)	Fatores de Avaliação(15)	Controles (494)[A]
NIST SP 800-53	Família (26)[B]	Controles(282)	Melhorias (até 982)[C]
NIST SP 800-37 [D]	-	-	Tarefas (47)
FSSCC [E]	Domínios (5)	Fatores de Avaliação(15)	Controles (494)[A]
HITRUST CSF v9.2	Categorias (14)	Objetivos de controle (49)	Controles (156)

[A] São 30 Controles cujas instruções são divididas em 05 níveis de maturidade (Baseline, Evolving, Intermediate, Advanced e Innovative)
[B] São 18 famílias de controles de Segurança da Informação complementados com 08 famílias de controles específicos para privacidade dos dados pessoais.
[C] Por tratar-se de um framework tailored (adaptável, em trad. livre), cada controle pode ser acrescido de melhorias, de acordo com a necessidade de implementação e avaliacão prévia de risco.
[D] Este framework é utilizado para se implementar um processo de gestão de risco e, portanto, não tem estrutura de objetivos de controle e controles, apenas atividades divididas em etapas
[E] Esse framework utiliza o framework do FFIEC como base.

Tabela 2 - Análise dos frameworks em números

Conclusão

Gerir a Segurança da Informação não é uma tarefa fácil. Se por um lado temos uma série de documentações, procedimentos e processos para contribuir para a segurança das informações das organizações, do outro temos a operação que exige cada vez mais agilidade e *time to market* da tecnologia, para não ser engolido pela concorrência.

Neste cenário, é fundamental que o gestor de segurança atue de forma pragmática, em parceria com todas as áreas da Tecnologia, mas principalmente com seu cliente interno, mostrando os riscos e as formas de se corrigir com muita clareza e transparência. Em casos que se necessite uma tomada de decisão mais estratégica, deve-se sempre tomar os cuidados necessários para que os riscos sejam mostrados de forma impessoal e na linguagem do negócio – não adianta falar que a "versão do website que vai para produção está vulnerável a ataques de *cross site scripting* e que isso pode permitir que atacantes sejam capazes de inserir scripts maliciosos em páginas que seriam confiáveis e usá-los para sequestrar o acesso de usuários e administradores" porque você não será ouvido.

Os *frameworks* estão aí para ajudar-nos a definir claramente o que se pretende controlar, além de demonstrar os prejuízos ao negócio caso haja uma materialização do risco; usá-los de forma racional, madura e consistente nos permitirá reduzir a exposição das organizações. Não é uma tarefa fácil mas, com muita disciplina, dedicação e conhecimento, pode-se chegar lá!

Julio Moreira

Formado em Tecnologia de Processamento de Dados pela Universidade Mackenzie, possui duas pós-graduações: em Análise de Sistemas pela Universidade Mackenzie e em Gestão Empresarial pela FEI. Tem mais de 20 anos de experiência em Tecnologia da Informação, sendo os últimos 19 anos dedicados à Segurança da Informação dentro de grandes empresas dos setores financeiro, seguros, consultoria, saúde e varejo. Ocupou posições de liderança como CISO (Chief Information Security Officer) nos últimos 15 anos, com grande experiência em iniciativas estratégicas de Segurança da Informação: desenvolvimento e implementação de estratégia de segurança da informação, gerenciamento de riscos e conformidade, gerenciamento e acesso a identidades, proteção de endpoint (proteção de estações de trabalho / laptops / dispositivos móveis), arquitetura de segurança corporativa , proteção de marca de mídia digital, gerenciamento de vulnerabilidades, entre outros. Trabalha atualmente como Consultor Sênior de Segurança da Informação e Privacidade na IBM. É apaixonado pela família, tecnologia, artes marciais e música (esta última como ouvinte apenas). Possui as certificações CISSP, CISM, EXIN PDPE (Privacy Essentials), ISO 27001 Lead Auditor e PCI-QSA.

Referências:

[1] https://databreachcalculator.mybluemix.net/ (necessário realizar cadastro), acessado em 20/08/2019

[2] https://originit.co.nz/the-strongroom/five-most-common-security-frameworks-explained/, acessado em 20/08/2019

[3] http://clubedosgeeks.com.br/sem-categoria/cifra-de-cesar-criptografia-monoalfabetica, acessado em 23/09/2019

[4] http://dsic.planalto.gov.br/legislacao/2_Guia_SICI.pdf, acessado em 23/09/2019

[5] https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2-1_PT-BR.pdf?agreement=true&time=1569190299454, acessado em 20/08/2019

[6] https://www.medicinenet.com/script/main/art.asp?articlekey=31785, acessado em 23/09/2019

[7] https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules_en, acessado em 23/09/2019

[8] https://www.itgovernanceusa.com/blog/top-4-cybersecurity-frameworks, acessado em 20/08/2019

[9] http://www.bmc.com/content/dam/bmc/collateral/third-party/Ponemon%2bReport.pdf, acessado em 20/08/2019

[10] https://www.simplilearn.com/top-cybersecurity-trends-for-2019-article, acessado em 20/08/2019

[11] https://www.techrepublic.com/article/how-to-choose-the-right-cybersecurity-framework/, acessado em 20/08/2019

[12] https://www.welivesecurity.com/2017/01/31/cybersecurity-5-basic-lessons-everyone/, acessado em 20/08/2019

[13] https://www.itgovernance.co.uk/what-is-cybersecurity, acessado em 20/08/2019

[14] https://logicalisinsights.com/2019/06/04/what-is-a-common-security-framework-updated-for-2019/, acessado em 20/08/2019

