

ESPECIAL: Primeiros passos para implantar BYOD
Quais os recursos disponíveis no mercado?



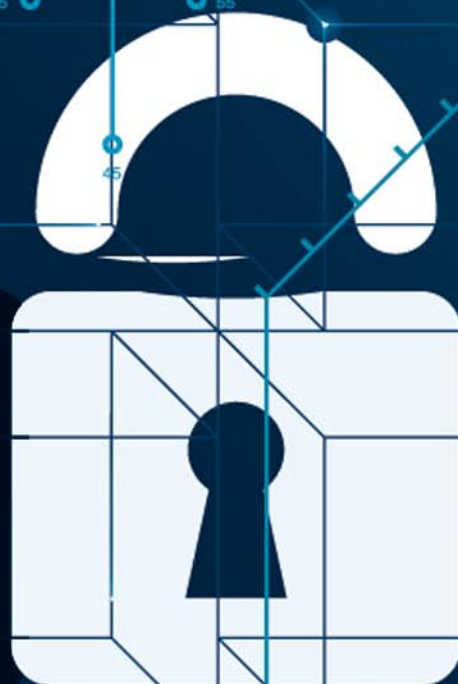
Edição 15



Introdução ao Wireshark
Saiba como avaliar o tráfego da rede

Revelando o conteúdo da memória RAM
Proteja o ambiente
de TI com Memoryze e Redline

COMO EVITAR INVASÕES COM HONEYPOTS



SEU CARRO TEM SEGURO, SUA SAÚDE TEM SEGURO, MAS E O SEU EMPREGO... TÁ SEGURO??



NÃO DEIXE JUSTAMENTE A SUA CARREIRA FICAR EM RISCO!

Manter-se atualizado com todas as novidades do mercado de desenvolvimento é obrigação de todo bom programador. Faça agora mesmo um seguro para a sua carreira. Seja um assinante MVPI

Saia do risco!

TENHA ACESSO A:



+DE **260** CURSOS ONLINE



09 REVISTAS MENSAIS



7.850 VÍDEO-AULAS

POR
APENAS **59,90**
MENSAL

QUEM TEM ESTÁ TRANQUILO.



DEV MEDIA

Acesse: www.devmedia.com.br/mvp



Edição 15 • 2014



Assine agora e tenha acesso a todo o conteúdo da DevMedia: www.devmedia.com.br/mvp

EXPEDIENTE

Editor

Eduardo Spinola (eduspinola@gmail.com)

Sub Editores

Marco Antônio Pereira Araújo (maraujo@devmedia.com.br)

Rodrigo Oliveira Spinola (rodrigo.devmedia@gmail.com)

Consultora Técnica

Daniella Costa (daniella@devmedia.com.br)

Jornalista Responsável Kaline Dolabella - JP24185

Capa e Diagramação Romulo Araujo

Atendimento ao leitor

A DevMedia possui uma Central de Atendimento on-line, onde você pode tirar suas dúvidas sobre serviços, enviar críticas e sugestões e falar com um de nossos atendentes. Através da nossa central também é possível alterar dados cadastrais, consultar o status de assinaturas e conferir a data de envio de suas revistas. Acesse www.devmedia.com.br/central, ou se preferir entre em contato conosco através do telefone 21 3382-5038.

Publicidade

publicidade@devmedia.com.br — 21 3382-5038

Anúncios — Anunciando nas publicações e nos sites do Grupo DevMedia, você divulga sua marca ou produto para mais de 100 mil desenvolvedores de todo o Brasil, em mais de 200 cidades. Solicite nossos Media Kits, com detalhes sobre preços e formatos de anúncios.

Fale com o Editor!

É muito importante para a equipe saber o que você está achando da revista: que tipo de artigo você gostaria de ler, que artigo você mais gostou e qual artigo você menos gostou. Fique a vontade para entrar em contato com os editores e dar a sua sugestão!

Se você estiver interessado em publicar um artigo na revista ou no site Infra Magazine, entre em contato com o editor, informando o título e mini-resumo do tema que você gostaria de publicar:



EDUARDO OLIVEIRA SPÍNOLA

eduspinola.wordpress.com

[@eduspinola](https://twitter.com/eduspinola) / [@Java_Magazine](https://twitter.com/Java_Magazine)

Conteúdo sobre Boas Práticas

04 – Revelando a memória RAM com Memoryze e Redline

[*Marcelo Lau e Nichols Jasper*]

Conteúdo sobre Boas Práticas

10 – Honeypots: Evitando invasões

[*Rodrigo Sanches Miani*]

17 – Python: Automação de tarefas

[*Mateus Espadoto*]

28 – BYOD: primeiros passos para implantação

[*Juliana Márcia Rezende Calado*]

Sumário

Revelando a memória RAM com Memoryze e Redline

Analise o conteúdo da memória de forma simples e gratuita

Na atualidade presenciamos o fato de que diversas empresas não atualizam os sistemas operacionais de suas estações de trabalho e servidores, assim como seus softwares, aplicativos, ferramentas e plug-ins. Esta ausência de aplicação das correções de segurança é aproveitada por diversos tipos de *malwares* (softwares que podem danificar, alterar ou furtar informações), que podem se instalar e comprometer o funcionamento de uma ou mais máquinas. Uma vez que isso aconteça, qual decisão tomar? Formatar uma estação de trabalho comprometida pode resolver o problema, mas não permitirá que se identifiquem as causas deste incidente de segurança, impossibilitando que se desenvolva uma compreensão deste e que lições possam ser aprendidas para evitar um problema futuro no ambiente. Existem soluções mais recomendadas para este tipo de problema do ponto de vista de segurança, e uma delas é identificar e isolar a máquina que está infectada, visando diminuir o impacto causado na rede.

Por si só um software antivírus apenas é eficiente caso o seu banco de dados esteja atualizado e em execução. Entretanto, é sabido que estas tecnologias possuem falhas e seus métodos não conseguem identificar plenamente os softwares maliciosos. Uma alternativa para contornar essa limitação é analisar a memória do computador em busca de comportamentos suspeitos que forneçam indícios de atividade maliciosa. Para executar esta ação é necessário capturar uma “imagem” da memória da máquina. Essa “fotografia” da memória em um dado momento do tempo é um item essencial para compreender as atividades de um computador, esclarecendo dúvidas em relação a atividades suspeitas dentro de um host ou na rede.

Fique por dentro

Este artigo apresenta as ferramentas da empresa Mandiant: Memoryze e Redline, em um estudo de caso aonde podemos empregar suas funcionalidades para compreender melhor as atividades em execução dentro da memória de um computador. Neste estudo considera-se um cenário hipotético, onde há suspeita de infecção por um malware desconhecido em um computador e provoca danos e lentidão em toda a rede. O objetivo é analisar os processos desta estação de trabalho em busca de sinais que possam endossar ou refutar a hipótese de infecção por malware neste host. Os pontos destacados neste artigo serão úteis para melhorar a compreensão de um perito em forense computacional quanto à importância e facilidade da análise de memória tanto para a validação de incidentes que possam comprometer a segurança computacional ou para melhor compreender os processos executados e registrados na memória do computador.

Quando dizemos atividades suspeitas dentro de um computador, nos referimos a comportamentos incomuns e muitas vezes indesejados em um sistema, como um download de um programa sem a autorização do usuário ou o desligamento do sistema de firewall pessoal do PC. Na prática essas atividades ocorrem através dos programas executados no computador, que alocam os recursos computacionais e do sistema operacional para conseguir fazer suas ações. Quando em execução, esses aplicativos são conhecidos como processos.

Os processos, no contexto computacional, representam as tarefas em execução, mas nem todas têm relação direta com algum aplicativo em uso no sistema. Muitos dos processos funcionam em *background*, isto é, não interagem diretamente com o usuário. Estes processos mantêm o sistema trabalhando devidamente e efetuam a gestão dos recursos deste, como o acesso a rede, a memória,

o uso do disco rígido, a proteção do software de antivírus, etc. Simplificando, podemos definir os processos como softwares que executam alguma ação e que podem ser controlados de alguma maneira, seja pelo usuário, pelo aplicativo correspondente ou pelo sistema operacional.

Estes processos estão presentes na memória e analisá-los nos permite compreender o que está ocorrendo no computador em um dado momento do tempo. Para tal, é necessário obter o conteúdo da memória RAM do computador, processo que chamamos de aquisição da memória. A aquisição da memória possibilita a extração da lista dos processos em execução no computador. Além de extraí-los, existem softwares no mercado que conseguem investigar as atividades correntes e atribuir-lhes uma classificação conforme seu potencial de execução de atividades maliciosas, exibindo a origem destes processos no sistema operacional e os arquivos e registros que interagem com eles no sistema.

Para executar a atividade de análise da memória, abordaremos duas ferramentas gratuitas desenvolvidas pela Mandiant que executam os processos de aquisição e análise forense da memória. Uma delas é o Memoryze, ferramenta leve e muito eficiente, cujos resultados por ele coletados podem ser visualizados com mais clareza e alto nível de detalhes em outro software da Mandiant, o Redline. Este software, além de visualizar resultados coletados pelas ferramentas do Memoryze, desempenha outras funções de análise forense, como a detecção de sinais de atividades maliciosas através da memória e em arquivos, criando um perfil de ameaças existentes dentro de um host.

Este artigo abordará ambas as ferramentas, dando enfoque no componente de análise de processos do Memoryze e nas funções de análise de processos do Redline. A seguir detalharemos um caso hipotético no qual as duas ferramentas podem trabalhar em conjunto para obter respostas a quesitos de perícia computacional.

Cenário pericial

Depois de observarem um aumento repentino no tráfego de rede e uma grande lentidão na rede corporativa de uma empresa, é decidido que esta condição deve ser mais bem investigada para que se descubra a razão deste incidente, atribuído pelos administradores de TI à presença de algum malware, visto que o aumento da latência de rede pode ser fruto da atividade de um worm de rede. Deste modo, um perito é chamado para investigar o incidente e se reúne com a equipe de TI para entender o cenário e o escopo da perícia.

Após se contextualizar do ocorrido e conversar com os administradores da TI da empresa, o perito percebe que algumas estações de trabalho funcionam com o sistema operacional Windows XP e que ele se encontra desatualizado, o que possibilita que *malwares* possam explorar com maior facilidade a segurança do sistema.

Visando encontrar processos suspeitos na máquina, o perito começa a isolar a estação de trabalho suspeita. Após o isolamento, o perito se propõe a iniciar uma varredura com o Memoryze ins-

talado em seu pendrive. Ele irá executar o script batch *Process.bat* (que será comentado posteriormente neste artigo) para coletar informações sobre os processos que estão em execução no sistema e posteriormente irá executar o Redline instalado em seu notebook, importar estas informações coletadas pelo Memoryze e analisar os resultados para gerar um relatório e validar se o computador realmente se encontra infectado.

Mandiant Memoryze

Para iniciar a resposta ao incidente vamos utilizar o Memoryze. Esta ferramenta se destina a adquirir o conteúdo da memória RAM de um computador através dos seus *scripts*, que podem relacionar e classificar todos os processos em execução, drivers e ganchos ou *hooks* (código que manipula chamadas de funções, mensagens ou eventos interceptados), que estão presentes na memória da máquina analisada.

Segundo o próprio site da desenvolvedora, o Memoryze consegue coletar as seguintes informações:

- Os *handles* – estrutura relacionada aos recursos do sistema operacional;
- Todas as bibliotecas DLLs utilizadas por cada processo contido na memória;
- Listagem dos sockets de rede abertos, identificando as conexões de rede abertas por cada processo;
- Funções importadas e exportadas por cada arquivo executável;
- Assinaturas digitais dos arquivos utilizadas por cada processo.

Além disto, a ferramenta consegue listar os componentes carregados no kernel do sistema operacional e os drivers de sistema que se encontram em execução. É importante salientar que diversos *rootkits* (ver **BOX 1**) se mascaram como drivers para se ocultar das ferramentas de segurança existentes no sistema operacional.

BOX 1. Rootkits

Um rootkit é um software malicioso que busca se esconder de softwares de segurança e do usuário utilizando diversas técnicas avançadas de programação. Rootkits camuflam a sua presença no sistema escondendo suas chaves no registro e escondendo os seus processos no Gerenciador de Tarefas, além de retornar sempre erros de “arquivo inexistente” ao tentar acessar os seus arquivos. Os rootkits mais sofisticados são de difícil remoção do sistema.

O Mandiant Memoryze pode ser instalado na máquina ou pode ser executado via dispositivo USB, de maneira que as análises possam ser feitas de forma menos intrusiva, dispensando a necessidade de instalação. Esta é uma abordagem menos invasiva no sistema alvo e visa não comprometer a integridade das evidências colhidas no sistema alvo. Este será o método que adotaremos para o caso abordado em breve.

Para utilizar a ferramenta adequadamente, devemos conhecer sua estrutura e funcionalidades. A estrutura do Mandiant Memoryze é composta por diversos scripts executáveis em formato .bat. São eles:

- **MemoryDD.bat:** Captura uma imagem da memória física;
- **Process.bat:** Enumera todas as informações sobre um processo específico, como memória virtual, portas de rede, handles e strings;
- **ProcessDD.bat:** Captura o conteúdo dos espaços de endereçamento de cada processo em execução na memória;
- **DriverSearch.bat:** Busca por drivers na memória analisada;
- **DriverDD.bat:** Captura a imagem de um driver específico;
- **DriverWalkList.bat:** Classifica todos os módulos e drivers numa lista;
- **HookDetection:** Busca *hooks* no sistema operacional.

Conhecendo a estrutura e as características básicas da ferramenta, deve-se configurar a mídia removível para executar os scripts .bat no sistema alvo da perícia. Para executar o Mandiant Memoryze via dispositivo USB, é necessário realizar o download do setup e seguir as instruções no guia de usuário para isto.

A instalação do Memoryze em um dispositivo USB necessita de um comando que deve ser digitado no prompt de comando do Windows para instalar a ferramenta com sucesso. Vale lembrar que a instalação não deve ser feita na máquina a ser periciada e sim em um computador apartado e confiável, de preferência uma máquina de uso restrito do perito computacional para evitar possíveis interferências no processo de criação da mídia.

Para instalar o Memoryze no dispositivo USB, devem ser realizadas as seguintes instruções:

1. Primeiramente, executar o prompt de comando e navegar até a pasta onde se encontra o setup da ferramenta. Utilize o comando `cd` para concluir esta tarefa;
2. Na pasta onde se encontra o setup do Memoryze (neste exemplo, a pasta *Meus Documentos*), utilizar o comando: `Msiexec /a MemoryzeSetup3.0.msi /qb TARGETDIR= letra correspondente à mídia removível`.

Um exemplo do comando de instalação em uma unidade USB seria: `Msiexec /a MemoryzeSetup3.0.msi /qb TARGETDIR=E:\Mandiant`, onde especificamos que o arquivo a ser instalado é o `MemoryzeSetup3.0.msi` e o diretório de destino é o `E:\Mandiant`. Neste caso a nossa unidade USB está montada no sistema operacional com a letra `E:\`.

Após instalarmos os arquivos do Memoryze no dispositivo USB, as seguintes pastas contendo os arquivos .bat estarão disponíveis:

- *x64* – contendo os executáveis para sistemas de 64 bits;
- *x86* – contendo os executáveis para sistemas de 32 bits.

Executando o Memoryze no host alvo

Com a ferramenta Memoryze instalada de forma portátil no pen drive e pronta para ser executada, devemos inserir a mídia no host com suspeita de comprometimento por malware. O objetivo é tentar descobrir o porquê do aumento de tráfego ocasionado na rede através da análise do conteúdo da memória do computador, em particular dos processos ativos no sistema. Esta informação

pode ajudar na detecção de softwares maliciosos que não são detectados pelos antivírus, mas que reconhecidamente causam problemas no sistema.

Para executar o script de coleta de processos, devemos navegar até o diretório do pen drive aonde instalamos o Memoryze e acessar a pasta *x86* (o sistema alvo em nosso caso é de arquitetura 32 bits). Lá executamos o script *Process.bat*, clicando duas vezes sobre ele.

Após a execução, será criada uma pasta denominada *Audits*, que armazenará todos os resultados da execução da ferramenta. Estes arquivos serão posteriormente submetidos a análise no Redline, que possui uma série de assinaturas e identificadores de processos indesejados.

Conforme citado, dentro dessa pasta (*Audits*) ficam armazenados os resultados de cada análise realizada pelos scripts .bat do Memoryze. Os arquivos resultantes estão em formato XML. A seguir há um exemplo de saída da ferramenta após analisar um Windows XP:

- *BatchResults.xml*: contém informações do comando utilizado na execução do Memoryze e as informações dos parâmetros de execução, assim como o nome do arquivo principal que contém os resultados da análise;
- *Issues.BatchResults.xml*: armazena as informações relativas à estrutura do XML;
- *issues.mir.w32processes-memory.083a0b33.xml*: contém mensagens que podem indicar erros no funcionamento do Memoryze. O conteúdo deste arquivo é útil para fins de depuração e podem ajudar na identificação de problemas na execução da ferramenta;
- *mir.w32processes-memory.083a0b33.xml*: é o arquivo com as informações mais relevantes para análise, pois contém os resultados da execução do Memoryze.

Após executar o Memoryze e obter os resultados, iremos percorrer sobre a ferramenta Redline, utilizando suas funções para analisar os arquivos que acabamos de coletar.

Mandiant Redline

O Mandiant Redline, assim como o Memoryze, também pode ser adquirido de forma gratuita através do endereço indicado na seção **Links**. No caso que estamos abordando, utilizaremos o Redline para visualizar os resultados dos testes executados com o Memoryze, porém o Redline também é capaz de realizar varreduras mais simples, completas ou diretamente nos diretórios suspeitos. Com toda essa capacidade, esta ferramenta acaba sendo mais invasiva no host suspeito, podendo influenciar na qualidade da evidência ao ocupar um maior espaço na memória e acessar uma maior quantidade de recursos da máquina alvo.

Com uma interface amigável e de fácil navegação, o Mandiant Redline apresenta para o analista, de forma muito simples, os resultados dos testes coletados via Memoryze, permitindo realizar a filtragem das informações que serão analisadas.

Seu processo de instalação é bastante simples, como o processo de instalação de qualquer outro software, não sendo necessário, por isso, sua apresentação de forma detalhada. Ao finalizar a instalação com sucesso, uma tela será exibida. Neste ponto basta clicar apenas em *Close* e o Redline será inicializado. A **Figura 1** apresenta a tela inicial do Redline.

Para realizar a análise dos dados colhidos com o Memoryze, escolha a opção *From a Collector* dentro da seção *Analyze Data*. Feito isso, uma tela como a da **Figura 2** deverá aparecer para que os arquivos sejam carregados no Redline.

Como visto na **Figura 2**, no campo *Audit Location*, que está destacado em vermelho, deve-se especificar o caminho para a pasta *Audit*. Esta pasta é aquela que foi criada automaticamente quando executamos o Memoryze e é nela que ficam os resultados da análise realizada anteriormente, em arquivos XML. Devemos então encontrar esta pasta dentro de nosso dispositivo USB, que foi onde instalamos o Memoryze, e selecioná-la como objeto de análise. Após a seleção da pasta, apenas clique em *Next*.

Em seguida, o Redline solicita que seja selecionada uma pasta de destino (que esteja vazia) para salvar o resultado da análise. Este arquivo é gerado num formato específico da ferramenta (.mans).

Após a geração do arquivo, podemos carregá-lo pelo Redline. Dentro da ferramenta, logo abaixo da opção *Start Your Investigation*, aparecem cinco opções com os seus respectivos descritivos, como mostra a **Figura 3**. Dependendo de cada contexto investigativo, uma das opções se adapta melhor conforme a necessidade da perícia em questão. A seguir explicamos sucintamente cada uma delas:

- *I am Reviewing a Triage Collection from MSO* – Analisa o arquivo de forma integrada com outra solução da Mandiant, o MSO, que coleta informações de um host suspeito para serem analisadas pelo Redline;

- *I am Investigating a Host Based on an External Investigative Lead* – É a análise padrão dos arquivos de auditoria da memória que coletamos. Com ela iremos analisar os arquivos XML em busca de conteúdo suspeito;

- *I am Reviewing a Full Live Response or Memory Image* – Opção usada para analisar uma imagem de memória focando no MRI (*Malware Risk Index*) dos processos. Ela gera uma classificação a partir dos MRI utilizando uma variedade de técnicas para avaliar o risco de um determinado processo ser considerado um software malicioso. Processos com um MRI maior são mais propensos a serem maliciosos, enquanto

um com classificação menor possui menor potencial. É possível observar um relatório do MRI de cada processo clicando duas vezes em um determinado processo e acessando a aba *MRI Reports*;

- *I am Reviewing Web History Data* – Analisa o histórico de navegação na Internet do computador. Ele pode ser coletado através do próprio Redline;

- *I want to Search My Data With a Set of Indicators of Compromise* – Opção para situações em que se deseja buscar por determinados IOCs (Indicadores de Comprometimento – ver **BOX 2**) dentro de uma série de dados escolhida pelo usuário.

