Cibersegurança: identificação de keystroke por dispositivo *Rubber Ducky*

Anderson Vagner Kobs[1], Sylvio André Garcia Vieira[1]

[1]Sistemas de Informação – Universidade Franciscana (UFN)
97.010-032 – Santa Maria – RS – Brasil

inf.avagner@gmail.com, sylvio@ufn.edu.br

Abstract. *With the personal and corporate data digitalization, information security is an increasingly common issue in organizations. Against this approach, this project brings a solution for forensic device identification, called Rubber Ducky. Known for executing automated scripts when connecting to USB ports, this application, based on Python language, will prevent the completion of these scripts, protecting the user from an eventual breach of digital document confidentiality.*

Resumo. *Com a digitalização de dados pessoais e corporativos a segurança da informação é pauta cada vez mais presente nas organizações. De encontro a esta abordagem este projeto traz uma solução para a identificação de dispositivo forense, denominado Rubber Ducky. Conhecido por executar scripts automatizados no momento da sua conexão em portas USB, esta aplicação, baseada em linguagem Python, impede a conclusão destes scripts, protegendo o usuário de uma eventual violação de confidencialidade documental digital.*

1. Introdução

A presença da tecnologia em tarefas rotineiras está cada vez mais em evidência. Equipamentos, antes compostos por uma série de componentes interligados, estão se tornando portáteis e de fácil transporte para qualquer lugar, carregando em seu interior um grande volume de informações. Muitas vezes estas informações possuem em seu teor uma confidencialidade muito elevada, pois podem representar o futuro de uma inovação, grandes prejuízos materiais, ou ser motivação de estelionatos, por exemplo.

A necessidade de uma educação adequada para um uso consciente dos sistemas digitais é de extrema importância. Tal fato é mencionado por [Carneiro 2013], evidenciando a importância da correta instrução do fator humano sobre segurança digital, pois a decisão de ação no momento mais oportuno, da ferramenta mais adequada à situação, e da crucialidade da segurança dos dados ainda são de decisão do gerente dos sistemas. Falhas minuciosas em algumas destas etapas podem representar o momento perfeito para captura de dados por pessoas não autorizadas, interessadas em obter o máximo de vantagem enquanto detentoras destas informações.

A sofisticação da tecnologia apresenta uma certa dualidade. Enquanto o tráfego de informações consiste no envio e recebimento de dados criptografados, grandes empresas do ramo tecnológico empregam expressivo esforço em técnicas que mantenham seguras todo o volume informacional. Esta segurança deve ser pertinente tanto no momento em que transitam de um lado para outro, bem como o seu acesso restrito somente a pessoas autorizadas depois de armazenadas.

Mesmo com toda esta preocupação, com igual intensidade, pessoas mal-intencionadas buscam aprimorar suas estratégias na prática de apropriação indevida de

informação alheia com alguma vantagem que esta pode proporcionar. Os ataques cibernéticos podem ser disparados com as mais diversas finalidades, como captura de senhas, fraudes bancárias, obtenção de arquivos pessoais, entre outros.

Considerando esta contínua sofisticação em invasão de dispositivos digitais e a consequente oferta de ferramentas e treinamentos tem-se como objetivo geral deste trabalho apresentar possíveis consequências desencadeadas pelo uso de equipamentos e sistemas *hacking*, como o dispositivo *Rubber Ducky* e ao mesmo tempo uma forma de proteção contra possíveis ações de obtenção de dados ilícitos a partir de um sistema computacional. Deste parecer generalista foram abordados objetivos específicos como:

- Estudar sobre o dispositivo *Rubber Ducky* e suas formas de captura de informações;
- Identificar cenários vulneráveis a ataques;
- Utilizar as premissas *Ethical Hacking* como guia de projetos;
- Desenvolver uma solução capaz de impedir a ação de um *Rubber Ducky*;
- Coletar resultados após tentativas/bloqueios de invasão pelo dispositivo.

2. Referencial Teórico

Este item terá como objetivo fundamentar em pesquisas bibliográficas um amparo explicativo, enfatizado com definições importantes sobre invasão de sistemas computacionais, abrangente com a temática abordada neste trabalho, bem como uma abordagem conceitual sobre as ferramentas alvo de estudo durante a evolução do projeto.

2.1. Segurança da Informação

Inevitavelmente todo ambiente institucional necessita estar interligado ao mundo externo para expandir as suas atividades, por meio de alguma forma de comunicação. Essa informação que circula entre as partes interessadas possivelmente estará sujeita a alguma interceptação. Assim sua gestão deve estar promovida de maneira estratégica para garantir o funcionamento e crescimento de toda a composição organizacional.

No que ser refere a sua segurança, [Konzen 2013] destaca que é imprescindível a identificação de vulnerabilidades que ofereçam algum tipo de risco a qualquer ativo de informação dentro de uma corporação. O armazenamento ainda ocorre das mais variadas formas ou meios, em dispositivos locais ou em rede, ou ainda em meio digital ou impresso. Apesar de ser armazenada em equipamentos tecnológicos, o autor enfatiza ainda que a segurança da informação depende ao mesmo tempo de uma gestão humana, capaz de manter o ambiente controlado, com políticas de restrição adequada. Deste modo será assegurado que o usuário destinado ao controle de determinada informação conduza o sistema de maneira eficiente e segura. Contemplando este conceito de gestão da Segurança da Informação, [Silva 2009] define que a seguridade da informação estará efetiva quando executado o sistema de gestão apresentado na Figura 1.

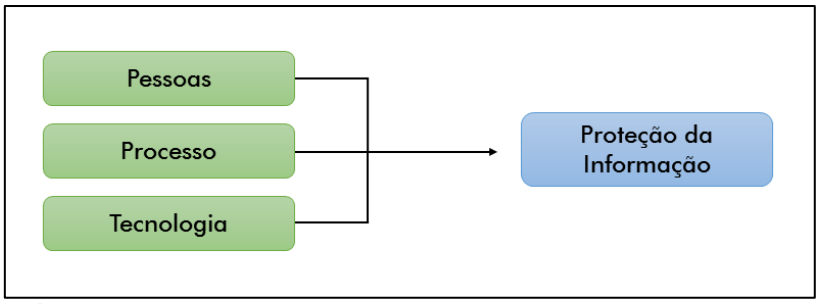

Figura 1. Os três principais recursos na gestão da informação [Adaptado de Silva 2009].