[15] https://searchsecurity.techtarget.com/tip/IT-security-frameworks-and-standards-Choosing-the-right-one?vgnextfmt=print, acessado em 23/08/2019

[16] https://www.sclogic.com/blog-what-is-a-nist-certification/, acessado em 23/08/2019

[17] https://digitalguardian.com/blog/what-fisma-compliance-fisma-definition-requirements-penalties-and-more, acessado em 23/08/2019

[18] https://www.compliancepoint.com/fisma-and-nist-standards, acessado em 23/09/2029

[19] https://digitalguardian.com/blog/what-nist-compliance, acessado em 23/08/2019

CURIOSIDADES

O que é MSR?

por Gabriel Negreira Barbosa

Este é um artigo introdutório, fortemente baseado no Intel SDM (Software Developer's Manual) [1] e seu conteúdo se aplica a CPUs Intel. Para maiores informações acerca de qualquer tópico abordado, favor consultar tal referência.

MSR (Model Specific Register ou Machine Specific Register) é um tipo de registrador provido pelo processador para realizar diversas funções, como configurar o comportamento de instruções, atualizar o microcódigo, monitorar performance, habilitar e desabilitar funcionalidades do processador, etc. Os MSRs são geralmente utilizados pela BIOS/UEFI, sistema operacional e hypervisor.

Há diversos MSRs. Cada um possui uma função específica e é unicamente identificado por nome e endereço. Vale lembrar que o endereço de um MSR não tem relação com a memória, é somente um código de identificação que é geralmente expresso em hexadecimal.

Independentemente se a CPU for de 32 ou 64 bits, cada MSR possui 64 bits que podem ser organizados em campos de 1 ou mais bits. Cada campo possui uma função específica. MSRs e campos que não sofrerão alterações em CPUs futuras são conhecidos como "arquiteturais". Por motivos históricos, a partir do Pentium 4 o prefixo "IA32_" foi adicionado ao nome dos MSRs arquiteturais.

Os MSRs podem ser lidos ou escritos. Para isso, há duas instruções dedicadas a acessá-los: RDMSR (para leitura) e WRMSR (para escrita). Dentre suas características mais relevantes à maioria dos(as) leitores(as) ressalta-se que, em modo protegido, se essas instruções não forem executadas em Ring 0 uma exceção será gerada (General Protection). Para mais informações, includindo mudanças de comportamento relacionadas à virtualização, recomenda-se consultar [1].

O Volume 4 do Intel SDM [1] possui uma lista de MSRs com informações a respeito de cada um, como, por exemplo, nome, endereço e descrição. A fim de ilustrar o conceito de MSR, a Figura 1 contém a descrição do MSR_PPERF (endereço 0x64E) dos processadores Intel Core de 6ª, 7ª e 8ª gerações.

Register Address		Register Name / Bit Fields	Scope	Bit Description
Hex	Dec			
64EH	1614	MSR_PPERF	Thread	Productive Performance Count (R/O)
		63:0		Hardware's view of workload scalability. See Section 14.4.5.1.

Figura 1 – MSR_PPERF (endereço 0x64E) dos processadores Intel Core de 6ª, 7ª e 8ª gerações. Figura adaptada da Tabela 2-39 (Volume 4) de [1].

Os MSRs possuem escopos diverso (exemplos: thread, core e package). Como observado na Figura 1, o MSR_PPERF possui o escopo de thread, o que significa que cada thread da CPU mantém seu próprio MSR_PPERF de forma independente. Esse escopo específico (thread) é similar ao dos registradores de uso geral (RAX, RBX, etc) onde, com relação ao acesso a tais registradores, diferentes threads da CPU que estejam em execução podem rodar programas diferentes sem que um afete o outro.

Para finalizar, ressalta-se que MSRs e seus campos possuem diferentes restrições de acesso. Conforme observado na Figura 1, o MSR_PPERF possui somente 1 campo de 64 bits que é somente leitura (R/O). Em geral, violações de acesso geram uma exceção (General Protection).

Referências

[1] Intel® 64 and IA-32 architectures software developer's manual combined volumes: 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D, and 4. Disponível em https://software.intel.com/sites/default/files/managed/39/c5/325462-sdm-vol-1-2abcd-3abcd.pdf. Acessado em 26/09/2019.

Gabriel Negreira Barbosa

Trabalha como principal security researcher no time STORM (STrategic Offensive Research & Mitigations) da Intel. Anteriormente, trabalhou como engenheiro de segurança de software 2 na Microsoft e como pesquisador de segurança líder na Qualys. Recebeu o título de bacharel em ciência da computação pela PUC-SP; e de mestre pelo ITA, onde participou de projetos de segurança para o governo brasileiro e a Microsoft Brasil. Já apresentou trabalhos em algumas conferências, como H2HC, SACICON, Troopers, Black Hat USA e BSides (PDX e DFW).

<Tempest, protegendo negócios no mundo digital.>

A Tempest atua há quase 20 anos no mercado de cibersegurança, monitorando, pesquisando e criando novas soluções de proteção digital.

Conheça mais sobre a maior empresa especializada em cibersegurança do Brasil em números:

+ de **280** colaboradores em Recife, São Paulo e Londres
+ de **250** clientes atendidos no Brasil, Estados Unidos, América Latina e Europa
+ de **400** projetos de consultoria realizados apenas em 2018
+ de **24 mil** horas de consultoria realizadas em 2018
+ de **35** projetos de integração realizados em 2018
+ de 1 **milhão** de anomalias reportadas aos clientes em 2018

TEMPEST
Protegendo negócios no mundo digital.