BOX 2. Indicators of Compromise (IOC)

Os Indicadores de Comprometimento são artefatos forenses relacionados a uma intrusão computacional que podem ser identificados em um host ou na rede. Estes indicadores podem representar qualquer evento computacional suspeito, como um tráfego de rede anômalo, a criação de uma chave de registro ou a instalação de um programa indesejado no sistema.



Figura 1. Tela inicial do Redline

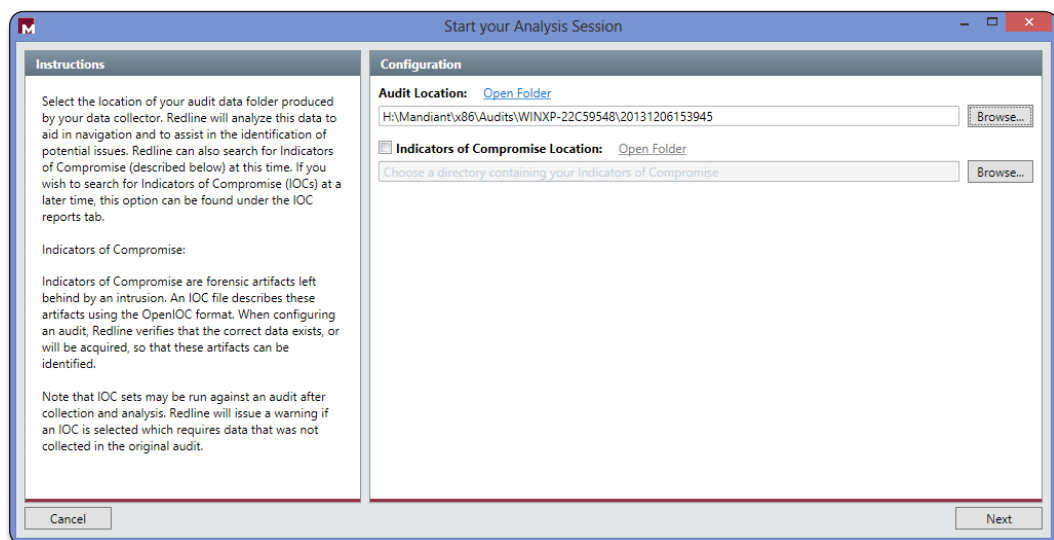


Figura 2. Buscando o local dos arquivos referentes à auditoria feita pelo Memoryze

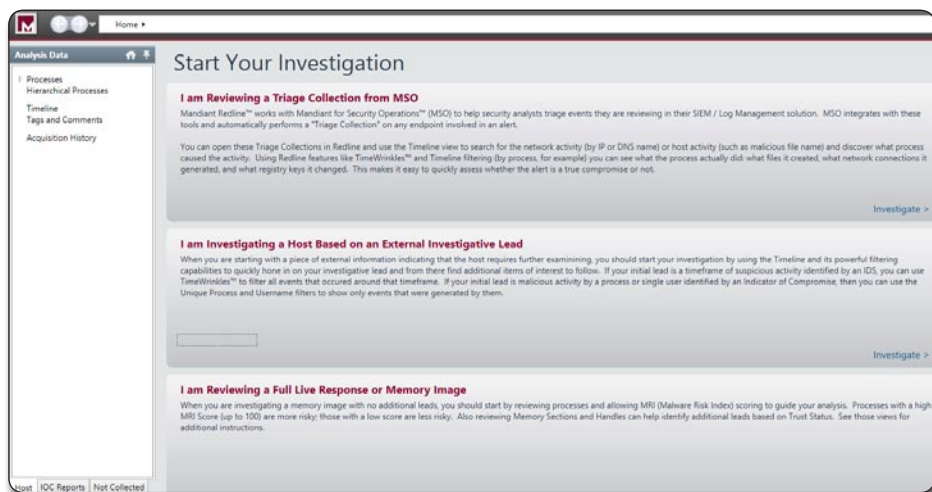


Figura 3. Opções de investigação fornecidas pelo Redline

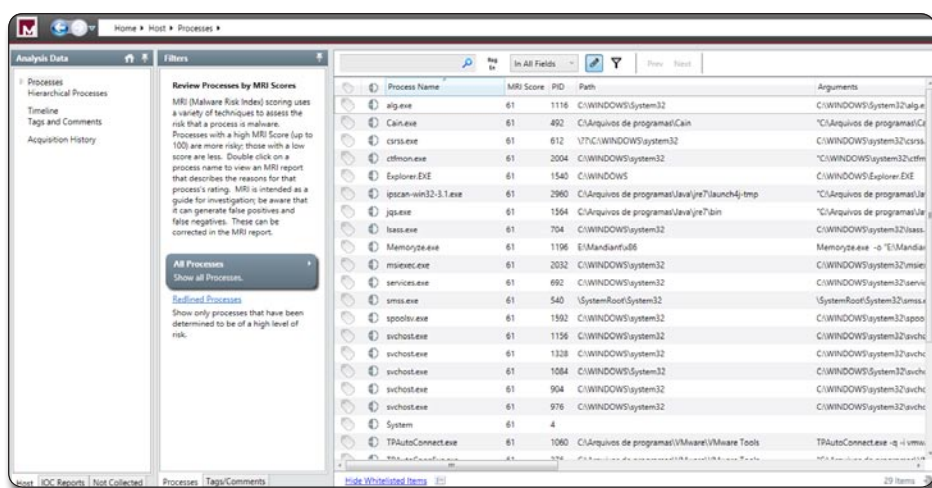


Figura 4. Relatório apresentado pelo Redline referente aos processos em execução

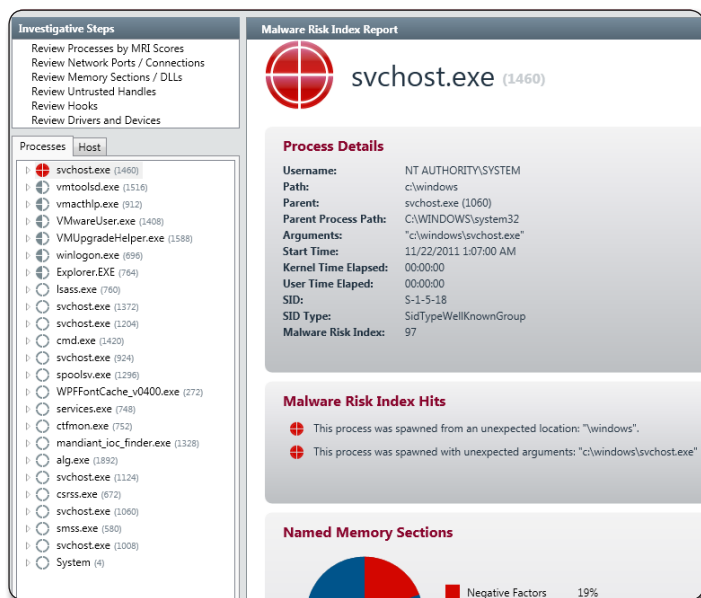


Figura 5. Exemplo de relatório com um processo suspeito – svchost.exe

Como estamos investigando um host externo que teve a memória coletada, devemos utilizar a opção *"I am Investigating a Host Based on an External Investigative Lead"*. Assim, clicando nessa opção, o Redline irá carregar as informações referentes aos processos que coletamos previamente com o Memoryze. Nesse momento veremos que o menu da esquerda poderá ser usado para exibir a hierarquia de processos e uma cronologia de utilização dos mesmos, isto é, a ordem em que eles foram executados e sua estruturação, onde podemos ver os relacionamentos entre eles. É possível visualizar a listagem de processos exibida pelo Redline na Figura 4.

Se um processo reconhecidamente malicioso for encontrado (com MRI alto, conforme os algoritmos de cálculo da ferramenta), o Redline irá apontá-lo deixando um indicador vermelho, assinalando a presença de algo indesejado no sistema. A Figura 5 exibe um exemplo de processo *svchost.exe* suspeito, pois está sendo executado de um diretório diferente do seu padrão e com parâmetros incomuns.

Analisando os processos abertos é possível detectar dois processos incomuns: *Cain.exe* e *ipscan-win32-3.1.exe*. Embora a ferramenta não os tenha considerados nocivos, uma breve pesquisa na Internet os identifica como os executáveis de duas ferramentas:

- **Cain&Abel**, software que possui diversas funções para quebra de senha, oferecendo também funções de falsificação de pacotes que permitem a coleta de informações sensíveis que estiverem passando pela rede;
- **Angry Ip Scanner**, um scanner de rede que visa encontrar portas abertas e identificar os serviços de rede disponíveis dentro de uma rede ou host.

Ambas as ferramentas possuem capacidade de gerar um grande volume de tráfego na rede, resultando na degradação da performance desta e possíveis alertas das ferramentas de segurança existentes.

Com essa informação podemos descartar a infecção por um worm na rede, entretanto devemos nos atentar para estas ferramentas e avaliar o porquê da

presença delas neste computador, documentando os resultados para mantermos o processo de resposta a incidentes devidamente atualizado com o registro de todos os incidentes de segurança.

Conclusões

Neste artigo, pudemos entender o processo de instalação e o funcionamento básico de duas ferramentas gratuitas que podem auxiliar em investigações de *malwares* e qualquer código ou aplicativo malicioso que esteja sendo executado (desta forma, utilizando a memória do recurso alvo), além de ajudar em manutenções preventivas ao fornecer um relatório com detalhes do uso de sua memória para que, com base neste, um usuário possa tomar medidas relacionadas aos arquivos e aplicativos que estejam requerendo muita memória, melhorando a performance do computador.

O Redline e o Memoryze são ferramentas simples e úteis, não tendo tantas opções e recursos como muitas outras do mercado, mas que fornecem bom desempenho nas tarefas que se comprometem a realizar. Dominar ambas as ferramentas irá agregar muito para o arsenal de um perito computacional, devido à facilidade e versatilidade destes aplicativos.

Como vimos neste artigo, a execução destes aplicativos é bastante simples e não necessita de conhecimentos avançados em

informática ou perícia forense computacional para o uso de suas funcionalidades. Entretanto, seu uso não é uma bala de prata. Mesmo com diversas assinaturas para detecção de *malwares*, algo pode passar por este filtro e um pente fino deve ser feito para identificar anomalias e violações às políticas de segurança.

Durante uma resposta a incidentes em sistemas vivos, isto é, ligados e em operação dentro de um ambiente computacional, uma das informações mais críticas e voláteis se encontra na memória RAM. O dueto formado pelo Memoryze e pelo Redline viabiliza a coleta dessas informações de forma simples, permitindo capturar e analisar os detalhes do que estava acontecendo de fato no computador, possibilitando assim uma visão nítida dos processos para que a empresa consiga desenvolver uma melhor compreensão dos seus incidentes de segurança.

Links:

Download da Ferramenta Memoryze

<http://www.mandiant.com/resources/download/memoryze>

Download da Ferramenta Redline

<http://www.mandiant.com/resources/download/redline>

Manual da Ferramenta Memoryze

<https://dl.mandiant.com/EE/library/MemoryzeUserGuide.pdf>

Verificação dos Processos do Windows – File Inspect Library

<http://www.fileinspect.com/>

Artigo sobre Rootkit

<http://www.linhadefensiva.org/2005/03/rootkit>

TecMundo - O que são processos de um sistema operacional e por que é importante saber

<http://www.tecmundo.com.br/memoria/3197-o-que-sao-processos-de-um-sistema-operacional-e-por-que-e-importante-saber.htm>

Analisando volcados de memória: MANDIANT Memorize - NEOFITO

<http://neosysforensics.blogspot.com.br/2008/12/analizando-volcados-de-memoria-mandiant.html>

Manual da Ferramenta Redline

https://dl.mandiant.com/EE/library/Redline1.10_UserGuide.pdf

Cain&Abel – Sniffer de Rede e Quebra de Senhas

<http://www.oxid.it/cain.html>

Angry IP Scanner – Scanner de Portas de Rede

<http://angryip.org/w/Home>

Autor



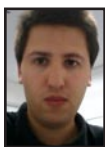
Marcelo Lau

marcelo.lau@datasecurity.com.br

Engenheiro, pós-graduado em Administração e Comunicação e Arte. Possui um mestrado pela Universidade de São Paulo e possui experiência em Segurança da Informação e Computação Forense em diversos grandes bancos do Brasil. Atualmente é o proprietário e diretor executivo da Data Security no Brasil. Leciona aulas como professor em diversas universidades do Brasil e cursos em países da América Latina, como Argentina, Bolívia, Colômbia, Paraguai e Peru.



Autor



Nichols Jasper

nichols.jasper@datasecurity.com.br

É especialista em segurança da informação pós graduado pelo IBTA-SP. Conta com experiência em serviços de consultoria nas áreas de forense computacional e investigação de incidentes de segurança, envolvendo casos de fraude corporativa com violação de propriedade intelectual e modificação fraudulenta de registros eletrônicos.



Honeypots: Evitando invasões

Conheça a ferramenta para análise e compreensão de ataques

No ramo militar, a busca de conhecimento sobre o inimigo é um importante passo para o estabelecimento das defesas contra o mesmo, seja por meio da descoberta de seus métodos de ataque, ferramentas, táticas ou objetivos. Essa mesma ideia se aplica facilmente à segurança de rede. É importante que as defesas contra as ameaças que rondam as redes das organizações sejam construídas a partir do conhecimento das motivações e técnicas de ataque adotadas pelos invasores.

Em um cenário real são empregados diversos meios para proteger a organização contra eventuais ataques. Tais métodos incluem o desenvolvimento de políticas de segurança da informação, o treinamento e conscientização de usuários e a implantação de sistemas de segurança como Firewalls, Sistemas de Prevenção de Intrusão (IPS), Redes Privadas Virtuais, entre outros. Essa situação claramente favorece a proteção do ambiente contra ataques, mas também pode limitar a geração de informação sobre os invasores, pois os métodos citados anteriormente são usados com o intuito de impedir ou reduzir os ataques e não gerar conhecimento a partir deles. As escassas informações sobre os ataques são, na maioria das vezes, restritas à análise de registros de eventos gerados pelos sistemas de segurança.

Para investigar de maneira mais detalhada os ataques que acontecem em uma rede, assim como as estratégias empregadas desde o início até o final do mesmo, é necessária a criação de um ambiente de monitoramento onde as invasões ocorram naturalmente, sem o uso de sistemas que bloqueiam ataques. Os sistemas conhecidos como honeypots (ou potes de mel) foram criados com esse intuito.

Em linhas gerais, um honeypot (ver **BOX 1**) é um recurso computacional de segurança dedicado a ser sondado, atacado ou comprometido. A ideia consiste em desenvolver um sistema que se mostre vulnerável para atrair e monitorar os invasores conectados.

Fique por dentro

Neste artigo serão abordadas as principais características dos honeypots. Serão apresentados: um breve histórico sobre o uso de honeypots para o conhecimento das técnicas e ferramentas usadas por invasores de rede, principais características de um honeypot, classificação de um honeypot de acordo com o nível de envolvimento do invasor com o sistema e definição de honeynet. Também será introduzida a ferramenta Honeyd, que é usada com frequência para a criação de honeypots. Por fim, serão mostrados três exemplos de projetos bem sucedidos que usam honeypots para melhorar a compreensão acerca das ameaças digitais.

BOX 1. Honeypot

O termo Honeypot foi adotado como uma gíria durante a Segunda Guerra Mundial para descrever as mulheres bonitas usadas como isca para revelar segredos militares de inimigos. No final da década de 90, o termo passou a ser empregado por profissionais de segurança que desenvolviam métodos para tentar descobrir a identidade dos invasores e desviá-los do foco principal dos ataques.

Aos olhos dos invasores esse sistema vulnerável não possui nenhuma diferença quando comparado a um sistema de verdade, criando assim uma espécie de armadilha para os criminosos virtuais.

Os principais objetivos de um honeypot são o estudo e compreensão do perfil dos ataques, a detecção de tendências relativas às vulnerabilidades de segurança mais exploradas e, em certos casos, a prevenção de ataques. Neste artigo será mostrado um breve histórico sobre o desenvolvimento de honeypots, o funcionamento básico e os requisitos, os diferentes tipos e algumas ferramentas utilizadas para a implementação de honeypots. Além disso, serão apresentados exemplos de projetos de análise e monitoração de ataques em funcionamento no Brasil e no mundo.

Breve história sobre Honeypots

Em agosto de 1986, Clifford Stoll, um dos administradores da rede de computadores do Lawrence Berkeley Laboratory

(LBL), um renomado centro de pesquisas dos Estados Unidos, vinculado ao Departamento de Energia e localizado em São Francisco, na Califórnia, identificou tentativas de ataque com sucesso no perímetro da rede do laboratório. Contudo, ao invés de bloquear os ataques e consequentemente manter o invasor longe dos sistemas, Stoll decidiu deixá-lo agir e ao longo dos dez meses seguintes monitorar as suas ações. Esse incidente, levado ao público em dezembro de 1989 no livro escrito por Stoll, é tido como o primeiro relato da implantação de mecanismos para acompanhar as atividades de um invasor em uma rede de comunicações.

Em 1991, Bill Cheswick, após implementar um dos gateways da AT&T, resolveu desenvolver ferramentas para monitorar serviços externos acessíveis por esse gateway, como FTP e Telnet. A principal motivação de Cheswick era descobrir a existência de invasores na rede da empresa e, em caso positivo, aprender sobre os métodos de ataque. Nascia então a ideia inicial dos honeypots. Em meados de 1991 Cheswick detectou uma tentativa de acesso não autorizado à rede da AT&T a partir de uma vulnerabilidade no serviço sendmail. Esse evento foi descrito com detalhes em um artigo publicado em 1992, chamado “An Evening with Berferd In Which a Cracker is Lured, Endured, and Studied”. A principal diferença do evento descrito por Cheswick para o evento ocorrido em Berkeley foi a preparação das máquinas. Stoll simplesmente estudou o incidente até encontrar a identidade do invasor, enquanto que Cheswick preparou um determinado conjunto de máquinas a espera de eventuais ataques.

Em 1998, outro passo para a criação de ambientes de monitoração de ataques surgiu com uma ferramenta desenvolvida por outro grande nome da segurança da informação, Fred Cohen. Tal ferramenta, denominada The Deception ToolKit (TDK), permite simular, no sistema operacional em que for instalada, a existência de um grande número de softwares com vulnerabilidades conhecidas. Basicamente, o seu funcionamento consiste em reproduzir saídas em respostas às tentativas de invasão dos atacantes, fazendo-os acreditar que estão explorando as vulnerabilidades e obtendo acesso ao sistema. Assim, surgia a primeira ferramenta cujo objetivo principal era obter informações a respeito das técnicas empregadas para a invasão de sistemas.

A definição clara sobre os conceitos, arquitetura e diferentes propósitos de um Honeypot surgiu somente em 2002, no livro Honeypots: Tracking Hackers, escrito por Lance Spitzner. Atualmente, projetos como o Project Honeypot e HoneyNet Project são desenvolvidos em larga escala pela comunidade de segurança, com o intuito de não somente melhorar o conhecimento acerca das técnicas que envolvem os ataques, mas também para compartilhar as lições aprendidas.

Honeypots e Honeynets

Um honeypot pode ser definido como um recurso computacional de segurança dedicado a ser sondado, atacado ou comprometido e pode ser formado por diversos equipamentos como computadores, roteadores e modems. O fato de estarem

dedicados exclusivamente à captura de ataques implica que todo o tráfego gerado para as máquinas que compõem o honeypot é considerado, potencialmente, um ataque e deve ser analisado.