Considerando a abordagem do conceito de Segurança da Informação sob as suas mais variadas perspectivas, entende-se que o armazenamento da informação é destinado a alguma finalidade em algum momento durante sua vida útil. Para garantir o cumprimento destes requisitos, a autora enfatiza que a segurança estará assegurada pela organização de três pilares: Disponibilidade, Integridade e Confidencialidade, destacados na Figura 2.

Figura 2. Tripé da Segurança da Informação [Adaptado de Silva 2009].

Da mesma forma [Konzen 2013], defende estes três itens como essenciais na promoção da Segurança da Informação. Algumas outras bibliografias, assim como ele, acrescentam ainda mais dois princípios: Autenticidade e Legalidade. Conceitualmente define-as como:

- **Confidencialidade**: representa a restrição de acesso apenas aos usuários autorizados. Implica a proteção das informações contra alterações, cópias, reprodução ou alguma outra forma de apropriação de conteúdos restritos.
- **Integridade**: no momento de acesso da informação seu conteúdo não pode estar corrompido, garantindo o arquivo de forma legítima e consistente ao longo do seu ciclo de vida.
- **Disponibilidade**: em algum momento uma determinada informação pode ser solicitada sob demanda por algum usuário ou processo. Estabelece assim a acessibilidade momentânea a aquela requisição efetuada ao evento requisitante.
- **Autenticidade**: no momento da confirmação da Integridade, é realizada a confirmação de que toda a informação obtida e proveniente de pessoas autorizadas, agregando credibilidade aos arquivos.

- **Legalidade**: toda informação deve estar sujeita a regulamentação estabelecida pela lei. Todos os princípios éticos devem ser revistos e assegurados no destino das informações a sua finalidade.

Seguindo estes conceitos a qualidade da informação estará assegurada, garantindo-se sigilo, acesso e solidez a quem interessar, no momento oportuno. Qualquer violação de um destes itens pode resultar em perdas variadas a respeito das informações solicitadas, como por exemplo, vazamento de dados de caráter sigiloso da própria instituição proprietária ou de seus clientes.

Quanto a classificação das ameaças referente a Segurança da Informação [Marciano 2006] tipifica alguns conceitos importantes:

- **Ameaças**: segundo o autor, embasa-se em eventos indesejáveis aos equipamentos de informação cujas finalidades consistem em remover, danificar ou destruir algum recurso ou informação;
- **Vulnerabilidades**: quando um sistema de informação é elaborado podem ocorrer pontos falhos com passividade de exploração por alguma ameaça. Pode ser em pontos físicos no momento da sua instalação, ou ainda algum software com comunicação aberta configurado por um usuário;
- **Ataques**: representam alguma forma de ação muitas vezes meticulosamente planejada normalmente originadas por atuação humana. As tentativas de acesso indevido geralmente consistem em apropriação de informações indevidas, mas nem sempre completam o seu objetivo.

2.2. Pentest (Teste de Penetração)

O risco de ataques a sistemas empresariais cresce a nível mundial na medida que toda a informação se transforma para a plataforma digital. A preocupação frente a estas ameaças tem motivado analistas e gerentes de segurança da informação a buscar soluções de precaução antes que um hacker promova algum tipo de dano por uma invasão.

A busca por soluções que promovam uma segurança eficiente frente a estas ameaças motivou especialistas da área de Segurança da Informação na promoção de técnicas cujo objetivo esteja sempre um passo à frente da ação dos hackers. Esta rotina de testes prévios na identificação de vulnerabilidades é conhecida atualmente como Teste de Penetração, ou da abreviação em inglês "*Pentest*".

Segundo [Rocha et al. 2016] um Teste de Penetração consiste em um padrão de técnicas de alto nível de segurança, com finalidade de encontrar potenciais vulnerabilidades em uma infraestrutura de tecnologia, seja um sistema, servidor ou uma estrutura de rede em geral. Este tipo de ação também pode ser pontual, direcionando-a a apenas itens cruciais, como por exemplo dados corporativos com registros estritamente sigilosos. Para [Pavan and Guardia 2015] mais do que uma metodologia, a eficiência dos testes fornece um relatório com todas as informações minimizando o impacto de potenciais ataques.

2.2.1. Classificação do *Pentest*

Em [Rocha et al. 2016] é apresentada uma classificação quanto aos tipos de *Pentest* que podem ser efetuados. A sua classificação é baseada no tipo de complexidade e informação disponível no momento do procedimento. A quantidade de informações no momento do ataque pode fazer muita diferença. Podem ser:

- ***Blind***: neste procedimento o auditor não tem nenhuma informação sobre o alvo, mas o alvo sabe que será atacado e os procedimentos que serão executados;
- ***Double Blind***: neste processo ninguém tem informações, o auditor não conhece nada sobre o alvo, nem o alvo não sabe que será vítima de ataque, nem dos procedimentos que serão executados;
- ***Gray Box***: o auditor tem um pouco de informações sobre o alvo, o alvo está notificado sobre o ataque no qual ele será vítima, e também sabe sobre os procedimentos que serão efetuados;
- ***Double Gray Box***: o auditor tem algumas informações sobre o alvo, o alvo tem conhecimento sobre o ataque, mas não sabe sobre os procedimentos que serão executados;
- ***Tandem***: o auditor tem informações detalhadas sobre sua vítima, o alvo sabe que será atacado e quais as tarefas que serão executadas;
- ***Reversal***: o auditor tem detalhes sobre o alvo, mas ele não sabe sobre o ataque, nem dos procedimentos que serão executados.

2.2.2. Etapas de um *Pentest*

Como qualquer planejamento de uma operação em um sistema de TI a ação de um *pentest* é composta por etapas, de modo que haja uma maior eficiência durante a sua execução. Dentre as referências consultadas há uma pequena discrepância na nomenclatura, mas os processos envolvidos são praticamente os mesmos.