Acesse **www.tempest.com.br**

ENGENHARIA REVERSA DE SOFTWARE

por Fernando Mercês

Manual Unpacking 101 - Parte 2

O artigo publicado na edição anterior desta revista [5] foi uma introdução ao conceito de MUP (*Manual Unpacking*), onde estudamos o funcionamento do *packer* MPRESS [1]. Nele vimos como as seções de um binário PE alvo são modificadas pelo packer e chegamos ao OEP (*Original Entrypoint*) utilizando o recurso de *breakpoints de hardware* através do x64dbg [2]. Agora, vamos seguir com o processo de *unpacking* para finalmente obter um binário *unpackeado*[1] *em disco*.

⚠ ***É necessário ler o artigo da edição anterior [5], caso ainda não o tenha feito, pois utilizaremos o mesmo crackme alvo, continuando da parte em que encontramos o OEP.***

O processo de dump

Após fazer com que o *debugger* pare no OEP, chegamos na tela indicada na Figura 1.

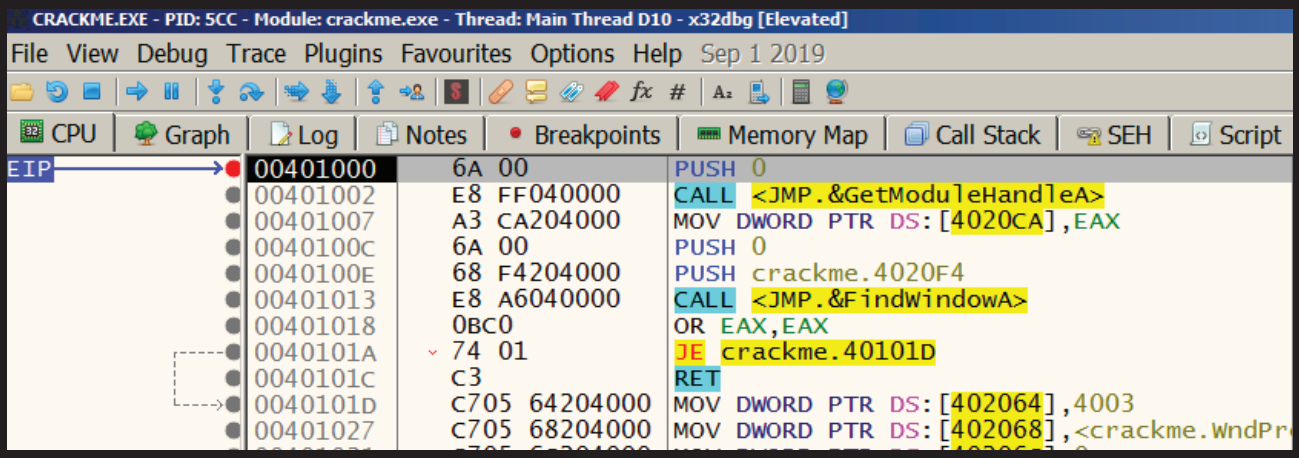


Figura 1 - *Debugger* parado no OEP (0x401000)

1 - Perdão por esse e outros neologismos usados ao longo do artigo, mas todo mundo fala assim.

Neste momento o processo está parado e o endereço do OEP encontra-se no registrador EIP, ou seja, será a próxima instrução a ser executada. Neste momento podemos tentar *dumpar*[2] o processo. Para isto, o x64dbg conta com um *plugin* chamado Scylla [3]. O pacote do x64dbg já inclui o Scylla e, para abri-lo, você pode escolher entre pressionar Ctrl+i, clicar em "Plugins -> Scylla", ou clicar no botão com um "S" vermelho e fundo preto na barra de ferramentas. A janela indicada na Figura 2 se abrirá.

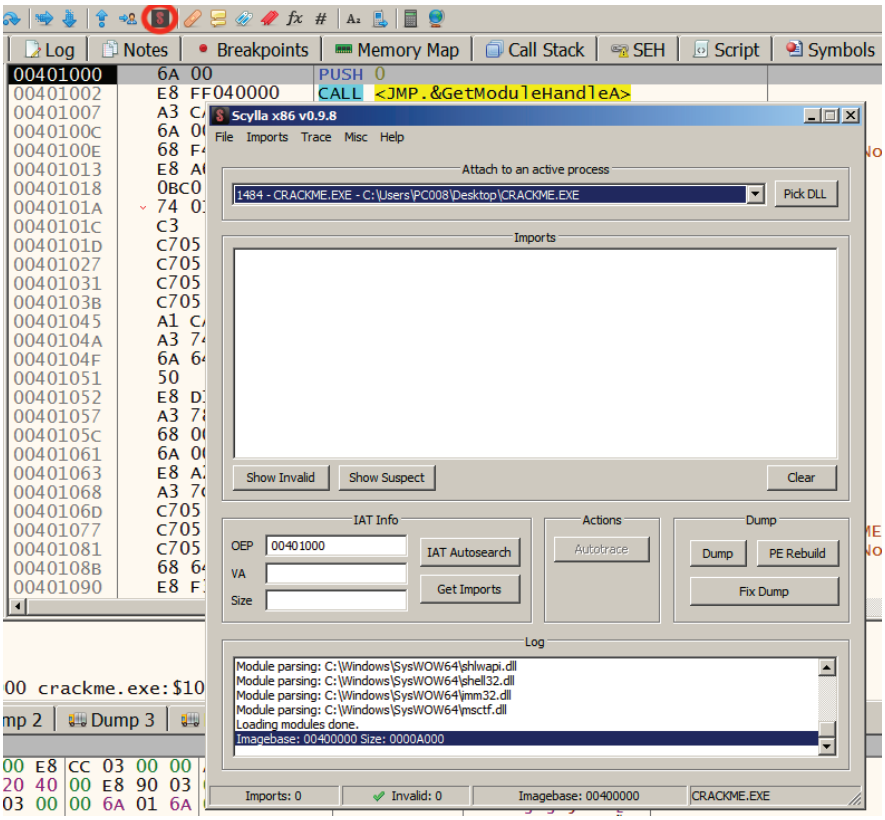


Figura 2 - Janela do Scylla, acionada pelo botão envolto em vermelho

O próximo passo seria clicar no botão "Dump", que vai sugerir salvar um novo arquivo com o nome *CRACKME_dump.exe*. No entanto, se fizermos isso, o arquivo salvo apresentará o erro indicado na Figura 3 caso seja executado.