Em uma rede de comunicação, a distinção entre tráfego malicioso e não malicioso envolve a implementação de políticas de segurança e é fundamental para auxiliar o bloqueio de ataques. Conforme discutido anteriormente, em honeypots essa distinção não é necessária, ou seja, o ambiente simplesmente captura os ataques que lhe são direcionados, facilitando a análise destes. Entretanto, podem-se citar duas desvantagens. A primeira, diz respeito à sua capacidade limitada de visão, ou seja, ele só captura os dados direcionados a ele mesmo. Não é capaz de reconhecer, por exemplo, ataques a um servidor web, ou a um servidor de dados, se este tráfego não for direcionado a ele. A segunda desvantagem refere-se ao risco de o invasor utilizar honeypots para atacar outros sistemas.

Os honeypots podem ser classificados de acordo com a função desempenhada na organização: pesquisa ou produção. Honeypots de pesquisa são utilizados essencialmente para a captura e disseminação de informações. Universidades e institutos de pesquisa são alguns exemplos de instituições que dedicam esforços para a construção de honeypots de pesquisa. Já os honeypots de produção atuam com o propósito de fornecer um nível adicional de segurança às organizações, agindo principalmente na melhoria da detecção e reação aos ataques. Os itens a seguir mostram como a detecção e a reação a ataques podem ser abordadas em um honeypot de produção:

- **Deteção:** em um ambiente real, a detecção de ataques pode ser comprometida devido ao alto tráfego de rede que é formado por tráfego malicioso e não malicioso. Em um honeypot todo o tráfego é considerado como um possível ataque, reduzindo de forma significativa o esforço despendido para a detecção de atividades maliciosas;
- **Reação:** Em um ambiente real, a retirada de uma máquina da rede para um estudo detalhado sobre o ataque pode ser uma tarefa complexa se o computador em questão for vital para as operações da organização. Em um honeypot isso pode ser feito sem maiores problemas, aumentando as chances de obtenção de provas concretas sobre a invasão e detalhes acerca do ataque realizado.

Outra forma usualmente empregada para classificar os honeypots está relacionada ao grau de interação do invasor com o sistema em questão. São três os níveis possíveis de envolvimento: baixo, médio e alto (ver **BOX 2**).

BOX 2. Grau de envolvimento com os honeypots

A literatura possui uma pequena divergência sobre a classificação de honeypots de acordo com o grau de interação do invasor com o sistema. Em alguns casos somente dois níveis (alto e baixo) são usados para representar os honeypots. Ao longo desse artigo adotaremos a classificação em três níveis proposta por Lance Spitzner.

O nível de envolvimento baixo equivale à interação do invasor com serviços falsos gerados por um determinado software. Nesse tipo de envolvimento o invasor fica restrito à interação simulada pelo sistema em questão, e, conseqüentemente poucos dados sobre o ataque podem ser obtidos. Um exemplo comum é o emprego da ferramenta Netcat para redirecionar todas as requisições feitas na porta 80 para um arquivo de log comum, a partir do comando: `nc -l -p 80 > /log/acesso.log`. Os serviços falsos criados pela ferramenta The Deception Toolkit (TDT), citada anteriormente, constituem outro exemplo de honeypot de baixa interatividade.

A interação entre o atacante e o honeypot aumenta no nível de envolvimento médio, o que não significa que ele terá acesso a um sistema real. Contudo, da mesma forma que aumenta o nível de interação, já que agora o invasor possui acesso a um sistema operacional, os riscos de comprometimento do honeypot também aumentam, o que exige um maior cuidado com relação às ferramentas e scripts que interagem com o invasor. O invasor poderia, por exemplo, usar os serviços disponibilizados para disparar ataques contra outros computadores. Uma ferramenta amplamente utilizada pela comunidade de segurança para auxiliar a criação de honeypots de média interatividade é o Honeyd. Esta ferramenta permite a criação de máquinas virtuais em uma rede em que podem ser configurados diversos serviços como FTP, SSH, Telnet, DNS e etc. Na próxima seção será apresentada uma breve introdução ao Honeyd.

Nos honeypots de alta interatividade os atacantes interagem com sistemas operacionais, aplicações e serviços reais. Tais honeypots equivalem, por exemplo, a uma rede física composta por computadores reais em que os dispositivos são configurados para receberem ataques. Como os serviços e sistemas envolvidos são reais, é possível registrar todos os passos do invasor e obter praticamente todos os dados de um determinado ataque. Contudo, os desafios relacionados para que um invasor não utilize essa infraestrutura real para a realização de novos ataques torna esse tipo de honeypot o mais complexo de ser implementado. A **Tabela 1** ilustra as principais diferenças entre os honeypots de acordo com o grau de interatividade.

Um exemplo de honeypot de alta interatividade é conhecido como Honeynet. Uma Honeynet é um conjunto de honeypots, ou seja, é uma rede projetada especificamente para ser comprometida.

As honeynets podem ser divididas em clássicas (reais) ou virtuais. As clássicas são construídas usando sistemas reais (físicos) em cada um de seus honeypots. Em geral, devido aos custos envolvidos e a dificuldade de administração, este tipo não é usado com frequência. As Honeynets virtuais, por sua vez, são construídas usando ferramentas de virtualização.

A complexidade de uma honeynet exige que diversos cuidados sejam tomados para diminuir os riscos de falha no projeto. Dois requisitos, em especial, são considerados críticos: controle e captura dos dados.

Como o honeypot é um dispositivo dedicado a ser atacado, a ocorrência de invasões fora do perímetro delimitado é um risco real que deve ser minimizado. Deve-se garantir, principalmente, que, uma vez comprometido, um honeypot, não possa ser usado para danificar qualquer outro sistema. Outro desafio é controlar o fluxo de dados sem levantar suspeitas nos invasores.

Após o comprometimento do honeypot, os invasores tentam abrir conexões externas, seja para instalar malwares ou conectar-se em canais IRC com o intuito de formar botnets. Assim, é preciso dar a eles permissões suficientes para executar tais tarefas, pois a desconfiança sobre a veracidade do sistema comprometido pode levar os invasores a abandonarem o honeypot. Em linhas gerais, quanto maior a flexibilidade sobre o sistema, mais se consegue aprender com eles, contudo, o risco também aumenta. Esse controle é frequentemente implementado com o auxílio de scripts para o tratamento das permissões sobre o sistema e também com devidas intervenções manuais.

O outro requisito é relacionado à captura das informações geradas pelo invasor. Tais informações serão analisadas para possibilitar o aprendizado das ferramentas, técnicas e motivações usadas pelos atacantes. O grande desafio é a captura do maior número possível de dados sem deixar que o invasor perceba. Isso é feito utilizando pequenas modificações nos honeypots como, por exemplo, o uso de ferramentas que identificam os comandos digitados pelos invasores (*keyloggers*). É essencial que as informações coletadas sejam armazenadas em um servidor externo, pois, caso contrário, elas podem ser encontradas e apagadas pelo atacante.

A **Figura 1** ilustra a arquitetura típica de uma Honeynet. O firewall separa a rede em três sub-redes distintas: a rede da organização, a Honeynet e a Internet. Qualquer pacote que entrar

Características	Honeypot de baixa interatividade	Honeypot de média interatividade	Honeypot de alta interatividade
Instalação	fácil	média/difícil	difícil
Manutenção	fácil	média	trabalhosa
Risco de comprometimento	baixo	baixo	alto
Obtenção de informações	muito limitada	limitada	extensiva
Necessidade de mecanismos de contenção	não	sim	sim
Atacante tem acesso ao S.O. real	não	não	sim
Aplicações e serviços oferecidos	emulados	emulados	reais
Atacante pode comprometer o honeypot	não (em teoria)	não (em teoria)	sim

Tabela 1. Diferenças entre honeypots de acordo com o grau de interatividade

ou sair da Honeynet terá que passar tanto pelo firewall quanto pelo roteador. Nessa arquitetura o firewall também é importante para permitir todas as conexões de entrada na Honeynet e também as conexões de saída até um limite estipulado. Isto dá ao invasor a flexibilidade necessária para executar as operações desejadas sem permitir o abuso.

A função do roteador é esconder o firewall no caso de um eventual comprometimento do honeypot, com o intuito de criar um cenário mais realista para os invasores. Outra função desse posicionamento é usar um segundo ponto de controle de acesso, auxiliando o firewall a evitar ataques provenientes das estações da Honeynet. O sistema de detecção de intrusão (IDS) captura toda a atividade da rede e emite alertas quando seus sensores detectarem atividades suspeitas. Por fim, os dados coletados pelo IDS são armazenados com segurança no servidor de informações.

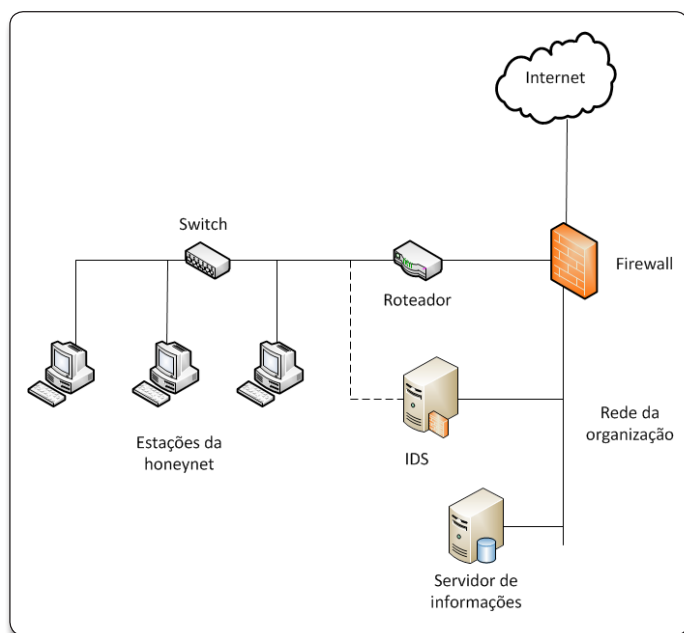


Figura 1. Componentes básicos de uma honeynet.

Construção de honeypots usando o Honeyd

Na seção anterior os honeypots foram classificados de acordo com o grau de interação do invasor com o sistema. Três tipos foram definidos: baixa, média e alta interatividade. Os honeypots de baixa interatividade são mais simples e a obtenção de informação é muito limitada. Já os honeypots de alta interatividade podem fornecer uma extensiva quantidade de informação, porém são mais complexos tanto na instalação quanto na manutenção. Assim, os honeypots de média interação aparecem como uma alternativa interessante para os profissionais da área de segurança que procuram um meio termo entre a complexidade de gerência e implantação dos honeypots de alta interatividade e a limitação das informações obtidas com os honeypots de baixa interatividade. Este fato motivou o desenvolvimento de ferramentas para auxiliar a criação de honeypots de média interação. O software Honeyd,

em especial, ganhou notoriedade na comunidade de segurança devido à flexibilidade em ser adaptado para diversas estruturas e plataformas.

O Honeyd é uma ferramenta desenvolvida e mantida por Niels Provos desde abril de 2002 para criar hosts virtuais em uma rede. Mesmo sendo uma ferramenta distribuída de forma livre sob a licença GNU e mantida somente nas horas vagas por Provos, ela tem sido amplamente utilizada na construção de honeypots (comerciais ou não) em todo o mundo. O software funciona em sistemas BSD, GNU/Linux e Solaris e pode criar máquinas virtuais de diferentes sistemas operacionais executando diferentes serviços da pilha TCP/IP, como Telnet, DNS, SSH, entre outros. O Honeyd também é capaz de simular estruturas de redes completas, configurar os protocolos de roteamento e calcular métricas como perda de pacotes e latência.

A interação do Honeyd com os invasores é limitada apenas ao nível de rede. Por isso, ao invés de simular todos os aspectos de um sistema operacional, como um ambiente de virtualização completo (Virtual Box ou VMware, por exemplo), ele simula somente a pilha TCP/IP do sistema operacional alvo. A arquitetura do Honeyd consiste nos seguintes componentes: base de dados de configuração e personalidade, central expedidora de pacotes, motor de personalidade, manipulador dos protocolos de rede e

Não perca tempo reinventando a roda!

COBREBEMX

Componente completo para sua Cobrança por Boleto Bancário e Débito em Conta Corrente

Mais de 40 exemplos em diversas linguagens de programação

Geração e leitura de arquivos (remessa e retorno) nos padrões FEBRABAN e CNAB

Testes e Downloads gratuitos em nosso site

ACESSE E CONHEÇA O COMPONENTE EM:
WWW.COBBREBEM.COM

um componente opcional chamado roteamento, usado somente quando o Honeyd faz modificações na topologia da rede simulada. A **Figura 2** ilustra a arquitetura e os respectivos relacionamentos entre si.

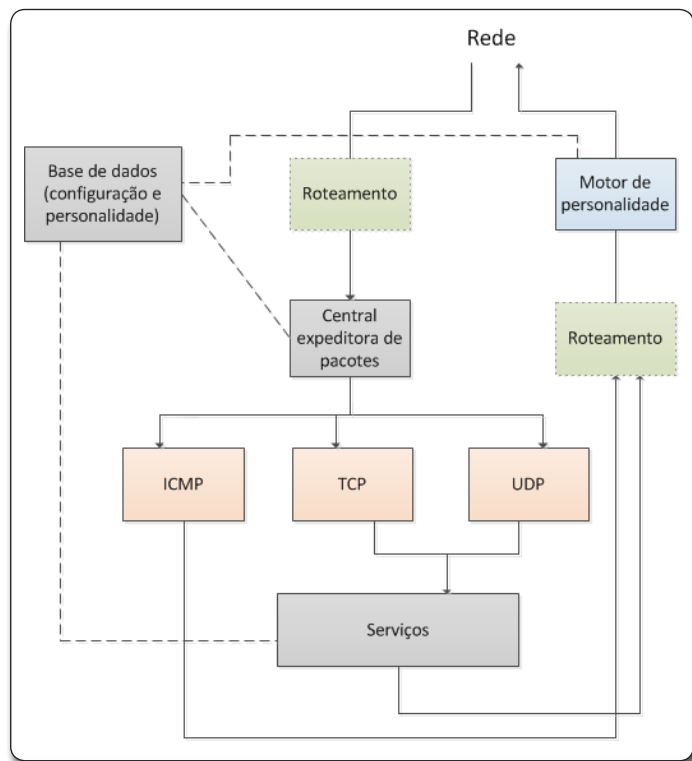


Figura 2. Arquitetura do Honeyd

Com base na arquitetura apresentada na **Figura 2**, vamos analisar o funcionamento básico da troca de pacotes entre o Honeyd e um invasor. Suponha que um dispositivo virtual foi criado e está pronto para ser atacado. Primeiramente, os pacotes de rede que chegam até o computador com o Honeyd instalado são processados pela central expedidora de pacotes. Somente pacotes TCP, UDP e ICMP são suportados. Outros tipos de pacotes são descartados. O próximo passo da comunicação consiste em encontrar o dispositivo virtual que corresponde ao destino do endereço IP. Se nenhuma configuração foi definida, um modelo padrão para o dispositivo é utilizado (uma máquina Linux, por exemplo). Se a configuração foi previamente definida, os pacotes enviados pelo invasor, assim como a configuração correspondente do dispositivo virtual, são entregues ao manipulador do protocolo de rede (TCP, UDP e ICMP).

Para pacotes ICMP, o Honeyd responde a pedidos de echo ou envia mensagens de destino inatingível de acordo com a disponibilidade do dispositivo na rede (ver **BOX 3**). Já para TCP e UDP as conexões são estabelecidas de acordo com os serviços que foram definidos no arquivo de configuração *honeyd.conf*. Os serviços são emulados com o auxílio de scripts que implementam o modo com que as requisições solicitadas pelo invasor

devem ser respondidas pelo dispositivo. As **Listagens 1, 2 e 3** ilustram trechos do arquivo *honeyd.conf* usado como um ponto único de configuração dos diferentes dispositivos virtuais. Esse arquivo contém a configuração de cada dispositivo, assim como o comportamento de cada um dos serviços que foram configurados. Já os scripts podem ser desenvolvidos ou adquiridos no próprio site da ferramenta.

BOX 3. Pacotes ICMP

O ICMP – Internet Control Message Protocol – é um protocolo que faz parte da pilha TCP/IP, enquadrando-se na camada de rede (nível 3), a mesma camada do protocolo IP – Internet Protocol. O seu uso mais comum é feito pelo utilitário ping. O ping envia pacotes ICMP para verificar se um determinado dispositivo está disponível na rede usando um pedido de ECHO (ECHO request). Se o dispositivo não está disponível o ICMP retorna mensagens de destino inatingível (Destination Unreachable).

Listagem 1. Exemplo de configuração de um sistema AIX respondendo na porta 80.

```
### Exemplo de uma configuração simulando um sistema AIX.  
create template  
set template personality "AIX 4.0 – 4.2"  
add template tcpt port 80 "sh scripts/web.sh"
```

Listagem 2. Exemplo de configuração de um Windows NT respondendo na porta 80.

```
### Exemplo de uma configuração simulando um sistema Windows NT.  
create windows  
set windows personality "windos NT 4.0 Server SP5-SP6"  
add windows tcp port 80 "perl scripts/iis-0.95/iisemul8.pl"
```

Listagem 3. Exemplo de configuração de roteador Cisco respondendo na porta 23.

```
### Exemplo de uma configuração simulando um roteador Cisco.  
create router  
set router personality "Cisco IOS 11.3 – 12.0(11)"  
add router tcp port 23 "/usr/bin/perl scripts/router-telnet.pl"
```

Por fim, antes de um pacote ser enviado de volta ao invasor, ele é processado pelo motor de personalidade, que ajusta o conteúdo do pacote para que pareça originado pela pilha TCP/IP referente ao sistema operacional configurado. Assim, a veracidade sobre o ambiente de rede em questão pode ser confirmada pelo atacante.

Manuais de instalação, arquivos de instalação e exemplos de configurações podem ser encontrados na página do projeto (ver seção **Links**).

Exemplos de projetos bem sucedidos

Atualmente existem diversos projetos de sucesso que envolvem o uso de honeypots para a coleta e disseminação de conhecimento sobre ataques. A seguir, três importantes projetos serão apresentados: Honeynet Project, Project Honeypot e Projeto de Honeypots Distribuídos.

O Honeynet Project é uma organização internacional de pesquisa em segurança sem fins lucrativos, focada em investigar os mais recentes ataques e desenvolver ferramentas de código aberto para melhorar a segurança da Internet. Com o auxílio

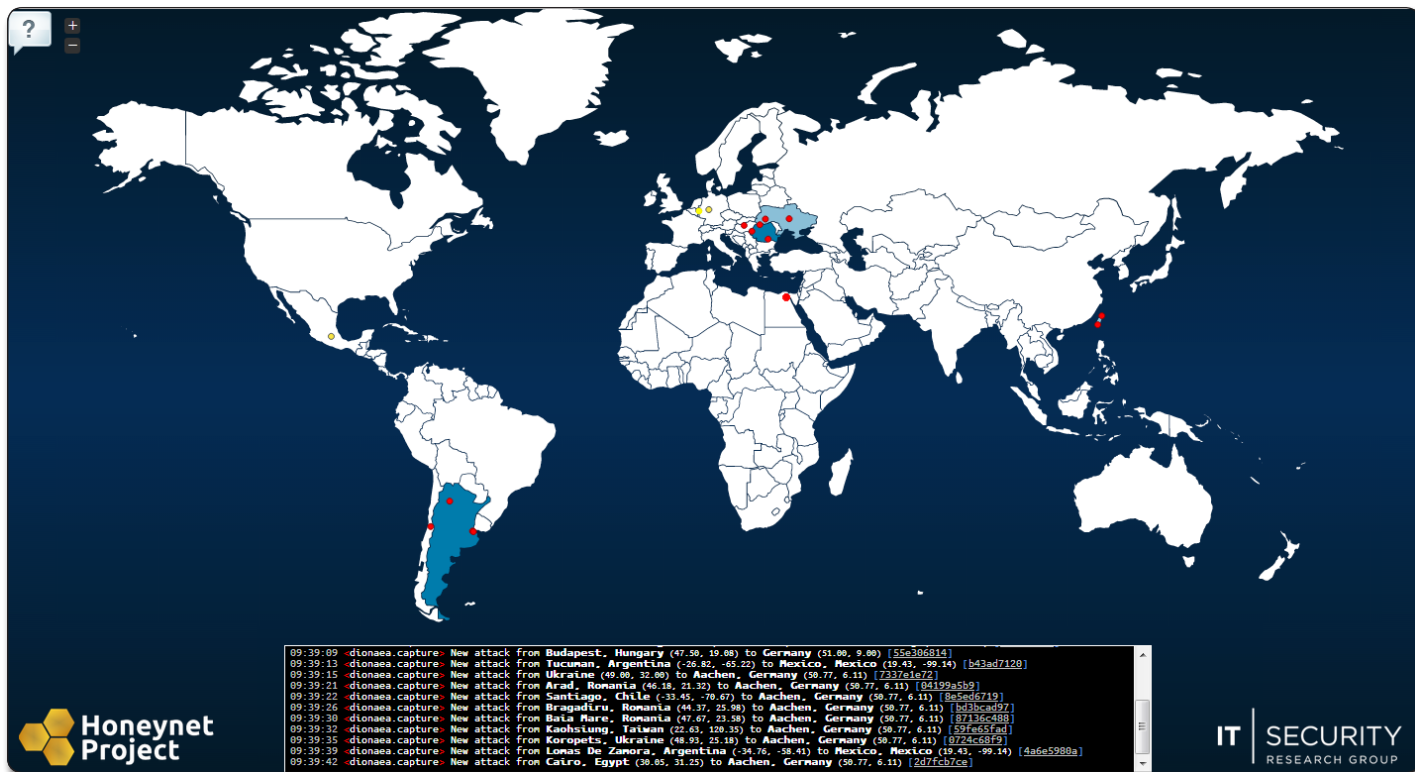


Figura 3. Interface do HoneyMap mostrando a origem dos ataques direcionados aos honeypots do Honeynet Project

de voluntários, a organização implanta honeypots em todo o mundo e compartilha os resultados obtidos com toda a comunidade de segurança.