Para [Santos 2018], as etapas a serem seguidas são descritas como:

- **Preparação**: refere-se a todos os procedimentos de comum acordo entre o *pentester*[1] e o cliente. Neste escopo serão definidos todos os processos esperados e seus possíveis resultados depois da execução. Dependendo do tipo de teste, conforme classificação apresentada anteriormente, serão levados em conta detalhes como a infraestrutura e os pontos onde deverá haver mais atenção.
- **Coleta de informações**: consiste na aquisição de informações sobre a situação do cliente. A obtenção destas informações direcionará o analista pelos meios a serem explorados.
- **Modelagem das ameaças**: baseia-se no desenvolvimento do plano de ataque. Dependendo do tipo de informação da organização, o plano de ação será direcionado aos elementos de crucial importância para sua existência.
- **Análise de vulnerabilidades**: nesta etapa virão à tona todas as deficiências em segurança do sistema. Poderão ser expostos todos os tipos de ações com possibilidade de interrupção dos processos críticos.
- **Exploração das falhas**: fase onde todas as vulnerabilidades encontradas durante o teste são acessadas.
- **Pós-exploração de falhas**: momento onde todas as vantagens que poderão ser obtidas pelo atacante são expostas.

---

[1] *Profissional com elevado conhecimento sobre técnicas e ferramentas utilizadas em identificação de falhas de segurança em sistemas de informação e infraestrutura de redes.*

- **Geração de relatórios**: depois de executados todos os procedimentos de invasão chega-se ao momento de recolher todas as informações úteis e essenciais para o cliente na melhoria de seus sistemas de segurança da informação.

Estas etapas conduzem a ação de análise de maneira clara e coerente tanto para o analista, como para o cliente, promovendo uma maior integração durante o processo.

2.2.3. *WannaCry*

Com base na teoria exposta e diante de inúmeros casos relatados de invasão há alguns cases de sucesso, referente a finalidade de invasão. Baseado em uma série de eventos de grandes proporções e em uma data não muito distante, mais precisamente em 12 de maio de 2017, um *malware* de escala mundial, o *Ransomware WannaCry* entrou em execução. Segundo [Oliveira 2018], este tipo de *malware* tem como objetivo extorquir os usuários pela informação que foi sequestrada. Em sua ação não era necessária a interação com o usuário final. Efetuava a varredura em um range de IPs da internet, e no momento da constatação executava *exploits* para infectar as vítimas. Em sua nomenclatura possui o termo inglês '*ransom*', que consiste em uma referência a resgate. Inesperadamente, ao ligar um computador a vítima era surpreendida com a tela apresentada da Figura 3.

Figura 3. Tela de avisos do *Ransomware WannaCry* [Oliveira 2018].

O *WannaCry* consiste em um caso clássico que mostrou total despreparo dos sistemas de segurança em tecnologia de barrar a sua disseminação, pois segundo os números da autora, no momento da sua aparição em 12 de maio do referido ano, a quantidade de máquinas infectadas em um primeiro instante consistia em cerca de 45 mil em 11 países, passando a 345 mil máquinas em mais de 150 países, em um segundo momento, mais precisamente em 17 de maio do mesmo ano. Este acontecimento evidenciou vulnerabilidades de diversas partes do planeta, independente de idioma ou outras técnicas.

Este foi apenas um de muitos casos de intrusão que acontecem com frequência em computadores conectados na internet.

2.3. USB *Rubber Ducky*

O padrão de conexão USB trouxe grandes benefícios no âmbito de conexões de periféricos a computadores. Antigamente essa conexão era tarefa de técnicos ou usuários com um significado nível de conhecimento em interconexões de dispositivos.

Conforme explica [Filho 2018], quando um dispositivo for conectado ao *host* (computador) ele faz uma série de solicitações ao dispositivo conectado, como informações básicas de tipo, tamanho, etc. Com o *driver* adequado instalado pelo *host* o dispositivo USB passa para o modo execução e assim o *firmware*[2] passa a controlar a função básica do periférico. Esse *firmware* vem geralmente programado de fábrica, e no momento da sua aquisição presume-se que ele seja executado adequadamente conforme a sua finalidade.

Filho destaca ainda o problema referente a possibilidade de reprogramação de *firmware* de alguns dispositivos. Isso se deve à falta de segurança de implementação para que se evite a reprogramação destas instruções.

A possibilidade de programação de instruções para ser reconhecidas no momento da conexão via porta USB é o princípio de funcionamento do dispositivo *Rubber Ducky*.

Fisicamente trata-se de um dispositivo com uma conexão USB, que no momento do seu acoplamento a porta USB do *host*, poderá executar um *payload* como uma diversidade de finalidades, como por exemplo, abertura de um *backdoor* para exploração da infraestrutura, ou apenas recolher determinados arquivos de interesse. A linguagem de programação utilizada para criar o *script* é o *DuckyScript*. A Figura 4 apresenta em detalhes a sua composição de hardware.

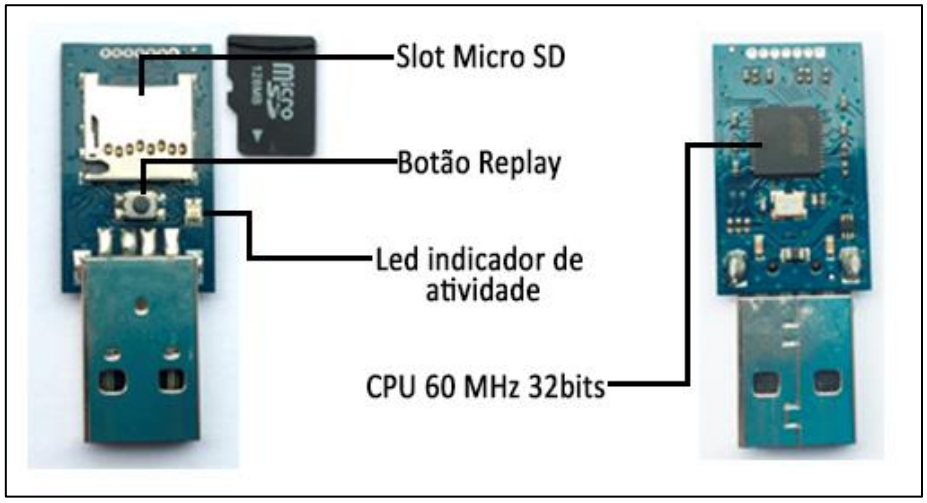

Figura 4. Dispositivo *Rubber Ducky* utilizado nos testes, fabricante HAK5 [Dos autores].