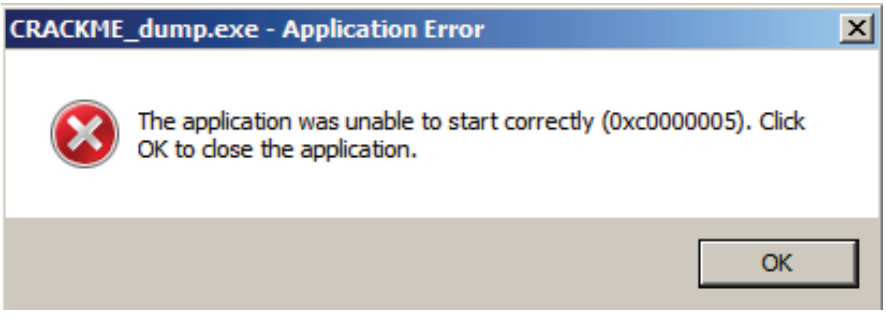


Figura 3 - Erro ao tentar executar o arquivo *dumpado* a partir do OEP

2 - Copiar a imagem do processo para um arquivo em disco.

IAT (Import Address Table)

A IAT é uma estrutura definida na especificação do PE [6] que, em algum momento após o binário ser executado (normalmente em *load-time*), endereços de funções importadas de DLLs são escritos nela. Ou seja, esses endereços são normalmente escritos pelo *loader* do Windows quando o binário é carreado. A fim de estudar tal estrutura, vamos utilizar o recurso "IAT Autosearch" do Scylla conforme mostra a Figura 4.

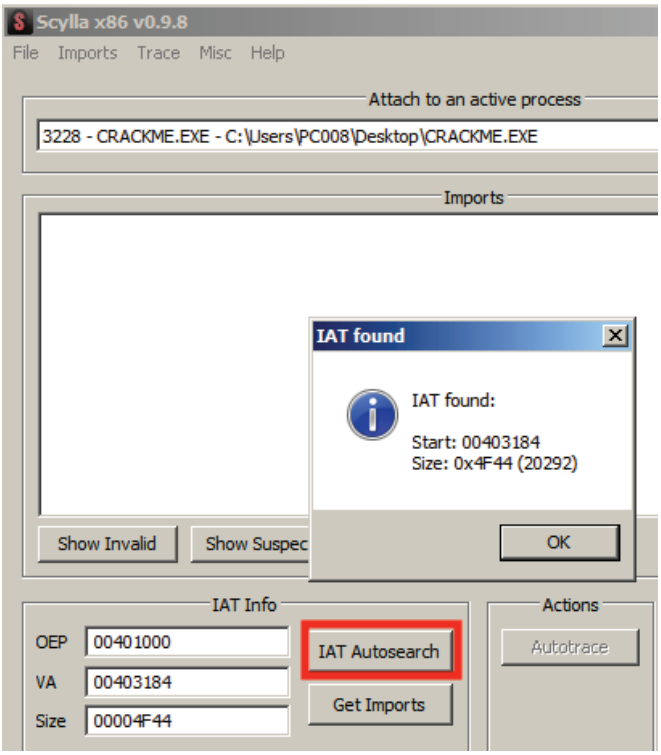


Figura 4 – Recurso "IAT Autosearch" do Scylla

Vamos abrir o endereço retornado na área de *dump* do x64dbg (nesta área podemos inspecionar e até modificar os *bytes* do programa). Para isso, podemos usar a barra de comandos, como indicado na Figura 5.