Além da implantação de honeypots, outro objetivo importante da organização é o desenvolvimento de ferramentas que auxiliam a construção de honeypots. Alguns exemplos de ferramentas incluem:

- **Cuckoo** - ferramenta para análise de códigos maliciosos usados pelos invasores;
- **Glastopf** - simula um servidor web com diversas vulnerabilidades;
- **Google Hack Honeypot** - identifica invasores que usam as páginas do Google como ferramenta para seus ataques;
- **HFlow2** - auxilia a coleta das informações de ataque a um honeypot;
- **HoneyMap** - mostra um mapa em tempo real com a origem e o destino de ataques usando dados de honeypots vinculados ao Honeynet Project. A Figura 3 ilustra o mapa de ataques. No mapa mundi é possível visualizar a origem dos invasores, e no quadro situado logo abaixo do mapa pode ser encontrado o honeypot de destino do respectivo ataque;
- **Kippo** - monitora os ataques de força bruta no serviço SSH e também toda a interação do atacante com a interface shell.

Desde 2010 o Honeynet Project também propõe pequenos desafios de segurança para a comunidade científica. Os desafios geralmente envolvem situações de ataque reais e os participantes

são encorajados a submeter não somente a solução do desafio, como também as ferramentas usadas para chegar até ela.

Já o Project Honey Pot usa uma infraestrutura de honeypots para identificar spammers e spambots (ver BOX 4). Em linhas gerais, a identificação funciona da seguinte forma: o projeto fornece endereços de e-mail falsos que são inseridos em páginas da Internet de todo o mundo. Esse método é empregado porque grande parte dos spammers utiliza ferramentas automatizadas que varrem a Internet em busca de endereços de e-mail. Os endereços de e-mail falsos são então usados como uma espécie de isca para atrair e monitorar os spammers, revelando informações importantes como endereços IP e o exato momento em que o endereço falso começou a receber e-mails.

BOX 4. Spammers e Spambots

Spammer é o termo usado para identificar aquele que envia spams. Spambot é um software criado para auxiliar no envio de spam. Spambots normalmente criam contas de e-mail falsas e enviam spam através delas. Alguns Spambots, entretanto, podem ser configurados para quebrar senhas e enviar spam usando contas de outras pessoas.

Este projeto foi desenvolvido pela companhia norte-americana Unspam Technologies e possui diversos parceiros. De acordo com a companhia, qualquer usuário que hospeda páginas na Internet pode inserir os endereços de e-mail falsos em suas páginas. A empresa garante que a inserção de tal endereço não irá aumentar o tráfego ou carregar a página. Em contra partida, a companhia

fornece todos os dados coletados em sua página e também trabalha diretamente com as autoridades para identificar os spammers e tomar as devidas providências legais.

No site do Project Honey Pot também é possível visualizar estatísticas sobre spammers e spambots ao redor do mundo. A **Tabela 2** sumariza o “Top 5” dos países que mais vasculham endereços de e-mail pela Internet, que mais enviam spam e que mais usam spambots configurados para quebrar senhas e enviar spam usando contas de outras pessoas. Curiosamente, os dados mostram o Brasil entre os países que mais enviam spam e que mais usam spambots configurados para quebrar senhas, o que indica que novas formas de orientação e prevenção sobre Spam precisam ser desenvolvidas aqui.

Países que vasculham endereços de e-mail	Países que enviam Spam	Países que empregam ataques para quebrar senhas de e-mail
1) China	1) China	1) Índia
2) Espanha	2) Brasil	2) Brasil
3) Estados Unidos	3) Estados Unidos	3) Rússia
4) Romênia	4) Alemanha	4) Estados Unidos
5) Alemanha	5) Rússia	5) China

Tabela 2. Estatísticas fornecidas pelo HoneyPot Project sobre dados de Spam em todo o mundo

Por fim, o Brasil também possui um projeto próprio de Honeypots, denominado Projeto de Honeypots Distribuídos. Este é mantido pelo Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT.br) e tem como objetivo aumentar a capacidade de detecção de incidentes, correlação de eventos e determinação de tendências de ataques envolvendo o Brasil.

O projeto possui uma rede distribuída de honeypots de média interatividade (usando o Honeyd) instalada em 24 cidades brasileiras, a maioria localizada no estado de São Paulo. Entre os trabalhos que utilizam essa infraestrutura destacam-se o *Spampots Project*, dedicado para obter novas informações sobre a disseminação de spam e também o *Distributed Honeypots for Attack Trend Analysis*, que investiga novas tendências de ataque e divulga estatísticas sobre o fluxo de rede gerado pelo honeypot.

Conclusões

A coleta de informações sobre invasores com o auxílio de Honeypots fornece uma nova perspectiva sobre as diversas técnicas e ferramentas empregadas nos ataques. O conhecimento gerado por tais dados pode ser usado como uma espécie de alerta inicial aos administradores de rede e assim melhorar a detecção e prevenção de futuros ataques.

Para cumprir com seus objetivos, é preciso que o honeypot seja construído de maneira adequada. Quando configurado de maneira indevida o honeypot pode ser usado para disparar outros ataques, pois os invasores possuem certo nível de liberdade no ambiente monitorado. Uma importante recomendação nesse sentido consiste em liberar todo o tráfego de entrada para o honeypot, porém, o tráfego malicioso deve ser impedido de

chegar à Internet ou nas redes internas da organização. Outra importante recomendação relacionada à implantação de honeypots passa pela definição do grau de interatividade do invasor com o sistema. Quanto maior a interação maior é a possibilidade de detalhar os passos do invasor no sistema, contudo o risco de comprometimento do ambiente também aumenta. A comunidade de pesquisa da área de segurança compartilha informações e fornece várias ferramentas para diminuir os riscos associados à construção de honeypots.

De uma forma geral, a busca por uma rede mais segura exige o conhecimento das técnicas e ferramentas adotadas pelos invasores. Neste contexto, os Honeypots podem trazer inúmeros benefícios para todo ambiente que possui requisitos de segurança.

Autor



Rodrigo Sanches Miani

rodrigomiani@gmail.com

Graduado em Matemática pela Universidade Federal de São Carlos (2005), mestrado em Engenharia Elétrica pela Universidade Estadual de Campinas (2009) e doutorado em Engenharia Elétrica também pela Universidade Estadual de Campinas (2013). Atualmente é pesquisador colaborador da Faculdade de Engenharia Elétrica e Computação na Universidade Estadual de Campinas. Possui experiência de oito anos em pesquisa e desenvolvimento de soluções para governo eletrônico e segurança de redes. Foi professor das disciplinas relacionadas à Segurança da Informação na Faculdade Campo Limpo Paulista – São Paulo (FACCAMP). Seus interesses de pesquisa incluem modelos de quantificação de segurança da informação, métricas de segurança, e estudos empíricos em segurança e criptografia.



Links e Referências:

Honeypots e Honeynets: Definições e Aplicações

<http://www.cert.br/docs/whitepapers/honeypots-honeynets/>

Honeynet Project

<http://www.honeynet.org/>

The Project Honey Pot

<https://www.projecthoneypot.org/index.php>

Site do Honeyd

<http://www.honeyd.org/>

Projeto Honeypots Distribuídos

<http://honeytarg.cert.br/honeypots/index-pa.html>

Artigo “Honeynets: Invasores, Ferramentas, Técnicas e Táticas.” escrito por Eugenia Jabour e Otto Duarte

www.gta.ufrj.br/seminarios/CPE825/tutoriais/eugenia/honeynets.PDF

Dissertação de mestrado “Proposta E Avaliação De Um Modelo Alternativo Baseado Em Honeynet Para Identificação De Ataques E Classificação De Atacantes Na Internet”, escrito por Alexandre Pina Andrucio

www.ravel.ufrj.br/sites/ravel.ufrj.br/files/publicacoes/tese_pinaffi.pdf

Livros

“Virtual Honeypots: From Botnet Tracking to Intrusion Detection”, escrito por Niels Provos e Thorsten Holz. Editora Addison-Wesley, 2008.

“Honeypots: Tracking Hackers”, escrito por Lance Spitzner. Editora Addison-Wesley, 2002.

Python:

Automação de tarefas

Faça o computador trabalhar para você na administração de ambientes de TI

Nos últimos anos acompanhamos o crescimento exponencial dos ambientes de TI, tanto em termos de complexidade quanto de quantidade de equipamentos e aplicações, os quais demandam gerenciamento constante. Existem dois caminhos para lidar com esse crescimento: contratar mais profissionais (ou como acontece muitas vezes, simplesmente sobrecarregar os atuais) ou investir em automação.

Automação de tarefas repetitivas não é nenhuma novidade, e é o motivo pelo qual os computadores foram criados. A questão é que cada vez mais encontramos profissionais que, muitas vezes por falta de conhecimento, se transformam em verdadeiros escravos das aplicações que cuidam, executando diariamente diversas tarefas repetitivas sem imaginar que pode existir uma solução melhor.

Com as iniciativas de Cloud Computing cada vez mais presentes, onde o provisionamento de servidores e aplicações é realizado com poucos cliques, a automação é ainda mais importante. É difícil imaginar como dar suporte a um ambiente com, por exemplo, 1000 servidores virtuais, sem rotinas automatizadas. O AWS (*Amazon Web Services*) e o OpenStack são dois exemplos de plataformas de cloud que fornecem diversos serviços de automação que permitem o uso de Python (esse é um assunto que provavelmente daria um artigo interessante no futuro).

Este artigo se propõe a apresentar uma breve introdução à programação utilizando a linguagem Python, e com base nos conceitos apresentados, expor alguns cenários de automação comuns no dia a dia dos profissionais de TI. A ideia é que estes cenários sirvam de *building blocks* para a construção de soluções específicas para os problemas de cada um. Assumimos como premissa que os leitores possuem ao menos uma noção básica de programação, do tipo que se aprende no colégio, faculdade ou cursos livres.

Fique por dentro

Conhecer uma linguagem de script e internalizar uma cultura de automação permite que se delegue o trabalho repetitivo para quem faz isso melhor (o computador), e libera tempo precioso dos profissionais de TI para se dedicarem a projetos e outras atividades mais complexas.

O objetivo deste artigo é apresentar a linguagem Python como um meio agradável e simples de automatizar tarefas do dia a dia e permitir que os sysadmins produzam mais com menos esforço.

Os leitores podem se perguntar: por que Python? Existem outras ótimas linguagens de script que podem ser utilizadas para automação, como Perl, os diversos dialetos de shell-script, Power Shell, VBScript, para citar algumas. A opção por Python foi direcionada por alguns fatores, baseados em experiência pessoal:

- **Simplicidade de sintaxe:** Perl é uma linguagem excelente e talvez mais poderosa em alguns aspectos, mas iniciantes têm alguma dificuldade em entender a sintaxe e se confundem com a quantidade de símbolos utilizados pela linguagem;
- **Disponibilidade em várias plataformas:** Power Shell e VBScript são nativas do Windows, assim como shell-script é nativo do Unix. A ideia é aprender apenas uma linguagem que possa ser utilizada em diversas plataformas;
- **Disponibilidade de bibliotecas:** Python possui uma coleção enorme de bibliotecas, muitas delas multiplataforma, que economizam muito tempo no desenvolvimento;
- **Uso futuro:** Python é uma linguagem de uso muito abrangente, de automação de rotinas a frameworks completos de desenvolvimento web. Se o leitor quiser investir em outras áreas da programação, é possível aproveitar o conhecimento já adquirido.

Apresentados os fatores, que fique claro que esta não é uma discussão do tipo “linguagem X é melhor do que linguagem Y”. O objetivo aqui é, acima de tudo, apresentar uma linguagem aos que ainda não programam. Para os leitores que já têm a sua linguagem de script favorita, é sempre possível aprender uma

coisa ou outra observando como as tarefas podem ser feitas em outra linguagem.

Introdução à linguagem Python

Python é uma linguagem de script interpretada e multiparadigma (ver **BOX 1**), que foi criada por Guido van Rossum no final da década de 1980, enquanto estava no CWI – *Centrum Wiskunde & Informatica* (Instituto Nacional de Pesquisas Matemáticas e de Ciências da Computação) em Amsterdam, na Holanda.

Desde o seu início, a linguagem foi concebida para ser simples de entender e possuir uma biblioteca padrão bastante abrangente, que possibilitasse alta produtividade por parte do programador. A documentação oficial (veja a seção **Links**) chama essa filosofia de “*batteries included*”, ou seja, a linguagem já vem com as bibliotecas que são mais comuns e mais utilizadas pela comunidade.

BOX 1. Linguagem Interpretada e Multiparadigma

Linguagem interpretada é aquela cujo programa não é compilado para um arquivo executável, mas sim executado linha a linha por um interpretador.

Multiparadigma é uma linguagem que suporta diversos paradigmas de desenvolvimento.

Atualmente temos como paradigmas mais comuns os de programação imperativa, orientada a objetos e funcional.

Python vive um momento de transição, em que foi criada uma nova versão da linguagem (3), que possui diferenças que causaram uma ruptura em algumas funcionalidades da versão anterior (2.7). Na prática é possível utilizar qualquer uma das duas versões, e em termos de produtividade as duas versões são equiparáveis.

Instalação do interpretador e bibliotecas

Todos os exemplos deste artigo foram criados utilizando a versão 3.2 do ActivePython, que é distribuído pela ActiveState (veja a seção **Links**), para Windows, Linux e Mac OSX. É recomendado que se utilize a versão de 32 bits porque é a que possui a maior quantidade de bibliotecas adicionais disponíveis no repositório. Os scripts e exemplos mostrados aqui devem funcionar tanto na versão 2.7 como na 3 do Python sem maiores problemas, independentemente de distribuição.

Modo batch e modo interativo (REPL)

Em Python existem dois modos de acionarmos o interpretador: o modo batch, que é o mais comum e é o que utilizaremos sempre que quisermos executar um programa, e o modo interativo, também chamado de REPL (acrônimo para “Read-Eval-Print Loop”).

No modo batch, para executar um programa, basta invocar o interpretador (assumindo que este esteja no PATH do sistema operacional) e passar o nome do programa:

```
python programa.py
```

Ao executar esse comando, o interpretador Python irá executar as instruções contidas no arquivo *programa.py*.

Se executarmos apenas o interpretador, sem passar nenhum programa, veremos uma saída no prompt de comando semelhante ao exemplo:

```
Python 2.7.5 (default, Aug 21 2013, 21:18:58)
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>>
```

Neste prompt é possível digitar comandos Python e ver os seus resultados, sem precisar escrever um programa completo. Veja alguns exemplos na **Listagem 1**.

Listagem 1. Exemplo do REPL do Python.

```
>>> 2+2
4
>>> print("Hello World")
Hello World
>>> 2+2
4
>>> 3/2
1
>>> 3.0/2
1.5
```

No nosso exemplo é possível encontrar algumas operações aritméticas e um comando de exibição de texto na tela (**print**). O REPL é bastante útil quando queremos realizar um teste simples, com poucas linhas, e não queremos escrever um programa para isso, ou simplesmente quando queremos analisar algum dado de forma interativa, pouco a pouco.

Variáveis e tipos de dados

Variáveis são os elementos fundamentais de qualquer linguagem de programação. Representam uma forma de guardarmos valores em memória para serem utilizados posteriormente durante a execução do nosso programa. Outra forma de pensar no conceito de variáveis é imaginar que estamos nomeando um determinado valor.

Em Python não há um comando especial para declaração de variáveis. Para criarmos uma variável basta atribuímos um valor a um nome, utilizando o sinal de igual (=). Veja o exemplo no REPL:

```
>>> num = 10
>>> num
10
```

Nesse exemplo é possível verificar que foi criada uma variável de nome “num” com valor 10. Ao digitar o nome da variável no REPL, este nos retorna o valor atribuído a “num”.

Em se tratando de valores numéricos, podemos supor que é possível realizar operações aritméticas, como exposto na **Listagem 2**.

Neste exemplo criamos as variáveis “a” e “b”, realizamos algumas operações com os seus valores, e guardamos o resultado de

cada operação em outra variável (na ordem, soma, multiplicação, divisão, exponenciação e subtração).

Todos esses exemplos possuem uma coisa em comum: utilizam números inteiros. Isso nos leva ao assunto dos tipos de dados em Python.

Existem dois grandes grupos de tipos de dados em Python, os escalares, que guardam um valor por vez, e as listas, que guardam vários valores de uma vez.

Os tipos de dados escalares em Python são quatro:

- **int** – números inteiros (1, 10, 100);
- **float** – números de ponto flutuante (1.5, 3.2);
- **str** – string, texto, delimitado por aspas, simples ou duplas;
- **bool** – verdadeiro ou falso (valores **True** ou **False**).

O tipo do dado é determinado dinamicamente pelo interpretador durante a execução, de modo a garantir que não seja realizada nenhuma operação inválida, como demonstra o código da **Listagem 3**.

Listagem 2. Exemplos de operações aritméticas.

```
>>> a = 2
>>> b = 2
>>> a + b
4
>>> c = a + b
>>> c
4
>>> d = a * b
>>> d
4
>>> e = a / b
>>> e
1
>>> f = a ** b
>>> f
4
>>> g = a - b
>>> g
```

Listagem 3. Operações inválidas.

```
>>> nome = 'Mateus'
>>> 10 / nome
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for /: 'int' and 'str'
```

Recebemos um erro ao tentar dividir um número inteiro por uma string, porque são valores incompatíveis para essa operação.

No entanto, é válido operarmos valores inteiros com valores **float**, pelo fato dos dois serem numéricos:

```
>>> 10/2.75
3.6363636363636362
```

No caso de uma operação entre **int** e **float**, o resultado é sempre **float**.

Da mesma forma que não é preciso declarar variáveis, não é preciso indicar o tipo de dado, o interpretador faz esse trabalho.

Os tipos de dados escalares são muito utilizados e estão presentes na maioria das linguagens de programação, com suas particularidades. O que torna Python uma linguagem poderosa são as suas listas, que permitem a manipulação de conjuntos de dados.

Existem quatro tipos de listas em Python:

- **tuple (tupla)** – conjunto de valores imutável;
- **array (matriz)** – conjunto de valores mutável;
- **dict (dicionário)** – tabela de chaves e valores;
- **set (conjunto)** – conjunto de valores distintos.

Os tipos **tuple**, **array**, **set** e **str** (que afinal de contas, é uma lista de caracteres) possuem em comum as formas de acesso que chamamos de *indexing* (acesso por índice) e *slicing* (fatiamento, em uma tradução livre).

Indexing é útil para extrairmos um elemento por vez de uma lista. Veja um exemplo na **Listagem 4**.

Slicing é útil para extrairmos um conjunto de elementos de uma lista. O resultado é uma nova lista ou uma nova string, dependendo do tipo utilizado. Veja um exemplo na **Listagem 5**.

Temos ainda a função **len**, que nos mostra a quantidade de elementos de uma lista, ou a quantidade de caracteres de uma string, como no exemplo da **Listagem 6**.