Equipado com um micro controlador de 60 MHz e uma arquitetura de 32 bits, pode executar *scripts* com uma grande variedade de finalidades.

2.4. *Duck Toolkit*

O *Duck Toolkit* trata-se de uma série de ferramentas que permitem codificar e descodificar os arquivos responsáveis por realizar a execução dos comandos do dispositivo *Rubber Ducky*. Disponível em sua *homepage*[3], após inserido o *script* com as funcionalidades a

---

[2] *Consiste em um tipo de software de fábrica que promove o funcionamento de um dispositivo específico.*
[3] *https://ducktoolkit.com/*

serem desempenhadas, permite a geração do *inject.bin*. Na sequência da geração do arquivo, o mesmo deve ser baixado e salvo no microSD que é inserido na placa do *Rubber Ducky*. Se for necessário o *script* inserido também pode ser salvo ainda em formato de texto.

A ferramenta também possui alguns modelos pré-definidos, os quais podem servir de teste ou exemplo durante o desenvolvimento do *script* com os comandos. Conforme a Figura 5 em seu menu lateral a direita pode-se observar todos os comandos responsáveis pela codificação do arquivo *inject.bin.*

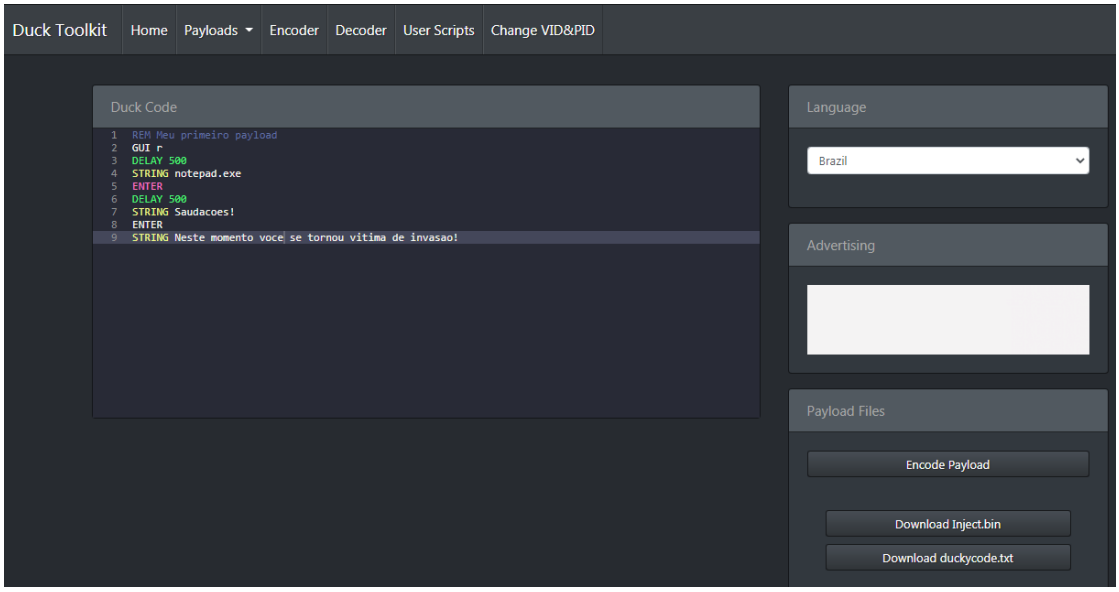

Figura 5. Interface *Web* para a criação de um arquivo *inject.bin* [Dos autores].

O seu funcionamento é simples. Cada nova linha é responsável pela execução de um comando. Na sequência segue uma exemplificação de um código.

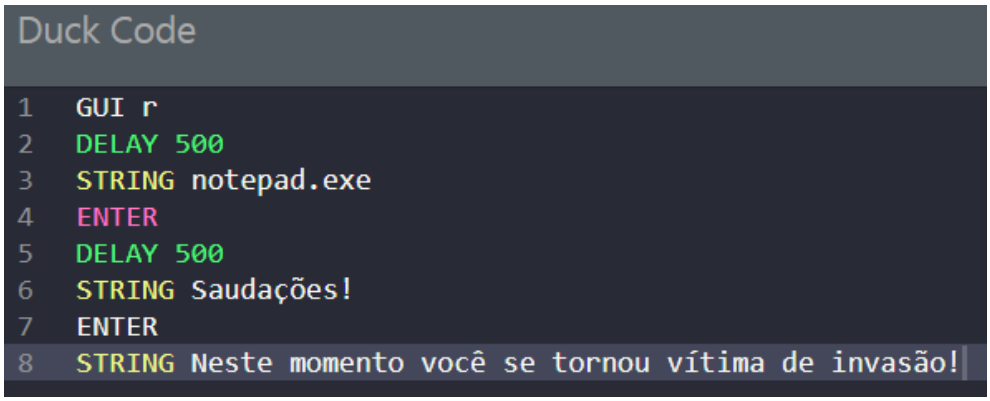

Figura 6. Sintaxe básica para a criação de um arquivo *inject.bin* [Dos autores].

Na Figura 6 pode-se ver uma sintaxe básica para execução no momento da conexão de um dispositivo *Rubber Ducky*. Nele estão definidos os seguintes procedimentos: abertura de um bloco de notas e digitação da frase: “Saudações! Neste momento você se tornou vítima de invasão!”.

Pode-se notar que se trata de uma sintaxe fácil de assimilar. O comando muitas vezes traz escrito por extenso a tecla referente a sua execução. Conforme a Tabela 1, alguns dos comandos mais usados são:

Tabela 1. Sintaxes mais usuais para a criação de um *DuckyScript*.

Sintaxe	Comando
REM	Insere um comentário
GUI	Emula a tecla “Windows”
DELAY	Executa uma espera em milissegundos
STRING	Insere a frase ou caracteres correspondentes
ENTER	Emula a tecla “Enter”
ALT	Emula a tecla “Alt”

[Dos autores].