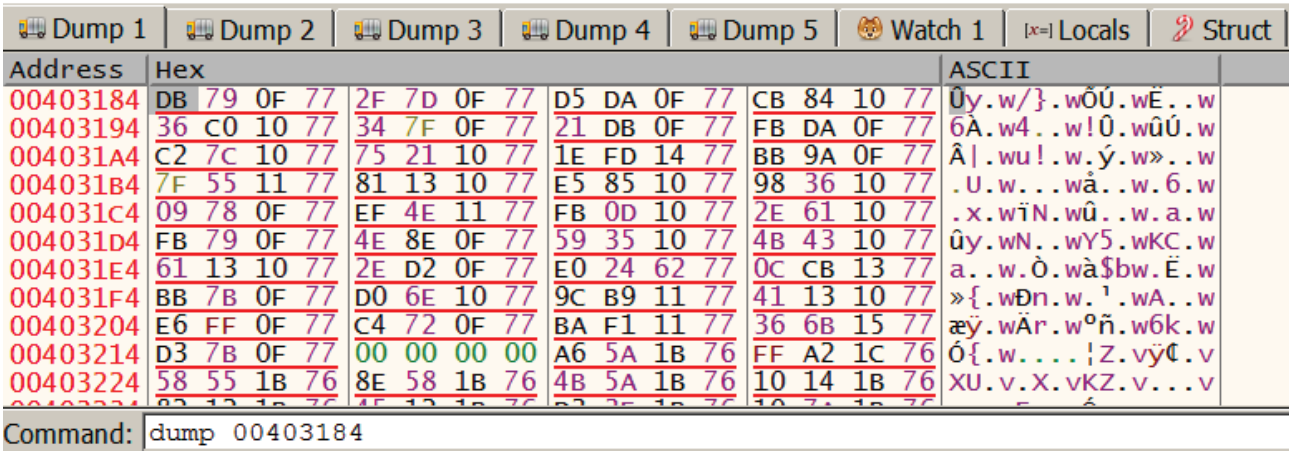


Figura 5 – Endereço retornado pelo Scylla sendo visualizado na área de *dump*

Conforme observado na Figura 5 (e conforme descrito na especificação do PE [6]), pode-se perceber que, nesse momento, a estrutura IAT pode ser entendida como uma lista de endereços de memória 32-bit (pois o binário que está sendo debuggado é um PE32) – ou seja, uma estrutura de tabela de endereços como o acrônimo IAT sugere. O primeiro endereço é 0x770f79db, seguido de 0x770f7d2f e assim sucessivamente.

No entanto, esta visualização em *bytes* não nos diz muita coisa. Felizmente, o x64dbg possui um recurso de visualizar endereços, onde ele já os resolve. Para isso, clique com o botão direito em qualquer área do dump e selecione "Address", como mostra a Figura 6.

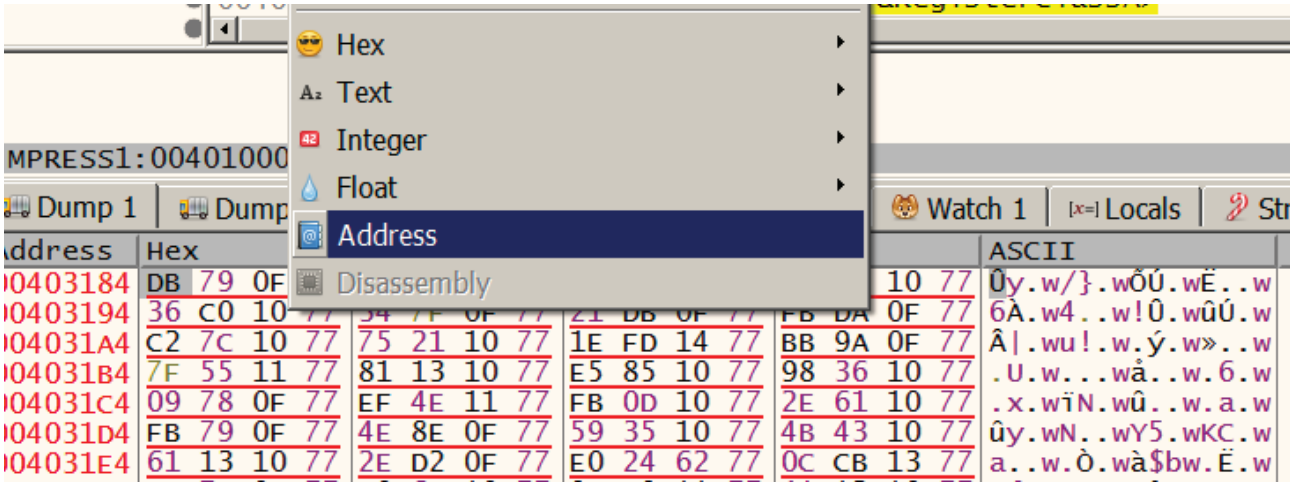


Figura 6 - Mudando a forma de visualização para "Address" no dump

A maneira com a qual os dados são exibidos mudará, e assim poderemos verificar que, conforme discutido anteriormente, os endereços da estrutura IAT dizem respeito a funções importadas de DLLs, como mostra a Figura 7.

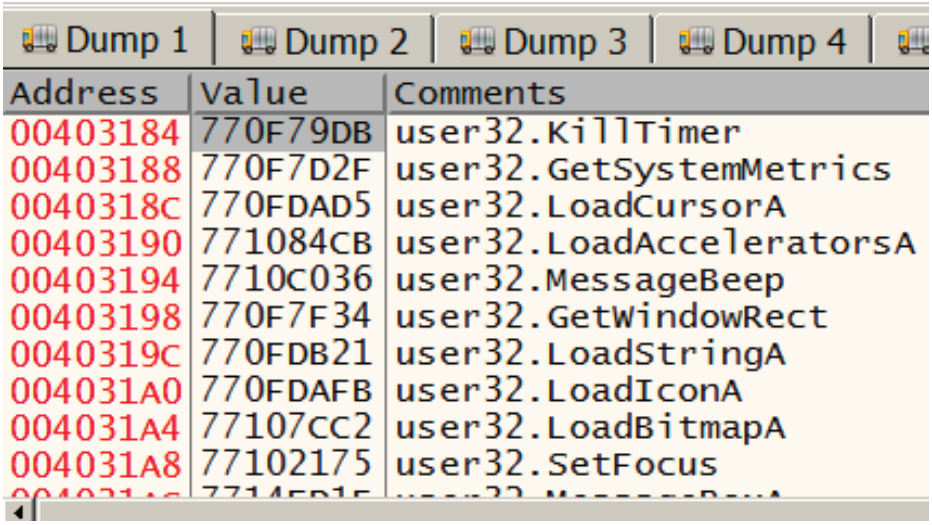


Figura 7 – Endereços resolvidos através do x64dbg

Há outras estruturas (além da IAT) utilizadas pelo PE que são relacionadas à importação de funções. Por exemplo, a Import Directory Table, que possui uma entrada para cada DLL referenciada pela imagem do binário; e cada uma dessas entradas possui diversos campos, incluindo uma referência à IAT. [6] O artigo que será publicado na 17ª edição dessa revista abordará esse assunto em mais detalhes.