Listagem 4. Acesso a elementos de uma lista utilizando indexing.

```
>>> x = ['a','b','c'] #lista com os elementos a, b e c
>>> x[0] #retorna o elemento na posicao 0
'a'
>>> x[1] #retorna o elemento na posicao 1
'b'
>>> x[2] #retorna o elemento na posicao 2
'c'
>>> s = "Jose da Silva" #mesma coisa, mas utilizando uma string
>>> s[0]
'J'
>>> s[1]
'o'
```

Listagem 5. Acesso a elementos de uma lista utilizando slicing.

```
>>> x[0:2] #retorna elementos entre 0 e um antes de 2
['a','b']
>>> x[1:] #retorna elementos de 1 até o final
['b','c']
>>> x[:1] #retorna elementos do início até um antes do final
['a','b']
>>> s[0:4] #retorna os caracteres entre 0 e um antes de 4
'Jose'
>>> s[5:7] #retorna os caracteres entre 5 e um antes de 7
'da'
```

Listagem 6. Exemplo da função len().

```
>>> x
['a','b','c']
>>> len(x)
3
>>> s
'Jose da Silva'
>>> len(s)
13
```

Para as strings existem três operações importantes e que serão muito utilizadas nos nossos cenários, que são a **concatenação**, **split** e **join**.

Em Python, é possível concatenar strings de duas formas, como no exemplo da **Listagem 7**.

É possível perceber que o segundo exemplo, apesar de aparentemente mais complicado, tem uma vantagem: o formato da string desejada está todo contido dentro uma única string, e os dados dentro de uma tupla, após o sinal %. Os dois sinais “%s” são placeholders, que indicam que duas strings são esperadas, e com um espaço entre elas.

Agora imagine que queremos juntar cinco strings, em vez de duas, como no exemplo da **Listagem 8**.

Listagem 7. Exemplo de concatenação de strings.

```
#Utilizando o operador +
>>> nome='Jose'
>>> sobrenome='Silva'
>>> nome_completo=nome + ' ' + sobrenome
>>> nome_completo
'Jose Silva'

#Utilizando o operador %:
>>> nome_completo2='%s %s' % (nome, sobrenome)
>>> nome_completo2
'Jose Silva'
```

Listagem 8. Concatenação de strings utilizando %.

```
>>> s1='a'
>>> s2='a'
>>> s3='a'
>>> s4='a'
>>> s5='a'
#utilizando +
>>> s_all=s1 + ' ' + s2 + ' ' + s3 + ' ' + s4 + ' ' + s5
>>> s_all
'a a a a a'

#utilizando %
>>> s_all2='%s %s %s %s %s' % (s1, s2, s3, s4, s5)
>>> s_all2
'a a a a a'
```

A segunda forma normalmente é mais limpa, e é a que será utilizada nos exemplos.

Utilizamos a função **split** quando queremos dividir uma string com campos separados por algum caractere delimitador, e obter uma lista que permita acesso individual aos campos. Por exemplo, em ambientes Unix/Linux, temos o arquivo */etc/passwd* que contém uma lista com os usuários do sistema. Essa lista possui linhas, que possuem campos separados pelo caractere “:”. Se quisermos manipular os campos de uma linha individualmente, podemos fazer como no exemplo da **Listagem 9**.

Da mesma forma, se temos uma lista e queremos juntar os seus elementos em uma única string, podemos utilizar a função **join**:

```
>>> user2=':'.join(campos_user)
>>> user2
'root:*:0:0:System Administrator:/var/root:/bin/sh'
```

Uma particularidade da função **join** é que ela é aplicada na string que contém o delimitador, como podemos ver pelo exemplo.

Uma última consideração sobre strings é o uso do caractere **r** (“raw string”) para desabilitar o uso de caracteres de escape, como no exemplo da **Listagem 10**.

No primeiro comando **print**, o caractere de escape **\t** é interpretado como um **<TAB>**, ou seja, um caractere invisível que ocupa o espaço equivalente a quatro ou oito espaços em branco, dependendo do ambiente. No segundo comando, com o **r** no início, os caracteres **\t** são interpretados literalmente.

O tipo **dict** serve para armazenar pares de chaves e valores. É bastante útil quando desejamos criar uma coleção de valores relacionados e que podem ser nomeados para permitirem uma identificação mais clara, como no exemplo da **Listagem 11**.

Listagem 9. Exemplo de uso da função split().

```
>>> user='root:*:0:0:System Administrator:/var/root:/bin/sh'
>>> campos_user=user.split(':')
>>> campos_user
['root', '*', '0', '0', 'System Administrator', '/var/root', '/bin/sh']
>>> campos_user[0]
'root'
>>> campos_user[4]
'System Administrator'
>>> campos_user[5]
'/var/root'
>>> campos_user[6]
'/bin/sh'
```

Listagem 10. Uso de caracteres especiais.

```
>>> print('Teste\tTeste')
Teste      Teste
>>> print(r'Teste\tTeste')
Teste\tTeste
```

Listagem 11. Uso do tipo dict().

```
>>> d = dict()
>>> d['chave1'] = 'valor 1'
>>> d['chave2'] = 'valor 2'
>>> d
{'chave1': 'valor 1', 'chave2': 'valor 2'}
```

Além dos tipos básicos, é possível criarmos os nossos próprios tipos, através do mecanismo de **classes**. Este assunto está fora do escopo deste artigo, mas é importante saber que uma classe é, a grosso modo, como um registro de uma tabela, que possui vários campos ou **atributos**, e pode possuir **métodos** que executam ações sobre os atributos. Podemos ter várias **instâncias** de uma classe, da mesma forma que podemos ter várias linhas em uma tabela, cada uma com as suas características.

Uma classe que utilizaremos frequentemente nos nossos cenários se chama **datetime**, e é parte da biblioteca de mesmo nome.

Quanto ao nome das variáveis, existem algumas regras que devem ser seguidas:

- Devem iniciar por uma letra;
- Podem conter letras, números e underline;
- Não podem conter caracteres especiais;

Python diferencia maiúsculas de minúsculas, ou seja, os dois nomes no exemplo da **Listagem 12** são diferentes para o interpretador:

Listagem 12. Diferenciação de maiúsculas e minúsculas.

```
>>> variavel = 1
>>> VARIABEL = 10
>>> variavel
1
>>> VARIABEL
10
>>> Variavel
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'Variavel' is not defined
```

É recomendado que se nomeie as variáveis de forma significativa, para garantir boa legibilidade do código. É importante que o seu código seja facilmente entendido por outros que eventualmente precisem alterá-lo.

Controle de Fluxo

Instruções de controle de fluxo são aquelas que interferem na sequência de execução dos comandos de um programa, seja fazendo com que determinados comandos sejam executados e outros não (decisão), seja fazendo com que comandos sejam executados repetidamente (loop).

Uma instrução de decisão avalia uma ou mais condições e, com base no resultado, executa uma opção ou outra. Em Python, a instrução **if/else** é quem faz esse papel, como podemos ver na **Listagem 13**.

Listagem 13. Exemplo de instrução if/else.

```
a = 1
b = 2
if a > b:
    print('a maior que b')
elif a == b:
    print('a igual a b')
else:
    print('a menor do que b')
```

Nesse exemplo, criamos duas variáveis, realizamos um teste, e executamos comandos baseados no resultado do teste. Se fossemos escrever a instrução apresentada na **Listagem 13** em português, teríamos algo assim:

Se a for maior do que b (comando **if**)
Diga "a maior que b"
Senão, se a for igual a b (comando **elif**)
Diga "a igual a b"
Senão, se nenhuma das opções anteriores for verdadeira (comando **else**)
Diga "a menor do que b"

Os comandos **if** e **elif** sempre recebem uma condição, enquanto que o comando **else** é utilizado como o caso alternativo, se todos os outros testes foram falsos. O único comando obrigatório em uma decisão é o **if**, isto é, não é preciso existir nem **elif** nem **else**. Na prática existem casos onde a ação alternativa é simplesmente não fazer nada, como no exemplo:

```
if a > b:
    print('a maior que b')
```

Nesse código, só mostraremos a mensagem caso **a** seja maior do que **b**. Caso a condição não seja verdadeira, o programa continua.

Qualquer expressão cuja resposta seja verdadeiro ou falso pode ser utilizada como condição. Tipicamente utilizamos operadores de comparação como **>** (maior), **<** (menor), **>=** (maior ou igual), **<=** (menor ou igual), **==** (igual) e **!=** (diferente). Expressões podem ainda ser combinadas com o uso de operadores lógicos como **and**, **or** e **not**, mais o uso de parênteses, como podemos ver na **Listagem 14**.

Listagem 14. Uso de parênteses e operadores de comparação.

```
if a > b and a > 0:
    print('a maior que b')
    print('E a maior que zero')

if s == 'Meu Nome':
    print('s igual a Meu Nome')

if (a > b and a > 0) or b == 2:
    print('a > b E a > 0, OU b igual a 2')
```

É importante notar dois aspectos da linguagem Python que foram mostrados nesses exemplos:

- Comandos que contêm várias partes, como **if**, utilizam o sinal ":" ao final de cada parte, para delimitar o início de um bloco;
- A indentação do código que está dentro do bloco é relevante e indica a qual bloco o código pertence.

Ou seja, quando temos um comando **if**, os comandos que serão executados dependendo da condição avaliada, devem estar indentados para "dentro" do bloco. Normalmente essa indentação é feita com quatro espaços, mas pode ser utilizado o caractere TAB. O importante é que o uso seja consistente, ou seja, ou utilizamos somente espaços ou somente TABs.

Para concluir a explicação sobre **if**, vale mencionar que é possível ter quantos comandos forem necessários dentro de um bloco **if**, até mesmo outro bloco **if**, como podemos ver na **Listagem 15**.

Repare a indentação do segundo bloco **if**, mais para dentro, para indicar que será executado dentro do bloco **if** mais acima.

Instruções de loop servem para executarmos um comando repetidamente, como por exemplo, para ler as diversas linhas de um arquivo texto, executamos um comando que busca a próxima linha de um arquivo dentro de um loop. Python possui algumas

instruções de loop, mas nos ateremos à mais comum, que é a instrução **for**.

Listagem 15. Exemplo com comandos if encadeados.

```
if (a > b and a > 0):
    print('a maior que b e a maior que 0')
if a == 10:
    print('a maior que 10')
```

A instrução **for** serve para acessarmos os vários elementos de uma lista individualmente e em sequência, como demonstra o código:

```
l = (1,2,3,4,5)
for i in l:
    print(i)
```

Nesse exemplo, o comando **for** é aplicado sobre a lista *l*, de modo que a cada iteração ou “volta” do loop, cada valor da lista é colocado na variável *i*, que então é mostrada na tela pelo comando **print**. Se fossemos escrever essa instrução em português, teríamos algo assim:

Coloque o próximo elemento de *l* em *i*
Mostre o valor de *i*

É possível perceber, como no **if**, o sinal de “:” ao final da linha do comando **for**, e a indentação do comando **print** para indicar que este pertence ao bloco do comando **for**.

Às vezes, desejamos utilizar uma sequência numérica no nosso loop. Nesse caso podemos utilizar a função **range** junto com o comando **for**, como no código:

```
for i in range(10):
    print(i)
```

A função **range** nesse exemplo retorna uma lista com 10 valores, iniciando em 0, ou seja, os números de 0 a 9.

Essa é a forma geral do comando **for**. Exploraremos outros usos desse comando nos cenários que serão mostrados no decorrer desse artigo.

Funções

Funções são blocos de código reutilizáveis. Sempre que percebermos que estamos escrevendo um mesmo conjunto de instruções repetidamente, este conjunto é um bom candidato a se transformar em uma função.

Por exemplo, é muito comum colocarmos nos nossos scripts comandos que geram mensagens para mostrar o progresso da execução, e é útil colocar a data atual junto das mensagens, para podermos, por exemplo, identificarmos quanto tempo determinada ação levou.

Para mostrar uma mensagem com data atual, formatada como `YYYY-MM-DD HH:MM:SS` podemos escrever um código assim:

```
>>> print('%s: %s' % (str(datetime.datetime.now()).strftime("%Y-%m-%d %H:%M:%S"),
'Erro na execucao'))
2013-10-26 17:22:19: Erro na execucao
```

Em vez de repetirmos diversas vezes essa linha inteira, podemos definir uma função que faça isso, como no exemplo:

```
>>> def show_msg(msg):
...     print('%s: %s' % (str(datetime.datetime.now()).strftime("%Y-%m-%d %H:%M:%S"),
msg))
```

De modo que, nas próximas vezes que quisermos mostrar uma mensagem, basta chamarmos a função passando como parâmetro a mensagem desejada:

```
>>> show_msg("Erro na execucao")
2013-10-26 17:24:31: Erro na execucao
```

Isso faz com que não tenhamos que repetir os comandos **print** e **datetime.now()**. Se um dia quisermos mudar o formato de data, basta mudar em um único local.

Bibliotecas e comandos internos

Os comandos que mostramos até aqui são internos da linguagem, como **for**, **if** e **print**. No entanto, boa parte do poder de Python vem do seu conjunto de bibliotecas, que são componentes que agrupam funcionalidades básicas e de uso geral, ou seja, que podem ser utilizadas por todos em qualquer programa.

Para utilizar uma biblioteca, é preciso indicar no programa que você deseja fazê-lo. Para isso, utilizamos o comando **import**:

```
import sys
```

Nesse exemplo, o comando **import** indica que pretendemos utilizar a biblioteca **sys**, que contém diversas funcionalidades que são de nosso interesse.

Por exemplo, os parâmetros de linha de comando passados para um programa ficam na lista **sys.argv**. Na **Listagem 16** temos um programa que mostra todos os parâmetros que foram passados na sua linha de comando.

Grave esse exemplo em um arquivo chamado *argv.py*, e o execute na linha de comando com alguns parâmetros, como podemos ver na **Listagem 17**.

Podemos reparar que o primeiro valor retornado é o nome do programa, e os demais são os valores que passamos na linha de comando, na ordem em que os especificamos, independentemente de tipo de dados.

Em outro exemplo de uso de biblioteca, queremos testar se determinado arquivo existe. Para isso utilizamos a função **os.path**.

`exists`, que testa se um arquivo ou diretório existe. Juntando com o exemplo anterior, passaremos um nome de arquivo como parâmetro na linha de comando, e testaremos se o mesmo existe, como podemos ver na **Listagem 18**.

Grave esse exemplo em um arquivo com o nome `exists.py`, e o execute como nos exemplos da **Listagem 19**.

Listagem 16. Parâmetros de linha de comando.

```
import sys

for arg in sys.argv:
    print(arg)
```

Listagem 17. Teste do script `argv.py`.

```
$ python argv.py 1 2 3 a b c "python" "script"
argv.py
1
2
3
a
b
c
python
script
```

Listagem 18. Exemplo do uso de bibliotecas.

```
import sys
import os

if len(sys.argv) < 2:
    print('Parametro nao foi passado!')
    exit()

file_name = sys.argv[1]

if os.path.exists(file_name):
    print('Arquivo existe:' + file_name)
else:
    print('Arquivo nao existe:' + file_name)
```

Listagem 19. Testes do script `exists.py`.

```
$ python exists.py
Parametro nao foi passado!
$ python exists.py exists.py
Arquivo existe: exists.py
$ python exists.py teste.txt
Arquivo nao existe: teste.txt
```

Essa é a forma básica de uso de bibliotecas. Para identificar quais são as bibliotecas que fazem as coisas que desejamos, basta procurar na documentação, que é bastante abrangente. Nos nossos cenários utilizaremos diversas bibliotecas, o que deve ajudar a orientar esse processo de busca.

Acesso a arquivos

A automação de tarefas comumente envolve acesso a um ou mais arquivos. O acesso a arquivos em Python é feito através da função `open`, como no código:

```
arquivo = open('argv.py', 'r')
```

Nesse exemplo executamos a função `open` e passamos como parâmetro o nome do arquivo a ser aberto (`argv.py`) e o modo como o arquivo deve ser aberto ("`r`" para leitura, "`w`" para escrita, "`a`" para escrita com `append`). Na variável "arquivo" agora existe uma referência para o arquivo em disco, que pode ser utilizada para ler os dados do arquivo, linha a linha, com um loop `for`, por exemplo:

```
for linha in arquivo:
    print(linha.rstrip())
```

A função `rstrip` utilizada no exemplo serve para remover o caractere `<ENTER>` (`\n`) do final de cada linha. Se não fizéssemos isso, teríamos uma linha em branco a mais para cada linha, porque o comando `print` adiciona um `<ENTER>` ao final de cada linha.

Para abrir um arquivo de escrita, basta trocar o modo de abertura.

Na **Listagem 20** mostraremos um script simples de cópia de arquivos texto, que exemplifica o processo de validação de parâmetros de entrada. Muitas vezes nossos programas precisam receber parâmetros externos que servem para alterar o seu comportamento a uma condição ou outra. Por exemplo, quando utilizamos o comando `dir` no Windows, podemos passar as opções `/s` para fazer uma listagem recursiva, `/b` para mostrar uma listagem simples, entre outras opções. A ideia é que sejamos capazes de implementar esse tipo de funcionalidade nos nossos programas em Python.

Manipulação de arquivos e diretórios

Boa parte do que fazemos com linguagens de script é manipulação de arquivos. A maior parte das funções de manipulação de arquivos nativas do Python é equivalente ao comando do sistema operacional que realiza a mesma ação, e não deve ser difícil inferir o seu uso. Na **Listagem 21** relacionamos as principais funções para manipulação de arquivos, com uma breve descrição.

Veremos na prática como utilizar algumas dessas funções nos cenários mais adiante.

Tratamento de exceções

Python fornece um mecanismo de tratamento de exceções para os casos onde algum erro não esperado é retornado durante a execução. Por exemplo, se tentarmos acessar um arquivo que não existe, será gerada uma exceção, como podemos ver na **Listagem 22**.

Se imaginarmos uma situação onde não sabemos se o arquivo existe, podemos colocar o nosso comando de abertura dentro de um bloco `try/except`, como podemos ver na **Listagem 23**.

Dessa forma, se ocorrer um erro, é possível executarmos comandos para contornar a situação.

Existe ainda mais uma opção que é o bloco `finally`, que é adicionado ao final do bloco `try`, para garantir que o que estiver ali seja executado, independentemente do que ocorrer nos blocos `try` ou `except`, como podemos ver na **Listagem 24**.

Listagem 20. Validação de parâmetros de entrada.

```
import sys
import os

if len(sys.argv) < 3:
    print('Parametro nao foi passado!')
    print('Uso: python copy.py <arquivo de origem> <arquivo de destino>')
    exit()

source_file_name = sys.argv[1]
dest_file_name = sys.argv[2]

overwrite = False

if len(sys.argv) == 4:
    overwrite = sys.argv[3]

if source_file_name == dest_file_name:
    print('Arquivo de origem igual arquivo de destino. Copia nao executada')
    exit()

if not overwrite and os.path.exists(dest_file_name):
    print('Arquivo de destino existe. Copia nao executada')
    exit()

if not os.path.exists(source_file_name):
    print('Arquivo de origem nao existe. Copia nao executada')
    exit()

source_file = open(source_file_name, 'r')
dest_file = open(dest_file_name, 'w')

for source_line in source_file:
    dest_file.write(source_line)

source_file.close()
dest_file.close()

print('Copia executada')
```

Listagem 21. Funções de manipulação de arquivos e diretórios.

```
os.remove('nome do arquivo') #remove arquivo
os.removedirs('caminho do diretorio') #remove estrutura de diretorios
os.rename('nome antigo', nome novo') #renomeia arquivo
os.getcwd() #retorna diretorio atual
os.rmdir('nome do diretorio') #remove diretorio
os.mkdir('nome do diretorio') #cria diretorio
os.chdir('nome do diretorio') #muda diretorio (cd)
os.listdir('diretorio base') #lista diretorio
os.walk('diretorio base') #navega pela arvore de diretorios
os.path.exists('nome do arquivo') #verifica se arquivo existe
os.path.basename('nome do arquivo') #retorna o nome do arquivo, sem diretorio
os.path.dirname('nome do arquivo') #retorna o caminho do arquivo, sem o nome
os.path.getmtime('nome do arquivo') #retorna a data do ultimo acesso
os.path.getctime('nome do arquivo') #retorna a data de criacao
os.path.getmtime('nome do arquivo') #retorna a data da ultima alteracao
os.path.getsize('nome do arquivo') #retorna o tamanho do arquivo em bytes
os.path.isdir('nome do arquivo') #verifica se o arquivo passado é um diretorio
os.path.isfile('nome do arquivo') #verifica se o arquivo passado é um arquivo
os.path.join('dir1', 'dir2') #concatena dois elementos para formar um caminho
os.path.split('dir1', 'dir2') #retorna lista com todas as partes de um caminho
shutil.copy('origem', destino') #copia arquivo
shutil.move('origem', destino') #move arquivo
```

Listagem 22. Exemplo de exceção não tratada.

```
>>> f=open('xxx','r')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IOError: [Errno 2] No such file or directory: 'xxx'
```

Listagem 23. Exemplo de tratamento de exceção com try/except.

```
>>> try:
...     f=open('xxx','r')
... except Exception as e:
...     print('Erro: %s' % e)
```

Listagem 24. Exemplo de uso da instrução finally.

```
>>> try:
...     f=open('xxx','r')
... except Exception as e:
...     print('Erro: %s' % e)
... finally:
...     print('Nao sei se deu erro, mas vou executar')
...
Erro: [Errno 2] No such file or directory: 'xxx'
Nao sei se deu erro, mas vou executar
```

O bloco **finally** é útil quando precisamos liberar recursos, como fechar arquivos ou conexões com serviços externos.