Conforme apresentado na Tabela 1, estas são apenas algumas das sintaxes usadas na criação de *scripts*. Outro detalhe perceptível refere-se a forma da escrita. O comando inicial, geralmente emulando uma tecla, é escrita em letras maiúsculas.

2.5. Linguagem de Programação Python

Em 2001 foi fundada a PSF (Python Software Foundation), a qual trata-se de uma organização sem fins lucrativos, coordenadora do uso da linguagem até hoje. Recebe apoio de grandes empresas do setor de tecnologia, como Google, Microsoft e a Globo.com.

Sobre a sua aplicação, Silva e Silva, destacam a sua versatilidade. Python trabalha de forma isolada, porém pode ser integrada a outras linguagens, como Java, C, JavaScript, entre outras.

De uma maneira generalista Python trata-se de uma linguagem de altíssimo nível, concretizando uma premissa de seu criador, o qual projetou-a para ser de fácil entendimento por programadores novos interessados em aprendê-la.

2.6. Py2exe

Conforme a *Homepage* Oficial, o módulo Py2exe consiste em uma extensão para a linguagem Python que converte *script*s Python (.py) em executáveis para o Sistema Operacional Microsoft Windows. Após a conversão os executáveis gerados podem ser executados de forma independente, onde o interpretador Python não necessita estar instalado [Python 2020].

Para seu funcionamento correto, durante a sua instalação é necessário estar atento a características como a arquitetura do Sistema Operacional (32 ou 64 bits), bem como a versão da linguagem Python instalada.

2.7. Pyhook

Também consiste em mais um módulo criado para a linguagem Python. Conforme especifica a *Homepage* Oficial, esta extensão tem a finalidade de fornecer retorno e atividades executadas por teclado e mouse de computador. Registra as atividades onde o usuário executa tarefas como ação do mouse direito, esquerdo e tecla digitada. Associado a isso pode retornar informações como data e hora da ação [Pyhook 2020].

3. Trabalhos Correlatos

Neste item serão apresentados alguns trabalhos desenvolvidos dentro da temática de invasão, utilizando técnicas baseadas no dispositivo *Rubber Ducky* e as formas de exploração de vulnerabilidades na execução destes *scripts* automatizados.

3.1. *Hacking Experiment by Using USB Rubber Ducky Scripting*

Neste trabalho de Cannols e Ghafarian, os autores abordaram de maneira prática a utilização da ferramenta *Rubber Ducky*. Seu uso, em apoio de mais algumas técnicas de intrusão mostraram como alguns segundos de desatenção são suficientes para exploração de falhas. Neste caso aproveitaram a situação de falha apresentada pelo Sistema Operacional Windows por possibilitar a obtenção das informações referentes a "Id" e "Senha" dos usuários cadastrados [Cannols and Ghafarian 2017].

Aliado ao uso do dispositivo *Rubber Ducky*, os testes no cenário obtiveram apoio de ferramentas como: *Mimikatz*, uma ferramenta de código aberto que permite a exploração do Windows LSASS (*Local Security Authority Subsystem Service*) através do seu módulo *sekurlsa*, o qual possui uma série de informações, entre os quais estão a "Id" e "Senha" de usuário. A sua execução permite coletar todas as informações desta ".dll". Um detalhe muito importante refere-se a sua segurança, pois este aplicativo é detectado pelo antivírus.

De acordo com a descrição das ferramentas, e quanto ao roteiro deste trabalho, com o uso do *DuckyScript* foi programado o "inject.bin" adicionado ao dispositivo *Rubber Ducky*. Quando conectado em uma porta USB possibilitou a execução do *PowerShell* como administrador, e a partir dele uma série de outros comandos, como execução da ferramenta *Mimikatz* e envio deste log ao servidor de hospedagem definido pelos programadores.

A conclusão disso infere que, apesar deste teste tirar proveito de uma falha do Windows, vários outros riscos podem ser submetidos por um pequeno descuido, permitindo a ação de pessoas mal-intencionadas, bem como critérios e políticas de segurança no manuseio de dispositivos desconhecidos em ambientes com informações confidenciais, tanto pessoais como corporativos.

3.2. *Wi-Fi password stealing program using USB Rubber Ducky*

Neste trabalho também envolvendo o dispositivo *Rubber Ducky*, Harianto e Gunawan realizaram um teste de penetração com o objetivo de capturar a senha de WI-FI a qual o computador está conectado. Foram usadas algumas ferramentas de apoio, mas o *Rubber Ducky* foi usado como dispositivo desencadeante da ação [Harianto and Gunawan 2018].

Segundo a Figura 7, o roteiro da pesquisa foi baseado em 3 passos principais.

Figura 7. Fluxograma de passos para a captura de senhas WI-FI [Harianto e Gunawan 2018].

Com o acesso a máquina física todos os passos restantes são executados no código armazenado pelo inject.bin. Uma ação clássica deste tipo de invasão faz uso de transferência de arquivos online para servidores via FTP (*File Transfer Protocol*), exatamente conforme aplicado neste projeto.

Durante a sua ação foram realizadas basicamente as seguintes tarefas:

1. Abertura de prompt de comando como administrador;
2. Desativação de Firewall;
3. Extração das informações de senha com o comando "netsh wlan export profile key=clear" e exportação para arquivo XML;
4. Compressão para arquivo .ZIP;
5. Upload de arquivo via FTP;
6. Eliminação dos arquivos e diretórios criados; ativação de Firewall;
7. Fechamento de prompt de comando.

Durante a análise dos resultados deste trabalho foi ainda possível avaliar o nível de força utilizado pelos usuários em suas senhas. A aplicação dos testes foi realizada por uma totalidade de 35 vezes, onde os resultados obtiveram sucesso em 33 vezes. Em 2 execuções ocorreu falha na captura pelo motivo da senha conter caracteres além de números e letras alfabéticas. Com base nisso pode ser destacada a importância de uma senha com nível apropriado de segurança.