Reconstrução de imports

A questão da IAT (e de outras estruturas que serão discutidas no artigo que será publicado na 17ª edição dessa revista) para o processo de *unpacking* é que, como o binário inicialmente executado foi o packeado, tais estruturas estão relacionados a ele, e não ao binário criado anteriormente a partir do *dump* (CRACKME_dump.exe).

Portanto, antes de obter um binário funcional quando se encontra o OEP, é preciso arrumar tais estruturas. É aí que entram ferramentas como o Scylla. Ainda parado no OEP, abra novamente o Scylla, clique em "IAT Autosearch", confirme com "Yes" que quer utilizar o "IAT Search Advanced result" e depois clique em "Get Imports". A tela deve ficar como a da Figura 8.

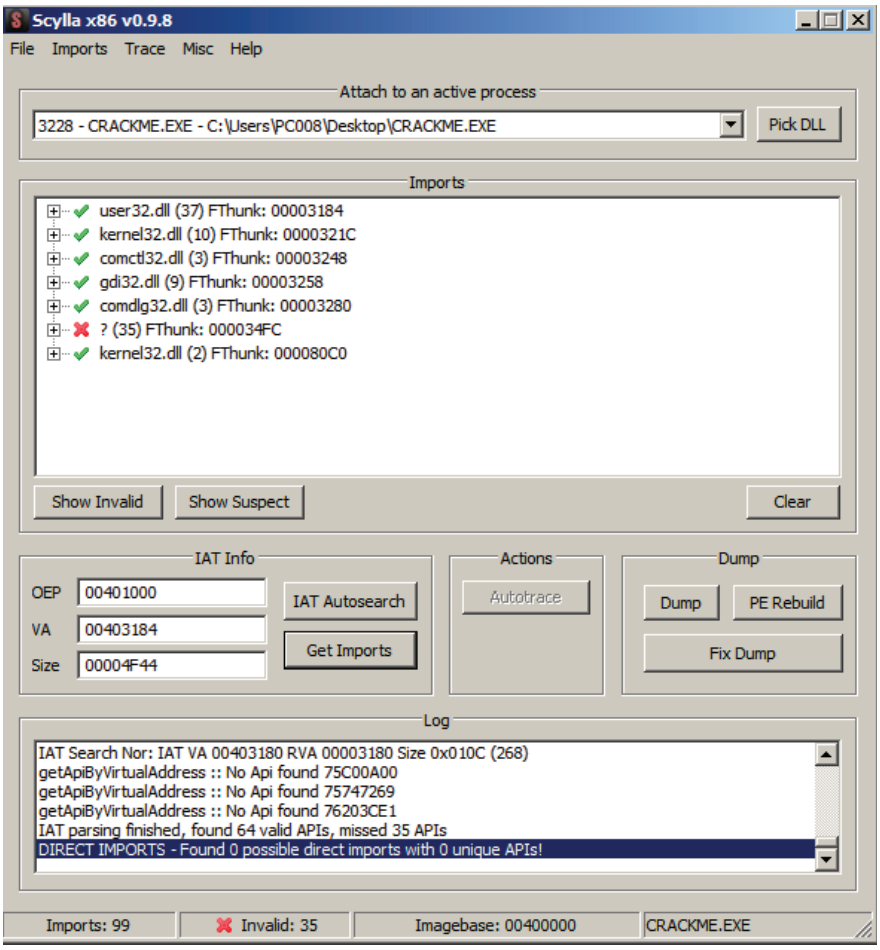


Figura 8 - Recurso "Get Imports" do Scylla

Este processo nem sempre é 100% aferido. No caso da Figura 8, aparentemente, 35 entradas de funções importadas de uma biblioteca (DLL) desconhecida não foram corretamente recuperadas. Não se preocupe, isso acontece com alguma frequência, mas em geral sua remoção não afeta o resultado final. Para removê-las, basta clicar com o botão direito na entrada errônea e escolher "Delete tree node". Após este processo, é possível verificar o resultado da Figura 9. Aí é só clicar em "Fix Dump" e escolher o binário *dumpado* anteriormente (CRACKME_dump.exe).