Cenários de Exemplo

Aqui são apresentados alguns cenários comuns no dia a dia de um sysadmin e que pretendem mostrar uma forma diferente de executar tarefas repetitivas. Partindo desses exemplos, que com base no material apresentado até aqui não devem ser de difícil compreensão, e com alguma pesquisa, a ideia é que o leitor esteja apto a escrever os seus próprios scripts em pouco tempo, e com isso perceber como pode ser simples a automação de algumas atividades que muitas vezes são realizadas manualmente.

Todos os scripts para Windows assumem que o usuário logado que está executando os scripts possui permissão de Domain Admin.

Cenário 1: Cópia de arquivos para múltiplos servidores

Você precisa atualizar semanalmente os binários de um sistema legado que está instalado em 50 servidores Windows espalhados pelo Brasil. A relação de binários a serem atualizados pode mudar a cada semana, e os desenvolvedores disponibilizam em um diretório do file server somente os arquivos que foram modificados.

Para automatizar essa rotina nós precisamos de duas informações:

- O diretório onde foram disponibilizados os arquivos;
- Um arquivo contendo a relação dos servidores e diretórios de destino.

Nossa relação de servidores será gravada em um arquivo com o nome *cenario1.txt*, com o seguinte formato:

```
SERVER01\APP\SERVER0
```

Onde:

- **SERVER0** é o nome do servidor;
- **APP\SERVER0** é o nome do compartilhamento e diretório para onde os arquivos devem ser copiados.

Na **Listagem 25** podemos ver como ficaria um script para realizar essa tarefa (lembre-se que a indentação é relevante em Python). As linhas que são iniciadas pelo caractere “#” são comentários dentro do código, e servem para explicar o funcionamento de alguns comandos.

Listagem 25. Script do cenário 1.

```
import os
import shutil
import datetime

#arquivo que contem a lista
server_list = 'cenario1.txt'

#diretorio que contem os arquivos a serem copiados
bin_dir = r'C:\APP'

#funcao padrao para mostrar mensagens com data e hora
def show_msg(msg):
    print('%s: %s' % (str(datetime.datetime.now()).strftime(
        ("%Y-%m-%d %H:%M:%S")), msg))

#verifica se o arquivo com a lista de servidores existe
if not os.path.exists(server_list):
    show_msg('Arquivo %s nao encontrado' % (server_list))
    exit(1)

#verifica se o diretorio com os arquivos a serem copiados existe
if not os.path.exists(bin_dir):
    show_msg('Diretorio %s nao encontrado' % (bin_dir))
    exit(1)

#lista os arquivos a serem copiados
bin_files = os.listdir(bin_dir)

#verifica se existem arquivos a serem copiados
if len(bin_files) == 0:
    show_msg('Nenhum arquivo encontrado em %s para ser copiado'
        % (bin_dir))
    exit(1)

#abre o arquivo que contem a lista de servidores
servers = open(server_list, 'r')

#monta loop que executara os comandos para cada servidor
for entry in servers:
    (server, dir) = entry.rstrip().split('|')
    path = r'\\%s%s' % (server, dir)

    #se o diretorio de destino nao existir, crie
    if not os.path.exists(path):
        os.mkdir(path)

    #monta loop que copiara cada arquivo para o servidor de destino
    for file in bin_files:
        shutil.copy(os.path.join(bin_dir, file), os.path.join(path, file))
        show_msg('Arquivo %s copiado para o servidor %s' % (file, server))

servers.close()
```

Se tudo correr bem na execução, devemos ver uma saída parecida com a mostrada na **Listagem 26**.

Listagem 26. Resultado da execução do script do cenário 1.

```
2013-10-26 20:32:04: Arquivo bin1.dll copiado para o servidor SERVER0
2013-10-26 20:32:04: Arquivo bin2.dll copiado para o servidor SERVER0
2013-10-26 20:32:04: Arquivo bin3.dll copiado para o servidor SERVER0
2013-10-26 20:32:04: Arquivo bin4.dll copiado para o servidor SERVER0
```

Cenário 2: Execução de comandos remotamente no Linux

A sua empresa definiu que cada analista terá um usuário nomeado nos servidores, com permissão de root, para garantir que os logs de segurança reflitam quem realizou qual ação no ambiente. No entanto, como não existe configurado um mecanismo de administração de usuários centralizado, não é possível garantir que não será criado nenhum usuário com permissão de root além dos permitidos pela empresa. Você precisa montar um script para auditar diariamente o seu parque de 100 servidores Linux, para garantir que os únicos usuários com permissão de root (id=0) são os usuários autorizados pela empresa.

Para automatizar essa rotina nós precisamos de duas informações:

- Um arquivo contendo a relação dos usuários autorizados;
- Um arquivo contendo a lista dos servidores a serem verificados.

Podemos ver como ficaria um script para executar esta tarefa na **Listagem 27**.

Cenário 3: Upload e Download de arquivos via FTP e HTTP

Quinzenalmente você precisa fazer o download de um arquivo com dados de empregados que o sistema de RH disponibiliza (via HTTP) e fazer o upload deste arquivo para o servidor FTP da empresa externa que faz o processamento da folha de pagamento. O nome do arquivo deve conter o nome da empresa e a data do arquivo no formato YYYYMMDD.

Para este cenário, precisamos saber:

- Servidor HTTP de origem;
- Servidor FTP de destino e credenciais;
- Nome do arquivo a ser baixado;
- Nome que o arquivo deve ter no servidor FTP.

Podemos ver como ficaria um script para executar essa tarefa na **Listagem 28**.

Cenário 4: Listagem de usuários em grupos locais (Windows)

Você precisa auditar todos os 500 desktops da empresa para garantir que não existam outros usuários no grupo local *Administrators* que não seja o usuário *Administrator* e do grupo Domain Admins.

Neste cenário temos um arquivo de nome *cenario4.txt* que contém a lista de desktops a serem verificados. Podemos ver um exemplo na **Listagem 29**.

Listagem 27. Script do cenário 2.

```
import paramiko

#inicia o cliente SSH
ssh = paramiko.SSHClient()
#define que as chaves publicas serao aceitas automaticamente
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())

#usuario e senha de conexao
user='root'
password='senha do root'

#lista de servidores a serem verificados
server_list = 'cenario2.txt'

#lista de usuarios que podem ser root
root_list = 'root.txt'

#cria lista vazia para receber os usuarios que podem ser root
root_allowed = []

#abre arquivo com relacao de usuarios que podem ser root
root_users = open(root_list, 'r')

#le a lista de usuarios que podem ser root e adiciona a uma lista em memoria
for root in root_users:
    root = root.rstrip()
    if len(root) > 0:
        root_allowed.append(root)

root_users.close()

#abre arquivo com relacao de servidores que devem ser verificados
servers = open(server_list, 'r')

#monta loop que executara os comandos para cada servidor
for server in servers:
    #se o nome do servidor estiver em branco, pare o loop
    if len(server) == 0:
        break
    #remove caractere \n do final do nome do servidor
    server = server.rstrip()

    #se conecta com o servidor
    ssh.connect(server, username=user, password=password)

    #executa o comando no servidor
    (stdin, stdout, stderr) = ssh.exec_command('cat /etc/passwd')
    user_list = stdout.readlines()

    #monta loop para ler cada linha retornada do arquivo /etc/passwd
    for user in user_list:
        #faz o split da linha utilizando o separador ":"
        user_fields = user.split(':')
        if len(user_fields) < 5:
            continue

        #obtem nome e id de usuario
        user_name = user_fields[0]
        user_id = int(user_fields[2])

        #se id igual a 0 e nome de usuario nao estiver
        #na lista de permitidos
        if user_id == 0 and user_name not in root_allowed:
            print('Servidor %s: Usuario %s possui permissao
de root indevidamente' % (server, user_name))

servers.close()
```

Listagem 28. Script do cenário 3.

```
import datetime
import ftplib
import urllib.request
import urllib.parse

http_url='http://servidor.intranet/arquivo_rh.zip'
ftp_server='ftp.empresa.com'
ftp_user='nome do usuario'
ftp_passwd='senha do usuario'
ftp_file_name='EMPRESA_%s.zip' % datetime.datetime.now().strftime('%Y%m%d')

#abre conexao com o servidor web
request=urllib.request.Request(http_url)
#envia request para fazer o download do arquivo
response=urllib.request.urlopen(request)

print('Inicio do download do arquivo')
#abre arquivo local em modo binario para escrita
local_file=open(ftp_file_name, 'wb')
#grava dados recebidos via HTTP
local_file.write(response.read())
local_file.close()
print('Download do arquivo %s concluido com sucesso' % ftp_file_name)

#abre o arquivo gravado em modo binario para leitura
local_file=open(ftp_file_name, 'rb')

print('Inicio do upload do arquivo para %s' % ftp_server)
ftp_client=ftplib.FTP(ftp_server)
#faz login com o servidor FTP
ftp_client.login(ftp_user, ftp_passwd)

#envia arquivo em modo binario
ftp_client.storbinary('STOR %s' % ftp_file_name, local_file)
print('Upload do arquivo %s concluido com sucesso' % ftp_file_name)
local_file.close()
```

Listagem 29. Exemplo de arquivo de configuração.

```
PC0
PC1
PC2
...
```

Um exemplo de script para executar essa tarefa é apresentado na **Listagem 30**.

Cenário 5: Configuração de serviços (Windows)

Você precisa listar todas as contas de serviço que são utilizadas em todos os serviços dos servidores Windows, para verificar se há algum serviço configurado com outra conta que não sejam as autorizadas.

Para este cenário é preciso instalar a biblioteca WMI, que não vem na instalação default do ActivePython. Veja o endereço para download na seção **Links**.

Neste cenário temos um arquivo de nome *cenario5.txt* que contém a lista de servidores a serem verificados. Podemos ver um exemplo deste arquivo na **Listagem 31**.

No script listaremos os serviços de cada servidor, com as respectivas contas. Um exemplo de como ficaria um script para realizar essa tarefa é apresentado na **Listagem 32**.

Listagem 30. Script do cenário 4.

```
import win32net

#arquivo que contem a lista de desktops
desktop_list = 'cenario4.txt'

#lista de membros permitidos
allowed_members = []
allowed_members.append('Administrator')
allowed_members.append('Domain Admins')

#parametros da funcao NetLocalGroupGetMembers
level=2
resume=0

desktops = open(desktop_list, 'r')

#nonta loop para executar comandos para cada desktop
for desktop in desktops:
    desktop = desktop.rstrip()
    #chama a funcao do Windows que retorna a lista
    #de membros de um grupo
    group_info=win32net.NetLocalGroupGetMembers
        (desktop, 'Administrators', level, resume)
    members=group_info[0]

    #monta loop para executar comandos para cada membro
    #do grupo Administrators listado
    for member in members:
        if member['domainandname'].split('\\')[1]
            not in allowed_members:
            print('Desktop %s: usuario %s membro do grupo Administrators' % (desktop,
                member['domainandname']))

desktops.close()
```

Listagem 31. Exemplo de arquivo de configuração.

```
SERVER0
SERVER1
SERVER2
...
```

Listagem 32. Script do cenário 5.

```
import wmi

server_list = 'cenario5.txt'
servers = open(server_list, 'r')

for server in servers:
    server = server.rstrip()
    wmi_client = wmi.WMI(server)
    for service in wmi_client.Win32_Service():
        print('%s\t%s' % (service.Caption, service.StartName))

servers.close()
```

No exemplo da **Listagem 32** utilizamos o caractere <TAB> (\t) no comando print para separar os campos que serão mostrados na saída do script. Esse formato de texto, com colunas separadas por <TAB>, é simples de importar para uma ferramenta como o Excel, por exemplo, para analisar os dados posteriormente, sem necessidade de conversão, somente realizando copy/paste. É possível utilizar outro separador, como vírgula ou ponto e vírgula, mas nesse caso seria necessário realizar um processo de importação de dados.

Conclusão

Neste artigo foi apresentada a linguagem Python como uma ferramenta poderosa para a automação de processos do dia a dia. Foram mostrados os tipos de dados comumente utilizados e suas particularidades, principais instruções de controle de fluxo, e recursos para manipulação de arquivos. Além disso, foram explorados alguns cenários comuns de uso, que podem servir de exemplo para a construção de soluções mais abrangentes de automação.

Autor



Mateus Espadoto

mespadoto@yahoo.com

Há 14 anos trabalhando com TI, já trabalhou em áreas tão diversas como desenvolvimento de software, infraestrutura e arquitetura corporativa. Atualmente trabalha como Arquiteto de Software.



Links:

Site da linguagem Python

www.python.org

Site da ActiveState

www.activestate.com

Endereço para download da biblioteca WMI

<https://pypi.python.org/pypi/WMI/>

BYOD: primeiros passos para implantação

Quais os recursos disponíveis no mercado?

BYOD é um termo criado para descrever a tendência global que viabiliza aos profissionais trazerem seus dispositivos pessoais para desempenhar suas atividades na empresa. Também chamado de BYOT (*Bring Your Own Technology*), sua abrangência inclui todos os tipos de dispositivos, sejam eles smartphones, tablets ou notebooks, e envolve tanto o hardware quanto o software.

Segundo o relatório do IDC, os consumidores estão utilizando cada vez mais os seus dispositivos móveis como seu ponto de acesso principal à Internet, migrando do antigo sistema *PC based*. A previsão é de que em 2015, pela primeira vez na história, o número de consumidores norte-americanos que utilizam estes dispositivos seja maior que o de consumidores que utilizam computadores. Somente nos EUA, o número de pessoas que acessam a Internet por meio de computadores pessoais vai reduzir de 240 milhões em 2012 para 225 milhões em 2016. Neste mesmo período, o número de dispositivos móveis irá aumentar de 174 para 265 milhões.

No Brasil, o crescimento do mercado de dispositivos móveis dobrou entre abril de 2012 e 2013. Este estudo da Flurry, empresa voltada ao mercado móvel, não revelou números exatos, mas considera que temos uma faixa de crescimento de 100% a 199%. Um ponto importante da pesquisa também mostra que muito do crescimento é resultado da adoção de mais de um dispositivo móvel por pessoa, por exemplo, os que possuem smartphones e tablets.

Junto a este crescimento, as aplicações já ultrapassam os 700.000, considerando somente a Apple. Isto sem lembrar que existem outros grandes fornecedores como Microsoft, BlackBerry e Google. No entanto, estamos apenas começando a geração das aplicações *smart*, que se utilizam das informações disponibilizadas pelo usuário para oferecer soluções *on-demand*. Hoje, por exemplo, as aplicações “inteligentes” são capazes de definir onde você está, quais as opções de restaurantes existem ao seu entorno, lojas, informações de deslocamento e sugestões de lazer.

Fique por dentro

O artigo levanta as questões principais que envolvem a implantação de um programa de BYOD (*Bring Your Own Device*) em um contexto empresarial. De forma objetiva, são salientadas melhores práticas em relação à segurança, mostrando algumas ferramentas já disponíveis no mercado que facilitam a implementação. Os programas de BYOD estão se tornando cada vez mais aplicáveis a todos os tipos de empresas, sejam elas de grande, médio ou pequeno porte. Este artigo é voltado para apoiar o profissional de TI no planejamento e implantação de BYOD, uma vez que é necessário repensar na segurança dos dados, recursos disponíveis e riscos organizacionais.

Diante deste rápido crescimento, seja na comercialização ou no desenvolvimento de aplicações cada vez mais interativas, o mercado de dispositivos móveis tem ampliado seu lugar. Com isso, tanto o mercado de desktops quanto o de notebooks têm sido ameaçados, uma vez que aparelhos como smartphones e tablets são um meio mais ágil de acesso a informações online.

No entanto, não somente esta constatação de aumento da participação de mercado justifica a adoção destes dispositivos em ambientes corporativos. As empresas investem e adotam soluções com o objetivo de terem algum retorno, seja ele financeiro, operacional ou produtivo.

O que leva as empresas a adotarem o BYOD?

Esta tendência está alinhada a três pontos centrais. Primeiro, a popularização dos dispositivos móveis, que vem acompanhada pelo crescimento do número de aplicações e pela disponibilidade de acesso aos recursos corporativos a partir destes aparelhos. Segundo, a facilidade propiciada aos usuários, pois eles conseguem integrar todas as suas informações online em um único perfil. Pela perspectiva do usuário, utilizar um único ponto de acesso para todas as informações pessoais e profissionais é um recurso que tende a melhorar a sua produtividade. Terceiro, para a organização representa uma economia interessante se levarmos em consideração que haverá uma redução dos investimentos na aquisição e manutenção destes dispositivos. Além disso, é possível, com BYOD, oferecer aos funcionários a mobilidade para

algumas de suas atividades, benefício antes ofertado apenas aos altos cargos.

Sobre este assunto, o diretor de operações do governo americano afirmou, em pesquisa realizada pela *Trend Micro commissioned Forrester Consulting*, que a produtividade no trabalho definitivamente aumentou após a adoção de um programa de BYOD. Segundo ele, os funcionários estão satisfeitos utilizando os dispositivos que escolheram e têm se comunicado com os outros colegas com mais frequência, melhorando o ambiente de trabalho.

Por estas razões, o BYOD não deve ser considerado apenas como uma tendência, mas como uma nova possibilidade de oferecer os recursos de TI organizacionais. Sendo assim, as estratégias das empresas têm sido tentar absorver esta demanda, uma vez que cada dia é mais complicado evitar o uso destes dispositivos em alguma atividade profissional. Hoje, tanto o acesso ao e-mail, quanto a arquivos, relatórios, informações, são acessíveis através de interface Web. Isto possibilita que estes dados estejam disponíveis em aparelhos móveis, independente se os mesmos sejam ou não gerenciados pelo BYOD.

Portanto, durante a análise de risco é importante que questões como estas sejam consideradas como vulnerabilidades. Tanto a segurança dos dispositivos quanto a dos sistemas precisam ser analisadas em conjunto para que, de forma sistêmica, seja elaborado o relatório final com os argumentos-chaves para adoção ou não de uma política organizacional mais ampla.

Num estudo feito em 2012 pela *Trend Micro commissioned Forrester Consulting* sobre os principais pontos que levam as empresas a implantar o programa de BYOD observa-se que o foco tem sido as facilidades propiciadas aos funcionários e o seu consequente aumento de produtividade.

É importante enfatizar que o BYOD pode ser considerado como um recurso de RH, pois é tratado como um benefício oferecido pela organização. Algumas empresas também o consideram como um diferencial competitivo. Hoje, muito se discute sobre os benefícios desta tecnologia, sendo que, em muitos casos, não são definidos de antemão quais os resultados esperados e como serão alcançados.

Quais as questões a serem consideradas na implantação?

Presenciamos nos últimos anos as primeiras implantações e resultados dos programas de BYOD. É preciso ter em mente que toda esta mudança ainda é muito recente e que estes programas foram implementados baseados em experiências anteriores, com o uso de práticas conhecidas e das atuais diretrizes de segurança organizacional, todas desenvolvidas para uso com outras tecnologias. Isto é, empregamos princípios de segurança da informação, gerenciamento de arquivos, governança de TI, gestão de pessoas e de projetos para delinearmos quais estratégias serão utilizadas na implantação.

No entanto, mesmo envolvendo todas estas áreas, o que mais tem preocupado as empresas num programa de BYOD são as questões relacionadas à segurança da informação. Por padrão, os proprietários dos dispositivos móveis são os responsáveis pela decisão de

como irão proteger seus equipamentos e informações. Além disso, o usuário tende a escolher o aparelho de melhor design e usabilidade, não se importando tanto com a segurança oferecida.