4. Metodologia

Consoante ao trabalho apresentado de [Cannols and Ghafarian 2017], e [Harianto and Gunawan 2018], o dispositivo *Rubber Ducky* consiste na execução de um *payload* automatizado, programado por uma linguagem própria. O disparo da sua execução se sucede depois da sua conexão a uma porta USB de uma máquina física. Primeiramente tornou-se necessário o estudo da linguagem responsável pelos seus comandos, o *Ducky Script*. A partir de sua sintaxe foram elaboradas as rotinas e comandos. Ao finalizar as instruções é criado o arquivo responsável pela ação no momento da sua conexão, o "inject.bin".

A implementação desta solução de defesa teve como princípio norteador a realização de teste originado a partir de um projeto open-source orientado e disponibilizado por [Pedro Miguel Sosa 2020]. Pedro é estudante de segurança e criptografia em computação pela Universidade de Califórnia, em Santa Bárbara/EUA. De maneira bem didática Pedro instiga o leitor a tentar compreender como um dispositivo projetado para a computação forense como um *Rubber Ducky* pode ser identificado.

Considerando os princípios de funcionamento deste dispositivo, podemos nos indagar com o seguinte questionamento: como identificar uma invasão por *Rubber Ducky* se no momento da sua conexão a uma porta USB um computador reconhece-o como um dispositivo de entrada, como um teclado? Como impedir uma série de comandos automatizados por um *script* se nem o mais sofisticado antivírus não possui capacidade de identifica-lo?

Com o objetivo de preencher esta lacuna de segurança, e para comprovação se de fato esta solução funcionaria, foi adquirido o dispositivo oficial da fabricante HAK5, uma das mais conceituadas empresas na fabricação deste tipo de hardware. Infelizmente em seu site oficial da Hak5[4] não há exportação direta para o Brasil, porém por intermédios foi realizada a sua aquisição.

[4] *Disponível em shop.hak5.org*

Baseado em seu *blog*[5] pessoal e *GitHub*[6], Pedro Sosa traz uma reflexão para identificar este tipo de invasão baseado em sua forma de funcionamento. Quando um *Rubber Ducky* é conectado fisicamente a um host começa a execução de *script*, o qual em algum momento certamente vai disparar um comando baseado em uma *string* de caracteres. Conhecendo alguns dos comandos básicos do Sistema Operacional pode-se caracterizar cada comando com uma média de 10 a 20 caracteres. Então neste ponto chega-se a um momento chave no desenvolvimento deste projeto: **identificar se o comando é humanamente possível de ser digitado dentro de um instante de tempo.**

A linguagem de programação Python possui módulos que fazem a captura de eventos de teclado e mouse. Sabendo-se da execução extremamente rápida de um *script* por um *Rubber Ducky*, este módulo terá a responsabilidade de armazenar um histórico com a entrada de caracteres. Se a velocidade de entrada for maior que o tempo estipulado, necessariamente haverá uma interrupção da entrada de caracteres, deixando a execução do *script* incompleta.

Para isso foram elaborados 3 arquivos e identificados como:

- **duckhunt.conf:** arquivo de configuração, o qual pode ser personalizado pelo usuário. Ao abri-lo com algum editor de texto haverá duas variáveis que podem ser editadas, estipulando a quantidade de caracteres monitorados e o espaço de tempo em milissegundos para a identificação;
- **setup.py:** arquivo necessário para a compilação de executável para o Sistema Operacional Windows pelo módulo Py2exe;
- **duckunt_Ajustável.py:** este é o arquivo que contém a programação responsável pela identificação da invasão. As importações estão destacadas na Figura 8.

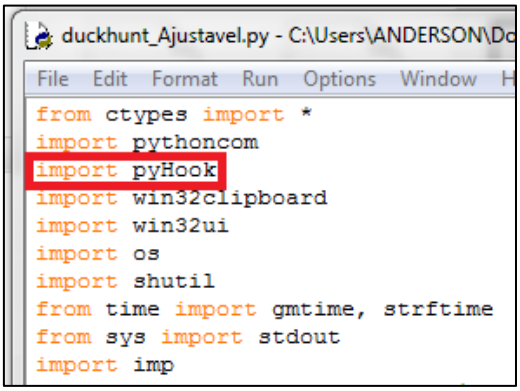

Figura 8. Importação dos módulos Python, com destaque ao módulo PyHook.
Fonte: Dos autores.

Depois da compilação do executável com o módulo Py2exe, são gerados dois diretórios: um denominado "build" e outro "dist". A pasta "dist" conterá o executável que deverá ser executado, conforme apresentado na Figura 9. Basta executá-lo e finalmente dentro dos processos do Windows, iniciará um processo com o mesmo nome. A partir deste momento o Sistema Operacional estará protegido.

[5] *http://konukoii.com/blog/2016/10/26/duckhunting-stopping-automated-keystroke-injection-attacks/*
[6] *https://github.com/pmsosa/duckhunt*

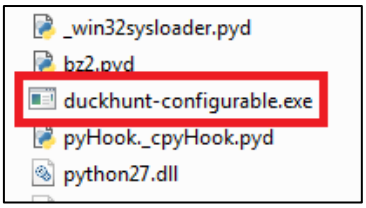

Figura 9. Compilação do arquivo executável com o módulo Py2exe [Dos autores].

Como o objetivo deste trabalho consiste em encontrar uma solução que de alguma maneira realize o bloqueio da ação deste tipo de dispositivo, o *script* criado não realizará tarefas muito complexas. A Figura 10 apresenta o seu *script* criado com a ferramenta on-line *Duck Toolkit*.