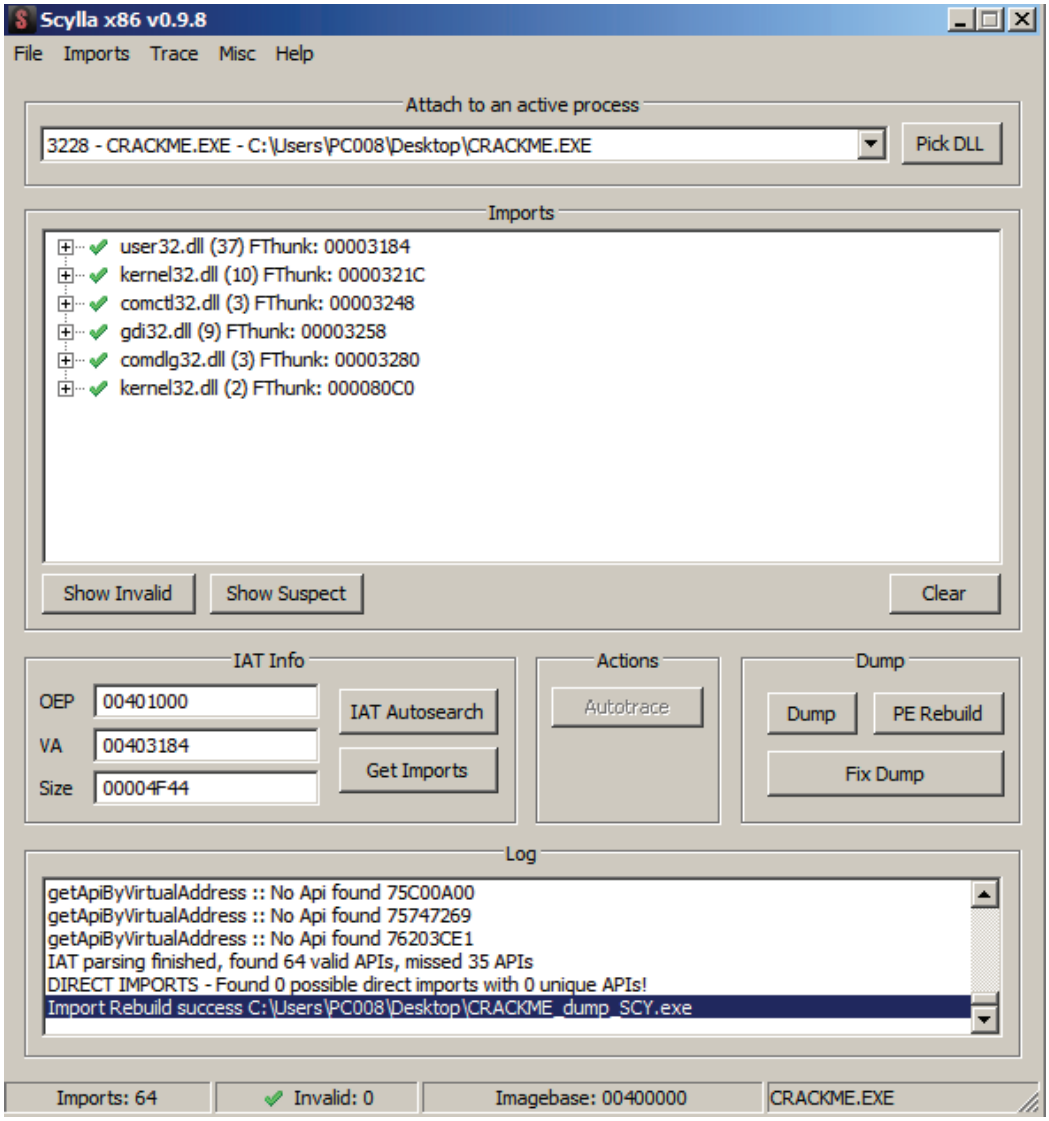


Figura 9 - Janela do Scylla após a deleção da árvore errônea e a correção do *dump* (Fix Dump)

Um novo binário CRACKME_dump_SCY.exe foi criado. Este é o binário anteriormente *dumpado*, mas agora com as funçõs importadas devidamente arrumadas, ou seja, é finalmente o binário *unpackeado* que queríamos obter! Você pode verificar que ele executa normalmente dando um duplo-clique nele e também verificar que seu EP é o OEP (Figura 10).

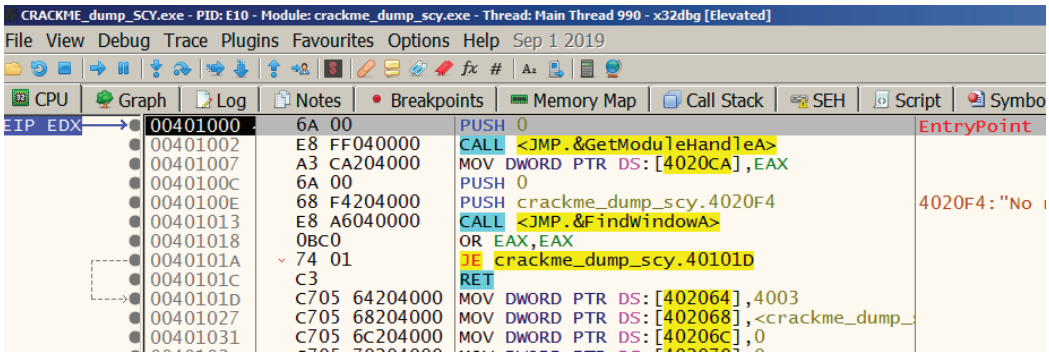


Figura 10 - Binário *unpackeado* após o processo de *unpacking*

Curiosamente, o DIE ainda "pensa" que o binário está *packeado* com o MPRESS (Figura 11).