As aplicações e downloads disponibilizados pelas lojas como a *Play Store* do Android e a *App Store* da Apple, também aumentam o risco de acessos não autorizados às informações contidas nos dispositivos. Uma vez que podem trazer *malwares* e *keyloggers*, elas possibilitam inclusive que seja infectada a rede da empresa.

Existe uma tendência dos riscos relacionados à segurança da informação continuar a crescer, principalmente porque os custos relacionados às soluções existentes ainda são caros para as pequenas e médias empresas, o que torna estes equipamentos um alvo interessante para ataques. Portanto, algumas questões devem ser pensadas antes da adoção de um programa de BYOD, como analisaremos a seguir.

Elaborar uma política de segurança da informação

A política deve ser voltada para a proteção dos dados corporativos. Com isso, ela tem que:

- Definir o tipo de informação que pode ser acessado e por quem: e-mails, sites corporativos, tipos de documentos, diretórios e arquivos compartilhados, entre outros;
- Estabelecer o direito de impor a política corporativa aos dispositivos, como: instalação de antivírus, obrigatoriedade de uso de senha complexa e instalação de software de gerenciamento remoto;
- Definir quais funcionários podem usufruir do programa de BYOD;
- Definir quais aplicações não devem ser utilizadas nos dispositivos que participam do programa, como: *Jailbreak*, *cracks*, aplicações para desbloqueio, softwares não licenciados;
- Definir os pré-requisitos ou modelos de dispositivos que poderão participar do programa.

Planejar o gerenciamento dos dispositivos no ambiente corporativo

Nesta fase o importante é analisar os pontos levantados anteriormente, planejando e trabalhando na operação do serviço de TI, com o objetivo de criar novos processos e soluções para atender a estes novos usuários. Neste contexto, foram levantados os principais tópicos que podem significar mudanças internas necessárias ao departamento de TI, para que sirvam como ponto inicial de análise. São eles:

- Aplicar a política de segurança definida anteriormente utilizando softwares de apoio, como os MDMs (*Mobile Device Management*), e as configurações de acesso à rede e arquivos;
- Elaborar o processo interno para atendimento a questões relacionadas a furtos e perda de dispositivos. Levar em consideração a remoção de acessos à rede da corporação e a exclusão das informações contidas no dispositivo;
- Elaborar o processo de tratamento para os dispositivos que estiverem com algum *malware*;
- Criar o processo de suporte e atendimento a estes dispositivos, definindo e alterando o que será oferecido pelo catálogo de serviços da TI;

- Treinar o corpo técnico da TI que realizará o suporte a estes dispositivos;
- Prever o impacto destes dispositivos na rede corporativa, como o aumento do consumo da banda de rede wireless e o número de atendimentos realizados pelo suporte.

Treinar os usuários

Os funcionários devem compreender as estratégias de divulgação de informações organizacionais, a importância de manter seus dispositivos seguros e como fazer isto. Seguem algumas sugestões de tópicos a serem abordados nesta fase:

- Malwares, o que são e como podem roubar os seus dados;
- Política de segurança da informação, o que prevê a política da empresa e quais são os pontos de apoio do funcionário em caso de dúvidas;
- Quais os recursos existentes voltados à segurança dos dispositivos;
- Como separar as informações pessoais das profissionais, focando no gerenciamento do tempo, na priorização de atividades e na otimização de resultados.

Estabelecer a política de gestão

Em paralelo a implantação, a administração e o departamento de Recursos Humanos devem criar uma política onde precisam prever o BYOD como um benefício a ser utilizado pela empresa, estabelecendo novas diretrizes para os funcionários que desejarem ingressar no programa. Assim sendo, apresenta-se a seguir algumas questões que poderão servir como norteadoras desta fase:

- Opções para implantação: dispositivos de funcionários ou compartilhados entre organização e funcionário. Neste caso, avalie quais os custos que os funcionários terão ao ingressar no programa, se o dispositivo será comprado pela empresa que o transfere legalmente ou vende para o funcionário; ou se ele deverá adquirir o modelo homologado à parte de qualquer programa corporativo;
- Como será incorporado o BYOD caso já exista o benefício de celular corporativo – caso o funcionário não queira participar do programa a empresa irá adquirir o aparelho? O dispositivo da empresa poderá ser usado para ligações e fins particulares?
- Estratégia de reembolso: Definir, dentro do gasto mensal do funcionário, quais os custos que serão absorvidos pela organização.

Num estudo da Forrester Consulting on behalf of Trend Micro, realizado em 2012, que levantou os principais desafios na implantação dos programas de BYOD, pode-se observar que as empresas estão realmente preocupadas com a definição das políticas e dos processos internos para suportar esta atividade.

Com base nas questões levantadas anteriormente, todas envolvendo decisões da alta gestão alinhadas a TI, é possível iniciar um programa de BYOD trabalhando na redução dos riscos, de acordo

com as expectativas organizacionais de custos e resultados. Neste sentido, é importante que o profissional de TI que esteja envolvido e alinhado ao programa esteja disposto a enfrentar os desafios que esta solução apresenta.

Alinhando estas questões pertinentes à organização, um dos fatores mais preocupantes para o setor de TI é o que se refere aos riscos relacionados à perda da segurança da informação. Esta preocupação foi confirmada pelo estudo da Symantec, cujo gráfico é exposto na **Figura 1**. Este confirma o aumento do número de vulnerabilidades nos dispositivos móveis em 32%, entre 2011 e 2012. Segundo a pesquisa, o roubo de informações encabeça a lista das atividades realizadas por *malwares*.

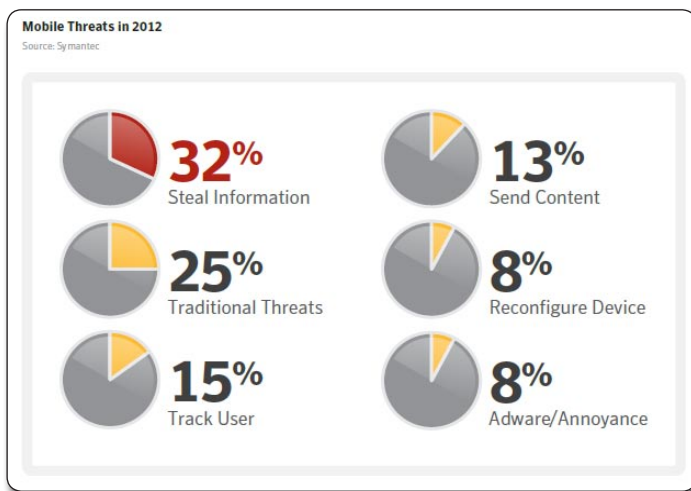


Figura 1. Números das vulnerabilidades dos dispositivos móveis em 2012

Por isto, hoje as soluções de gerenciamento de dispositivos móveis – *Mobile Device Management* – veem para auxiliar as organizações neste controle de informações, aplicações, *downloads* e gestão destes equipamentos.

Com o intuito de apoiar a definição de qual das ferramentas disponíveis no mercado mais se aplica à realidade da sua organização, foi realizado o levantamento a seguir das aplicações mais utilizadas atualmente. O objetivo é oferecer uma visão geral das suas principais características, citando suas compatibilidades e soluções oferecidas.

Softwares de gerenciamento de dispositivos móveis

Hoje temos várias opções na hora de escolher um software de MDM (*Mobile Device Management*) mais adequado para um programa de BYOD. O importante desta escolha é tentar conciliar os custos de aquisição, instalação e treinamento da aplicação com as diretrizes listadas na política de segurança. Por isto, é importante que neste momento já seja possível mapear quais os recursos indispensáveis, os necessários e os opcionais, para que se defina qual o software mais apropriado a sua empresa.

Além disso, é necessário verificar se as aplicações de MDM possuem algumas limitações relacionadas à gestão da informação que precisam ser trabalhadas de outra forma dentro da organização.

Por exemplo, o acesso a arquivos corporativos através de plataformas web é uma das limitações que não é tratada por este tipo de solução de gerenciamento.

De qualquer forma, teste e avalie algumas plataformas antes de realizar a compra e implantação do programa de BYOD. Praticamente todas as aplicações que serão listadas a seguir possuem uma versão de demonstração, a qual se recomenda fortemente que seja instalada e avaliada antes de se tomar a decisão pela melhor solução.

1 – AirWatch

A AirWatch possui uma solução de MDM que apresenta suporte a todas as plataformas móveis do mercado: Android, Apple iOS, BlackBerry, Symbian, Windows Phone e Windows Embedded. Funciona com base na gestão voltada a ativos, onde os administradores podem atribuir níveis diferenciados de propriedades, isto é, corporativo compartilhado, dedicado ou de propriedade do funcionário. O console de administração pode ser acessado via Web e tem boa usabilidade, permite rastreamento de ativos, bloqueio, formatação e gerenciamento remoto, criptografia de informações e a aplicação das políticas, tais como configuração de Wi-Fi, atualizações e instalação remota de aplicações.

Possui estrutura que suporta escalabilidade, alta disponibilidade, monitoramento e recuperação de desastres. Ele se integra a sua infraestrutura local, ao serviço de diretórios, e-mails, certificados digitais, eventos de sistema, aplicações internas e repositórios de conteúdo.

Além disso, oferece alguns recursos voltados à gestão do tempo de vida dos ativos, como ferramentas de varredura, testes e diagnósticos, para identificação proativa de problemas. Na **Figura 2** é possível verificar a tela de gerenciamento dos ativos.

2 – Amtel

A Amtel oferece versões de sua aplicação a um custo muito baixo, o que tem ampliado o seu mercado e facilitado a sua divulgação. É recomendada para empresas de pequeno e médio porte por disponibilizar apenas os recursos básicos principais para a implantação de um programa de MDM. Na sua versão gratuita, lançada em 2012, o produto oferece os serviços de configuração de políticas de senha, relatórios sobre Apps, inventário de aparelhos e status geral.

Como o intuito da Amtel é promover a gestão centralizada de dispositivos, a última versão já agrega o gerenciamento de dispositivos Mac, com previsão para agregar também a plataforma Windows para computadores.

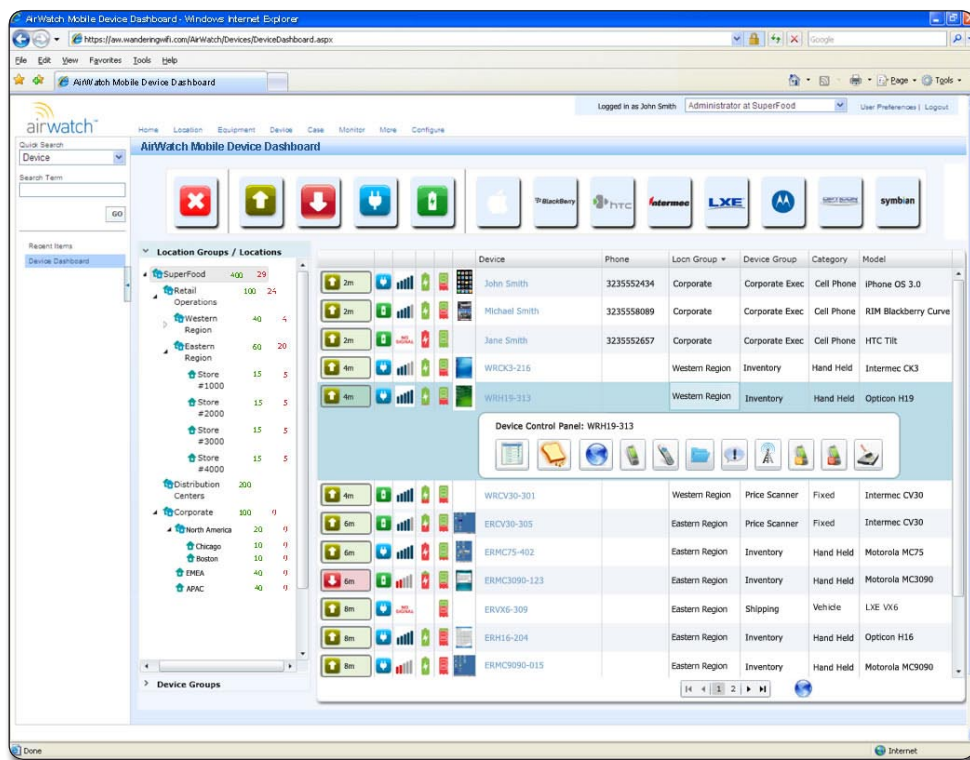


Figura 2. Screenshot do software da AirWatch

Dentro da sua interface, é possível separar as informações pessoais da corporativa, facilitando tanto para o usuário quanto para a gestão local. Possui recursos de criptografia, formatação e bloqueio remoto, gerenciamento de conteúdo, com lista negra do Play Store e App Store, e desativação de recursos remotamente, como câmeras. Na **Figura 3** é possível visualizar uma das telas de gerenciamento de aplicações. Hoje, a última versão suporta o gerenciamento de dispositivos Android, iOS, BlackBerry, Windows Phone, Symbian e WebOS.

3 – Excitor

A ferramenta *Dynamic Mobile Exchange* da Excitor introduz a ideia de containerização. O administrador centraliza o gerenciamento somente da Partição Corporativa (contêiner corporativo), tendo com isto a possibilidade de limpar os dados desta área quando o funcionário é desligado ou o equipamento é perdido. Esta solução garante a separação entre os dados privados dos corporativos, e isto libera ao usuário adicionar e subtrair recursos e funcionalidades do seu aparelho. No entanto, durante todo este processo os dados corporativos que estão dentro do contêiner permanecem seguros, protegidos e não são afetados.

A solução ainda oferece uma central para distribuição de aplicações e um aplicativo seguro para o acesso a e-mails, calendários e agenda de contatos e garante a remoção completa das informações caso necessária. O princípio básico do DME é o administrador não ter que gerenciar uma solução de VPN interna para garantir a segurança nestes aparelhos. Na **Figura 4** é possível visualizar o ambiente de configuração e administração dos dispositivos e usuários.

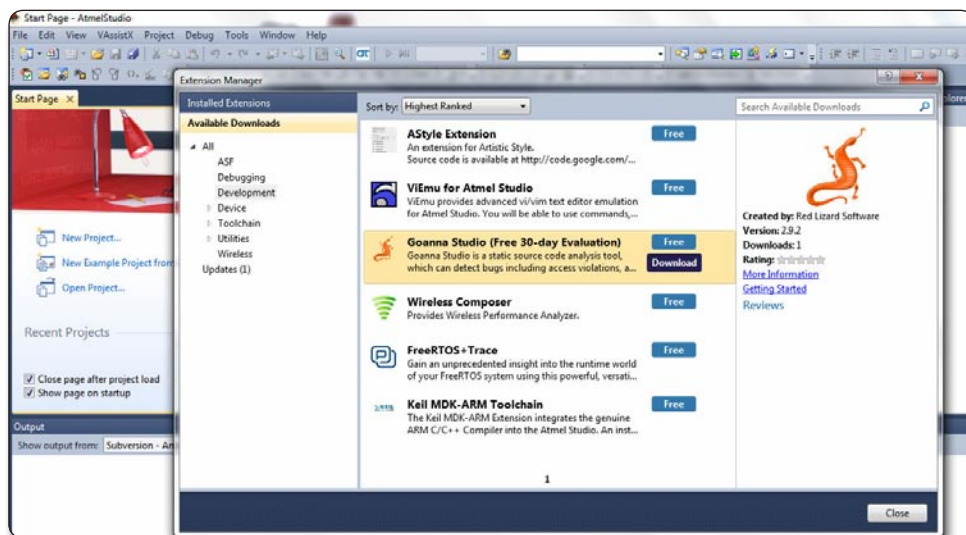


Figura 3. Screenshot do software da Amtel

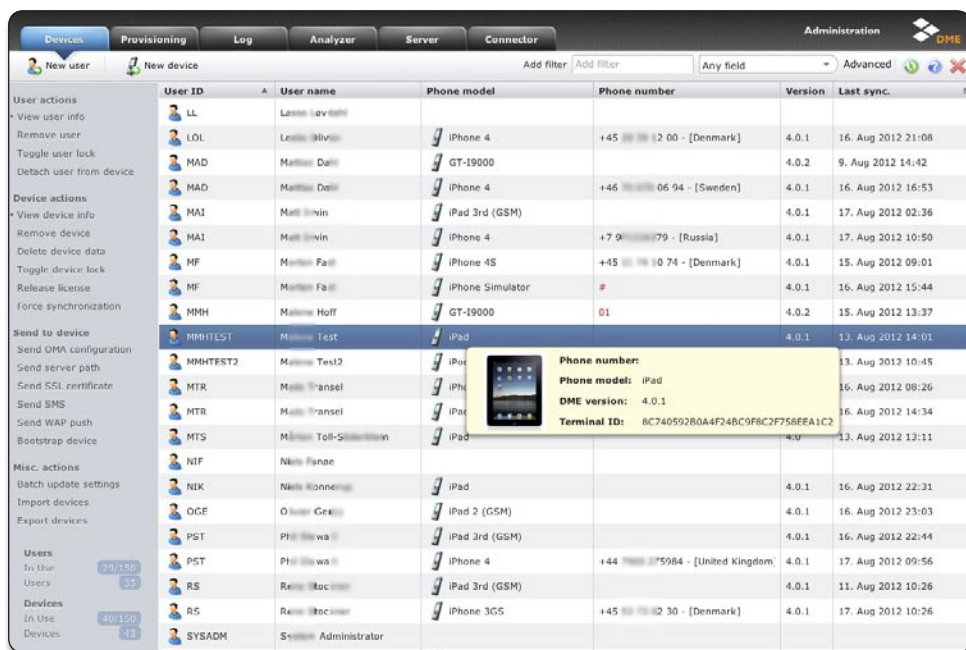


Figura 4. Screenshot do software da Excitor

4 – BlackBerry

A BlackBerry lançou na versão 10 da sua aplicação de MDM o *BlackBerry Enterprise Server* (BES), uma solução multiplataforma que suporta o iOS e Android. Desta forma, a ferramenta utiliza a rede segura da BlackBerry e VPN para comunicações. O espaço de trabalho, através do recurso *BlackBerry Balance*, utiliza o conceito de contêiner e de espaços lógicos distintos e o *sideloading* permite que você publique as aplicações organizacionais para distribuir entre os dispositivos.

O BES 10 possibilita configurar um servidor HTTP na rede privada da sua empresa e construir aplicativos corporativos que se conectam diretamente ao servidor. Assim os usuários poderão

baixá-lo de qualquer lugar de forma segura, protegida através de firewalls, sem a necessidade de gerenciar um ambiente protegido. Você ainda poderá criar políticas de controle de downloads, como *whitelists* das aplicações disponíveis no BlackBerry App World.

Além disso, o BES 10 permite um controle maior das interações via redes sociais, gerenciando o conteúdo quase em tempo real, onde todas as mídias passam pelo mesmo filtro, utilizando uma única aplicação. A Figura 5 mostra a tela de administração das políticas de e-mail.

5 – APPTec

A aplicação da APPTec tem um custo baixo e por isto se torna muito atrativa. Possui dentre as funcionalidades básicas de gerenciamento de ativos a gestão de inventário, custos, aplicações, conexões, segurança, PIM, conteúdo, backup e monitoramento. Além disso, na Figura 6 vê-se que a interface da ferramenta é de fácil usabilidade, o que também auxilia na sua divulgação.

Como recurso para proteção de informações, a ferramenta possui uma solução que prevê, por exemplo, os casos de extravio do aparelho. Também trabalha o conceito de containerização dos dados corporativos, criptografia e integração com o serviço de diretório local. Com ele é possível restaurar dados e aplicações em um novo aparelho, independente da plataforma, de forma simples e eficiente.

Ademais, essa aplicação possui suporte às plataformas iOS (iPhones e iPads), Android e Windows Phone. Atualmente

a solução está disponível também como um appliance virtual, funcionando como uma máquina virtual (VM).

6 – Citrix XenMobile

O XenMobile venceu o prêmio de inovação na área de soluções para tecnologia empresarial. Ele suporta a maior parte das plataformas, incluindo iOS, Android, BlackBerry, Symbian e Windows Phone. Como diferencial, inclui recursos que permitem uma maior segurança no acesso a e-mails Exchange, além da interface com o Sharepoint, SMG (Secure Gateway Mobile) e DLP (Sharepoint Data Leak Prevention).

O ZSM Lite é um componente criado para realizar a varredura de

dispositivos em ambientes BlackBerry e Windows. Ele oferece um conjunto de ferramentas de suporte remoto que auxilia na solução de problemas, controle e atualizações para dispositivos.