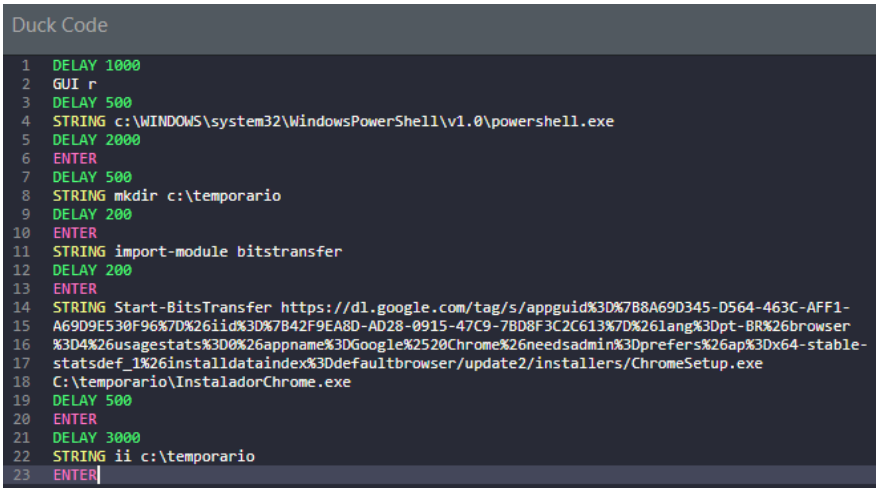

Figura 10. Elaboração do *DuckScript*, o qual dará origem ao arquivo *inject.bin* [Dos autores].

Uma das tarefas mais comuns realizadas com este tipo de hardware consiste em transferência de aplicações a partir de servidores on-line via FTP e utilização de comandos robustos via *PowerShell*. Para demostrar de maneira didática como o processo de transferência de arquivos é simples, o *script* acima define a execução sequencial das seguintes tarefas:

1. Abertura da caixa de diálogo Executar no Windows;
2. Abertura do Windows *PowerShell*;
3. Criação da pasta "temporario" no disco local C:\, para onde o arquivo será baixado;
4. Criação de um serviço de transferência de arquivos;
5. Download do arquivo instalador do navegador Google Chrome e ao mesmo tempo renomeando-o como "InstaladorChrome.exe";
6. Exibição de tela referente ao diretório para onde o download foi executado (C:\temporario);

Após a compilação destes comandos basta apenas copiar o *inject.bin* criado para o cartão MicroSD que será inserido dentro do slot do *Rubber Ducky*. No momento da sua conexão a alguma máquina o disparo destes comandos será realizado de forma automática. O resultado da execução após a sua conexão pode ser observado na Figura 11.

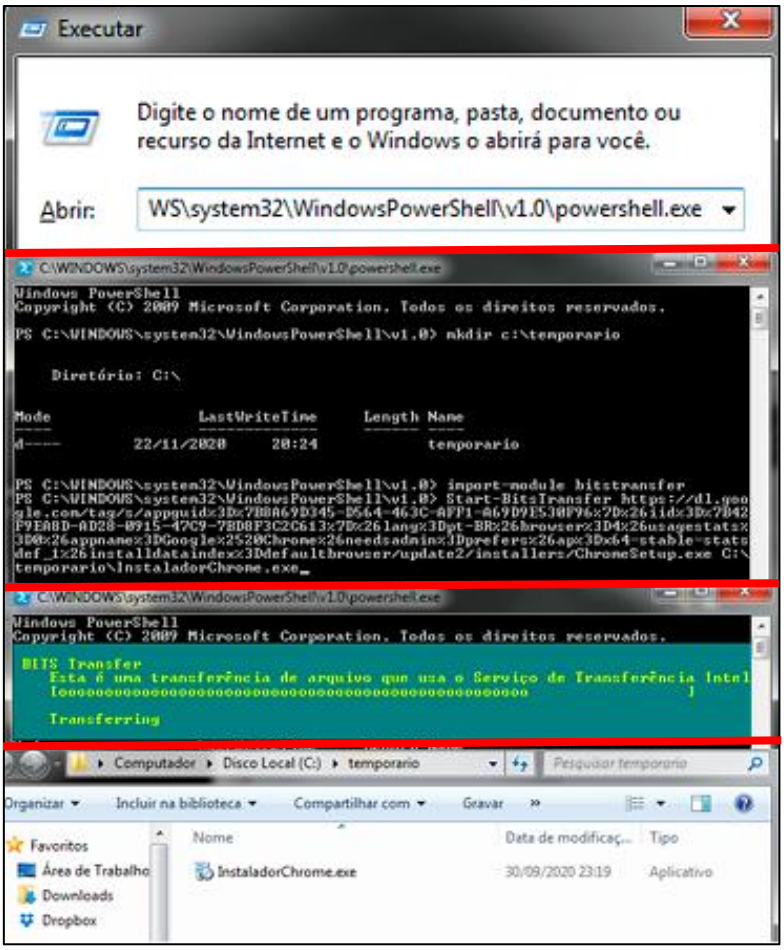

Figura 11. Etapas de execução do *Rubber Ducky* após a sua conexão [Dos autores].

Observando as etapas de execução do *Rubber Ducky* na Figura 11, pode-se detalhar os seguintes passos: invocação do *PowerShell* via caixa de diálogo Executar, criação do diretório "temporário" para onde o arquivo será baixado; criação de serviço para download do instalador; exibição do diretório para onde o arquivo foi baixado.

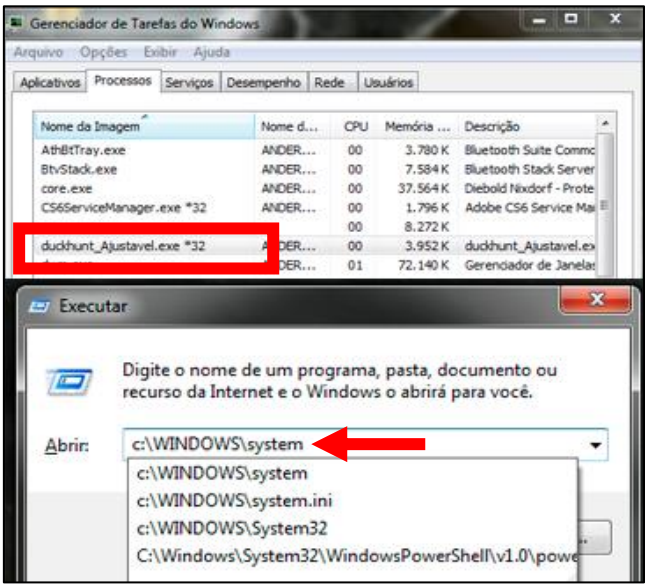

Figura 12. Interrupção de execução do *script* do *Rubber Ducky* [Dos autores].