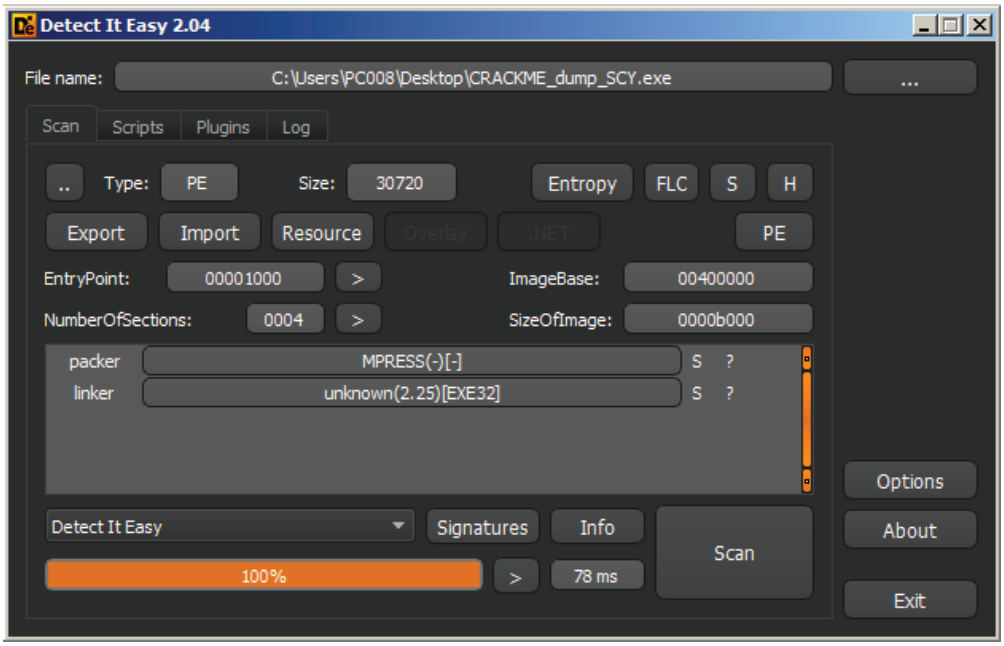


Figura 11 - DIE detectando MPRESS erroneamente

A verdade é que a maneira como o DIE detecta cada *packer* é diferente. No caso do MPRESS, ele busca uma *string* que o MPRESS adiciona no binário. Como essa *string* não foi removida pelo processo de *unpacking* manual que realizamos, o DIE continua acreditando que o binário ainda está *packeado*. Você pode verificar a lógica por trás dessa detecção clicando no pequeno "S" (de "Signature") ao lado de onde o DIE mostra "MPRESS(-)[-]" (Figura 12).

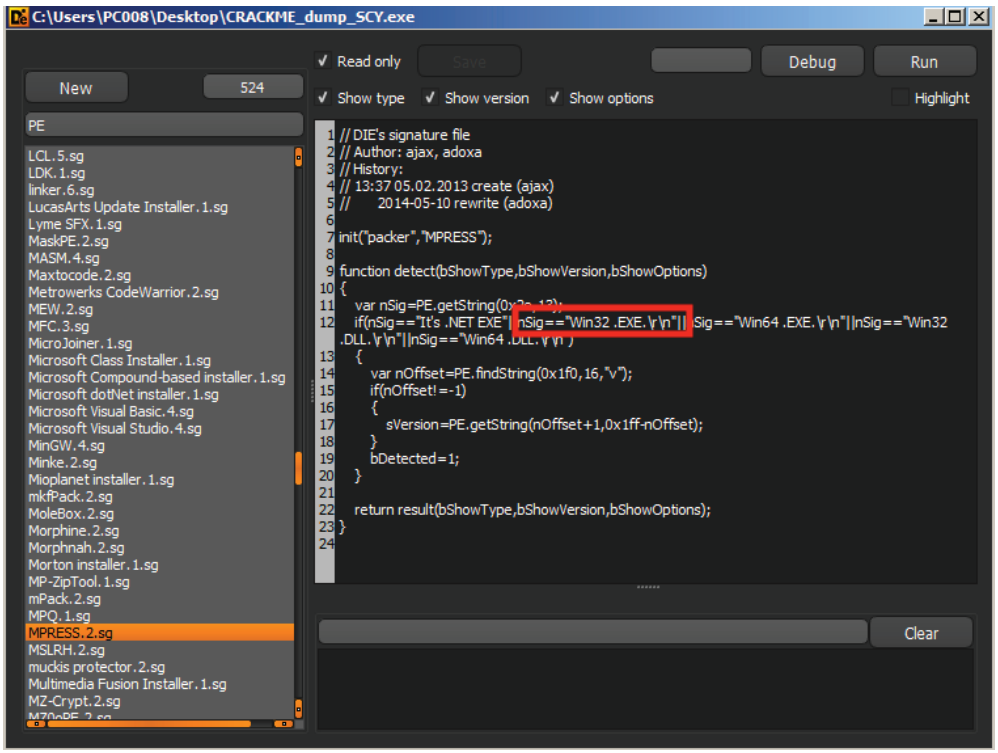


Figura 12 - Lógica do DIE para detectar o MPRESS

O processo de unpacking mostrado nesse artigo gera um binário PE unpackeado funcional, mas não idêntico ao original: essa string remanescente é um exemplo de diferença.

Unindo o artigo anterior a este, estudamos todo o processo de *unpacking* manual de binários *packeados* com o MPRESS. Packers diferentes possuem características diferentes, logo o processo descrito neste artigo pode necessitar de alterações para outros *packers*. Até a próxima edição! o/

Referências

[1] http://www.matcode.com/mpress.htm

[2] https://x64dbg.com

[3] https://github.com/NtQuery/Scylla

[4] https://github.com/horsicq/DIE-engine/releases/download/2.00/die_win32_portable_2.00.zip

[5] https://www.h2hc.com.br/revista/RevistaH2HC_13.pdf

[6] https://docs.microsoft.com/en-us/windows/win32/debug/pe-format

Fernando Mercês

Pesquisador de Ameaças na Trend Micro, onde atua como investigador de ciber crime utilizando engenharia reversa e técnicas de inteligência de ameaças no time de Pesquisa de Ameaças Futuras (FTR). Criador de várias ferramentas livres na área de segurança, é constante palestrante nos principais eventos de segurança no Brasil e no exterior. É também professor e fundador da comunidade Mente Binária, comprometida com entrega de conteúdo gratuito para estudantes de segurança da informação.


H2HC
HACKERS TO HACKERS CONFERENCE


H2HC
HACKERS TO HACKERS CONFERENCE