Com o recurso AppController o XenMobile permite, de forma segura, entregar Web e SaaS a partir de uma loja de aplicativos corporativa unificada. Além disso, o XenMobile oferece a solução Worx Mobile Apps, que distribui e gerencia os aplicativos móveis criados internamente, utilizando o Worx App SDK (interface de desenvolvimento). Ademais, oferece soluções para controle de ativos, inventário, políticas de segurança e serviço de autenticação de usuários. A **Figura 7** expõe a tela de gerenciamento das aplicações e documentos, com interface intuitiva e fácil de trabalhar.

7 – McAfee EMM

O McAfee EMM (*Enterprise Mobility Management*) gerencia desktops, laptops, dispositivos USB, smartphones e tablets. Possui recursos que viabilizam o acesso seguro a aplicativos móveis, como soluções antimalware, autenticação, arquitetura expansível e geração de relatórios de conformidade.

Com uma aplicação que suporta as plataformas mais populares (iOS, Android, Windows Phone e BlackBerry), a McAfee implementou a integração de serviços automatizados e ferramentas para apoiar o atendimento de TI.

A integração ao McAfee ePolicy Orchestrator (McAfee ePO) permitiu implementar políticas de segurança e facilitou o gerenciamento de conformidade, ajudando a reduzir os custos de suporte e gerenciamento de dispositivos móveis. Além disso, é possível personalizar o aparelho com credenciais únicas de um usuário para permitir o acesso a serviços de aplicativos específicos e a execução de políticas baseadas em perfis.

Também inclui funcionalidades como serviço de inventário, gerenciamento de dispositivos, controle de aplicações e loja virtual de aplicações corporativas e criptografia de dados. Na **Figura 8** pode-se verificar a interface de status e gerenciamento de ativos, com opções para aplicação de políticas de segurança e informações para inventário.

8 – MobileIron

A plataforma de gerenciamento móvel MobileIron suporta os principais ambientes do mercado (iOS, BlackBerry, Android e Windows Phone). Com recursos para gestão de dispositivos, aplicativos e conteúdo, o administrador gerencia a partir de um console baseado em web, dispositivos, políticas para criptografia e bloqueio, aplica senhas, bloqueia e apaga remotamente as

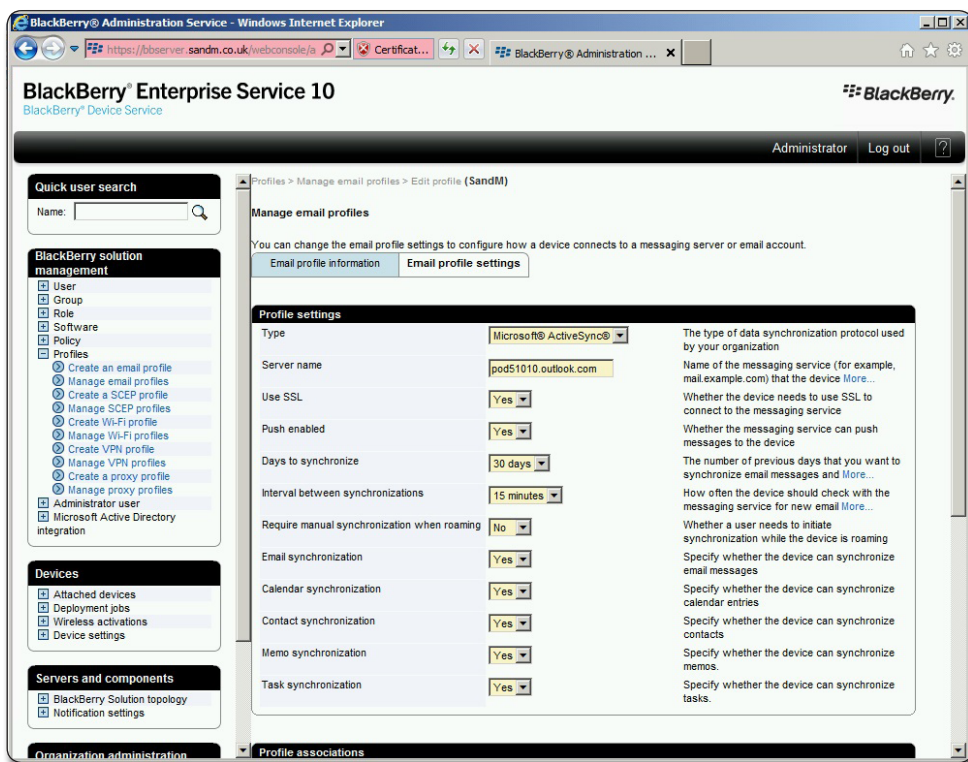


Figura 5. Screenshot do software da BlackBerry

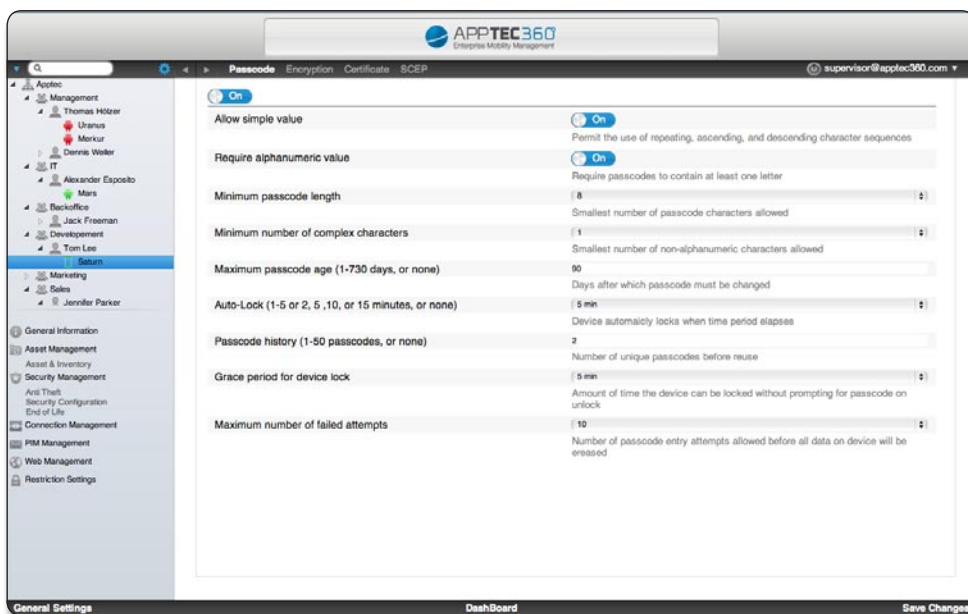


Figura 6. Screenshot da AppTec360

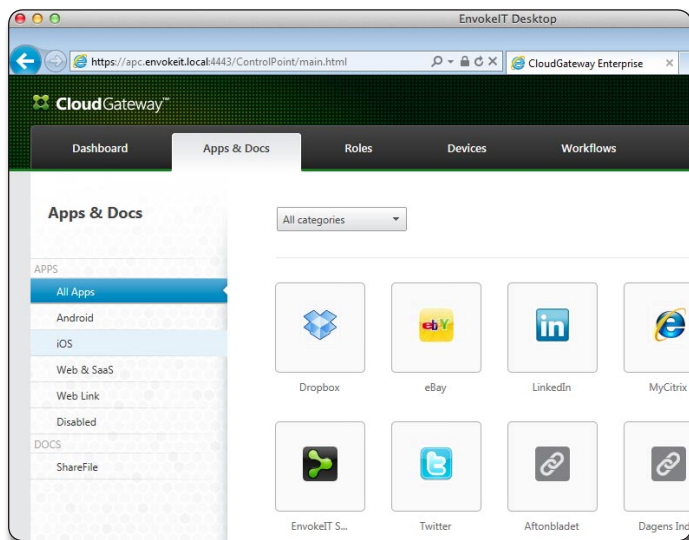


Figura 7. Screenshot do software da Citrix

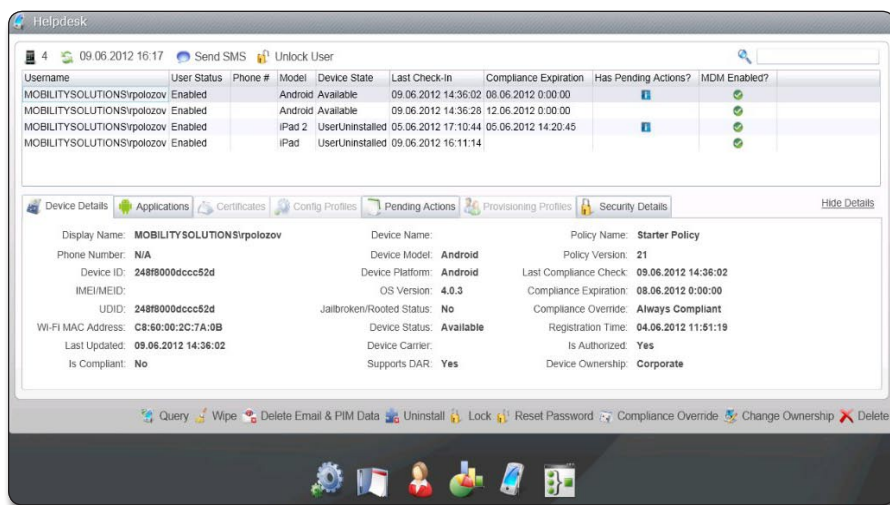


Figura 8. Screenshot do software da McAfee

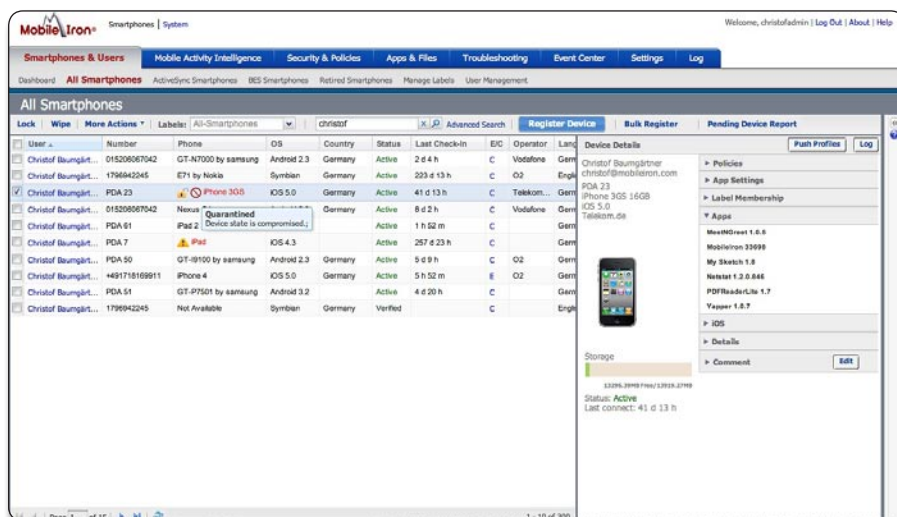


Figura 9. Screenshot do software da MobileIron

informações dos dispositivos, além de permitir que o usuário final possua uma central de autoatendimento.

A ferramenta gerencia o inventário de aplicativos e oferece a capacidade de criar uma loja de aplicações corporativa, fornecendo proteção contra aplicações maliciosas. Tem uma estrutura robusta que suporta até 100.000 dispositivos conectados para cada console de administração, permitindo o registro simultâneo de aparelhos e download de 30.000 aplicações concorrentes. Veja na **Figura 9** a interface de gerenciamento destes equipamentos.

O controle de segurança tem conexão via ActiveSync e inclui política de privacidade, distribuição de certificados, recuperação de dados de equipamentos extraviados e aviso em tempo real de roaming. O sistema integra-se aos serviços de diretórios, ao serviço de autoridade certificadora, à aplicação da BlackBerry (BES) e a algumas API de web services.

9 – Symantec Mobile Management (SMM)

O Symantec Mobile Management 7.2 é uma solução que suporta as principais plataformas do mercado (iOS, Android e Windows Phone). Através da central de gerenciamento, vide **Figura 10**, é possível visualizar de forma centralizada informações dos dispositivos e obter acesso a ferramentas de configuração remota.

O SMM permite fazer instalações de aplicativos e configurações de e-mail remotamente, protegendo tanto os dispositivos móveis quanto os dados. Isso inclui a ativação de senhas e controles de acesso, o recurso que possibilita a separação dos dados corporativos e pessoais, autenticação com LDAP e status de jailbreak/roteamento.

A ferramenta envia aos dispositivos as configurações e requisitos para autenticação de e-mails, Wi-Fi e VPN configurando automaticamente estes serviços, e integra-se ao Symantec MPKI e às Autoridades de Certificação da Microsoft. Como diferencial, a ferramenta se integra a outras soluções da Symantec, como o Endpoint Protection e o IT Management Suite, além do Microsoft System Center e Exchange Client Access Server. Portanto, caso tenha em sua estrutura alguma destas soluções já implantadas, o SMM deve ser considerado como um dos MDMs a ser analisado, já que possui este facilitador de integração.

10 – Spiceworks

Spiceworks é uma solução que integra gerenciamento de rede, help desk, inventário, monitoramento, gerenciamento de usuários, estoque e mapeamento dos recursos da rede. Na sua próxima versão, ainda em beta, ele irá

integrar uma ferramenta de MDM; tudo isto a custo zero.

A ferramenta escolhida para esta nova funcionalidade de gerenciamento de dispositivos foi a MaaS360 e, através de um acordo com a Fiberlink, irá trazer para o ambiente as principais funcionalidades desta solução.

Com isto, será possível através do Spiceworks a configuração remota de senhas, e-mails, perfis, Wi-Fi, VPN, conexões remotas e a distribuição de aplicações.

Ainda não se tem informações muito detalhadas sobre como será a versão final da ferramenta, mas para os que já quiserem arriscar, é possível fazer o download da versão 7, em beta, para teste.

Conclusão

A implantação de um programa de BYOD envolve muito mais do que somente a instalação e configuração de um software de gerenciamento de dispositivos. Os recursos atualmente disponíveis no mercado devem ser avaliados e/ou adaptados segundo uma estratégia corporativa, que precisa estar definida a priori. Neste processo, é fundamental que a TI participe de todo o projeto, levando à administração da empresa as questões relacionadas à segurança da informação e aos riscos relacionados a este tipo de ação.

Neste artigo foram apenas introduzidas as questões pertinentes ao assunto, pois existem muitas variáveis que envolvem o BYOD. Além disso, as questões levantadas para elaboração e construção de uma estratégia corporativa foram tratadas de forma sucinta, e são apenas algumas dicas que devem ser estudadas e exploradas posteriormente.

Ainda neste sentido, vale ressaltar que existem importantes ferramentas de gestão, como o ITIL, Cobit e as práticas de gerenciamento de projetos, que irão auxiliar na gestão de ativos, mudanças, serviços, entre outros, sendo importantes fontes de apoio no planejamento e estruturação.

Como qualquer tema que envolve questões relacionadas à segurança da informação, o BYOD não é um assunto que possa ser esgotado tão facilmente. As ferramentas e recursos estão na fase inicial de aprimoramento, sendo que algumas soluções ainda estão surgindo.

Lembrando que este é um tema recente, as primeiras experiências com a adoção deste programa já começaram a ser disponibilizadas na rede, sendo elas boas fontes de pesquisa. Deste modo, amplie seu conhecimento, busque as informações disponíveis, aprimore e melhore os processos através dos erros já conhecidos. O intuito é levá-lo a avaliar o seu ambiente, examinar as soluções existentes, preparar e planejar as ações, envolver e promover a participação de outros departamentos e, por fim, implantar o BYOD.

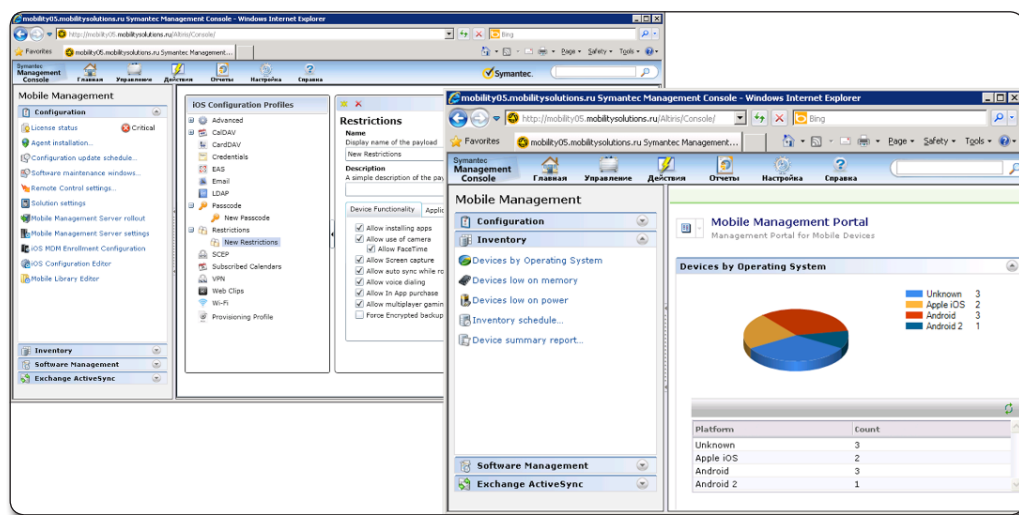


Figura 10. Screenshot do software da Symantec

Autora



Juliana Márcia Rezende Calado

jmrcalado@hotmail.com

Especialista em Gestão de Infraestrutura de TI pela Pontifícia Universidade Católica de Minas Gerais. Atua como professora de pós-graduação no curso de Gestão de TI pelo SENAC e pelo Centro Universitário UNA. Trabalha como Analista de TI na área de administração de serviços e gerência de redes. Experiência de 13 anos na área de redes de computadores e administração de servidores, além de três anos como docente de ensino superior na graduação e pós-graduação. Certificada Microsoft Certified System Administrator (MCSA), Microsoft Certified Technology Specialist (MCTS) e Microsoft Certified Desktop Support Technician, Information Technology Infrastructure (ITILv3) Foundation e Information Technology Infrastructure (ITILv3) Intermediário em Operação de Serviço (OSA).

Links:

AirWatch

<http://www.air-watch.com/br>

Amtel

<http://www.amtelnet.com>

Apptec360

<http://www.apptec360.com>

BlackBerry

<http://us.blackberry.com>

BYOD, a implantação inteligente é o novo desafio

<http://administradores.com.br/noticias/tecnologia/byod-a-implantacao-inteligente-e-o-novo-desafio/75980/>

BYOD security challenges: control and protect your most sensitive data

<http://www.sciencedirect.com/science/article/pii/S1353485812701113>

Citrix

<http://www.citrix.com>

Somos tão apaixonados por tecnologia que o nome da empresa diz tudo.

Porta 80 é o melhor que a Internet
pode oferecer para sua empresa.

Já completamos 8 anos e
estamos a caminho dos 80, junto
com nossos clientes.

Adoramos tecnologia.
Somos uma equipe composta
de gente que entende e
gosta do que faz,
assim como você.



Estrutura

100% NACIONAL.
Servidores de primeira
linha, links de alta
capacidade.



Suporte diferenciado

Treinamos nossa equipe
para fazer mais e melhor.
Muito além do esperado.



Serviços

Oferecemos a tecnologia
mais moderna, serviços
diferenciados e
antenados com as suas
necessidades.



1-to-1

Conhecemos nossos
clientes. Atendemos
cada necessidade de
forma única.
Conheça!



Porta 80
WEB HOSTING

Hospedagem | Cloud Computing | Dedicados | VoIP | Ecommerce |
Aplicações | Streaming | Email corporativo

porta80.com.br | comercial@porta80.com.br | twitter.com/porta80

SP 4063-8616 | RJ 4063-5092 | MG 4063-8120 | DF 4063-7486

Ele está trabalhando em seu saque e não na sua nova aplicação.



Mais de 95%* dos times de desenvolvimento e testes de aplicativos relatam tempos de espera e atrasos para acessar os sistemas que necessitam para realizar suas atividades. A Virtualização de Serviços elimina estas dependências gerando ambientes simulados similares à realidade, permitindo o desenvolvimento em paralelo. Isto significa que suas aplicações serão lançadas mais rapidamente, com maior qualidade e menor custo. Game over!

Conheça mais em ca.com/br/GoDevOps

*De acordo com o estudo 2012 North America/Europe Service Virtualization Study, Coleman-Parkes.