Finalmente depois da execução da solução desenvolvida para identificação do *Rubber Ducky*, a qual pode ser observada na Figura 12, tem-se o "Gerenciador de Tarefas" com execução do bloqueio e após a inserção do tamanho da palavra definida no arquivo "duckhunt.conf", o sistema identificou que estava sendo vítima de um *Rubber Ducky*. Deste modo a execução do *script* foi interrompido e o restante das instruções não foram executadas.

5. Conclusões

Ao abordar o funcionamento de um dispositivo Rubber Ducky, pôde-se constatar que o mesmo desempenha com eficiência a execução das tarefas programadas através do *Duck Toolkit*.

Um cenário composto por uma sala equipada com vários computadores instalados e com informações privilegiadas armazenadas caracteriza-se como um potencial risco iminente para pessoas mal-intencionadas na obtenção de informações alheias. Basta um momento de descuido e conforme a simulação dos testes, a ação deste dispositivo pode ser instantânea. Com estes testes, expôs-se uma forma de penetração computacional extremamente eficiente, e ao mesmo tempo foi desenvolvida uma solução para tal bloqueio, completando a premissa *Ethical Hacker*, a qual consiste exatamente na identificação/solução de vulnerabilidades em ambientes de tecnologia da informação.

Por sua vez a versatilidade da linguagem Python, juntamente com seus módulos adicionais, como o PyHook e o Py2exe ofereceram uma solução opcional para impedir o sucesso da ação deste tipo de dispositivo por meio da interrupção de *script*.

Assim, por meio dos resultados coletados pelas tentativas de ataque e de bloqueio pelo dispositivo e pela solução desenvolvida, chegou-se a uma solução parcial, que neutraliza o ataque de software após sua inicialização, porém não interrompendo-o em seu funcionamento físico.

A segurança cibernética deve estar em constante revisão, comprovado pela exposição de mais um tipo de dispositivo, capaz de causar sérios danos na confidencialidade de informações.

5.1. Trabalhos Futuros

Considerando os resultados obtidos, sugere-se para trabalhos futuros e novas pesquisas em objetivos que façam abordagem a:

- Complementação de proteção para outros sistemas operacionais, como Android e MacOS;
- Outras Linguagens e módulos com suporte capazes de identificar este tipo de dispositivo;

Referências

Cannols, B. e Ghafarian, A. (2017) "Hacking Experiment by Using USB Rubber Ducky Scripting", Artigo, Revista Systemics, Cybernetics and Informatics, vol. 15, n. 2, http://www.iiisci.org/journal/CV$/sci/pdfs/ZA340MX17.pdf, maio.

HAK5 (2020) "USB Rubber Ducky", https://shop.hak5.org/collections/hotplug-attack-tools/products/usb-rubber-ducky-deluxe, maio.

Harianto, H. E. e Gunawan, D. (2018) “Wi-Fi password stealing program using USB rubber ducky”, Artigo, TELKOMNIKA (Telecommunication, Computing, Electronics and Control), ISSN 1693-6930, Vol.17, No.2, April 2019, p.745-752, http://journal.uad.ac.id/index.php/TELKOMNIKA/article/view/11775/6428, outubro.

Konzen, M. P. (2013) “Gestão de riscos de Segurança da Informação baseada na Norma NBR ISSO/IEC 27005 usando padrões de segurança”, Dissertação de Mestrado em Engenharia de Produção, Universidade Federal de Santa Maria, https://repositorio.ufsm.br/bitstream/handle/1/8276/KONZEN%2c%20MARCOS%20PAULO.pdf?sequence=1&isAllowed=y, abril.

Marciano, J. L. P. (2006) “Segurança da Informação – uma abordagem social”, Tese de doutorado em Ciência da Informação, Universidade de Brasília, https://repositorio.unb.br/bitstream/10482/1943/1/Jo%c3%a3o%20Luiz%20Pereira%20Marciano.pdf, abril.

Oliveira, J. C. de (2018) “*Ransomware* - Laboratório de Ataque do *WannaCry*”, Trabalho de Conclusão de Curso em Engenharia de Software, Universidade de Brasília, https://bdm.unb.br/bitstream/10483/23052/1/2018_JessicaCristinaDeOliveira_tcc.pdf, abril.

Pavan, P. V. A. e Guardia, H. C. (2015) “*Pentest* para auditoria de segurança de rede em ambientes corporativos”, Artigo, Revista Tecnologias, Infraestrutura e Software, v. 4, n. 2, mai-ago, http://revistatis.dc.ufscar.br/index.php/revista/article/view/314/113, maio.

PyHook. (2020) Página Inicial. https://pypi.org/project/pyHook/#description, outubro.

Python: Py2exe (2020) Página Inicial. http://www.py2exe.org/index.cgi/FrontPage, outubro.

Rocha, A. T. Q. et al. (2019) “*Pentest* para Quebra de Criptografia Wireless”, Artigo, Caderno de Estudos Tecnológicos, nº1, v. 6, ed. Junho, http://www.fatecbauru.edu.br/ojs/index.php/CET/article/view/203/174, maio.

Santos, R. E. C. dos (2018) “Laboratório Virtual Para *Pentest* Na Análise De Vulnerabilidade”, Trabalho de Conclusão de Curso em Engenharia da Computação, Centro Universitário de Brasília, https://repositorio.uniceub.br/jspui/bitstream/235/12374/1/20967289.pdf, maio.

Silva, C. A. da (2009) “Gestão da Segurança da Informação: um olhar a partir da Ciência da Informação”, Dissertação, Mestrado em Ciência da informação, Pontifícia Universidade Católica, http://tede.bibliotecadigital.puc-campinas.edu.br:8080/jspui/bitstream/tede/819/1/Claudete%20Aurora%20da%20Silva.pdf, abril.

Silva, R. O. da e Silva, I. R. S. (2019) “Linguagem de Programação Python”, artigo, Revista Tecnologias em Projeção, ISSN: 2178-6267, v10, nº 1, p. 55-71, http://revista.faculdadeprojecao.edu.br/index.php/Projecao4/article/view/1359, outubro.

Sosa, P. M. (2020) “DuckHunter”, Github, https://github.com/pmsosa/duckhunt, setembro.