OpenVAS 4

Análise detalhada

O projeto OpenVAS acaba de lançar a versão 4 de seu sistema de avaliação de vulnerabilidades — razão suficiente para verificar os novos recursos e aprender, na prática, como criar sua própria solução de checagem para seus sistemas.

por Stefan Schwarzer

O OpenVAS é um sistema de código aberto para a avaliação de vulnerabilidades, distribuído sob a licença GPL. Ele é um ambiente completo de avaliação de segurança, com uma série de serviços e componentes que podem ser organizados em diversas formas para construir um ambiente de avaliação adequado à sua rede. O *OpenVAS Manager* age como um serviço central para o controle do ambiente OpenVAS, comunicando-se com os componentes responsáveis pela varredura através do protocolo OMP (*OpenVAS Management Protocol* – Protocolo de Gestão do OpenVAS) baseado em XML e armazenando os resultados em uma conveniente base de dados do tipo SQLite.

O OpenVAS[1] está passando, atualmente, por rápidas mudanças e acaba de atingir a versão 4. Neste artigo apresentarei o sistema de varredura e uma visão do sistema de clientes e suas vulnerabilidades. Você também receberá algumas dicas para a implementação do OpenVAS e sobre como escrever seus próprios plugins.

Figura 1 O Desktop Greenbone Security é baseado no Qt.

Novos recursos

O ambiente OpenVAS mudou consideravelmente desde a sua versão 3, mas continua compatível com as ferramentas e protocolos conhecidos da versão 2. Além do novo OMP, que auxilia na gestão dos processos de varredura, o *OpenVAS Administration Protocol* (OAP, Protocolo de Administração do OpenVAS) foi adicionado para administrar clientes e novos serviços. O OpenVAS Manager cuida da comunicação dentro do ambiente e armazena todos os dados de varredura em sua base de dados interna SQLite, aliviando parte da carga dos novos clientes baseados em OMP e auxiliando na organização de acessos

simultâneos aos dados de varredura armazenados. A comunicação com o serviço responsável pela varredura (o scanner `openvassd`) ainda usa o *OpenVAS Transfer Protocol* (OTP, Protocolo de Transferência OpenVAS). O serviço OpenVAS Administrator fornece aos administradores uma abordagem conveniente à gestão de usuários e certificados.

Novos clientes de varredura estão agora disponíveis, em adição ao anteriormente usado OpenVAS Client, baseado em Gtk. Um servidor, chamado Greenbone Security Assistant (GSA) [2] é executado em um navegador. O aplicativo desktop, Greenbone Security Desktop (GSD, **figura 1**) [3], que é executado em Windows (graças ao Qt), ou a ferramenta simples para linha de comando `omp` completam a seleção de ofertas livres.

Os clientes desktop, que eram incompletos na versão 3, hoje cobrem o escopo completo de funções do OpenVAS. Em um ambiente de produção, o cliente clássico OpenVAS ainda é o mais fácil e rápido. Os pontos fortes são a organização hierárquica dos clientes e sua habilidade de organizar alvos de varredura em Alvos (*Targets*) e Escopos (*Scopes*). Após definir as propriedades padrão para a varredura, você pode atribuir as propriedades de um alvo para um escopo e, a partir daí, modificá-las conforme o necessário.

Este recurso facilita a modelagem e a realização rápida de novas varreduras e ajuda muito na seleção entre alvos de varredura e seus resultados. A definição de uma varredura em clientes web ou desktop é, comparativamente, confusa. O processo começa com as definições individualizadas de configurações de varredura, credenciais, alvos de varredura e tarefas.

A promessa de permitir módulos reutilizáveis acaba tornando-se uma armadilha quando há a necessidade posterior de modificar a configuração. Por exemplo, você não pode, retrospectivamente, modificar ou substituir os alvos em suas tarefas de varredura – isto significa que você deve criar novas entradas em sua configuração, o que pode ser bastante desagradável ao longo do tempo. Novos clientes não suportam, atualmente, o uso de listas baseadas em arquivos para a definição de alvos de varredura. Em outras palavras, caso você necessite analisar centenas de máquinas virtuais para descobrir as vulnerabilidades de um servidor web, irá apreciar o cliente clássico que apenas precisa de uma lista de hosts em um arquivo texto.

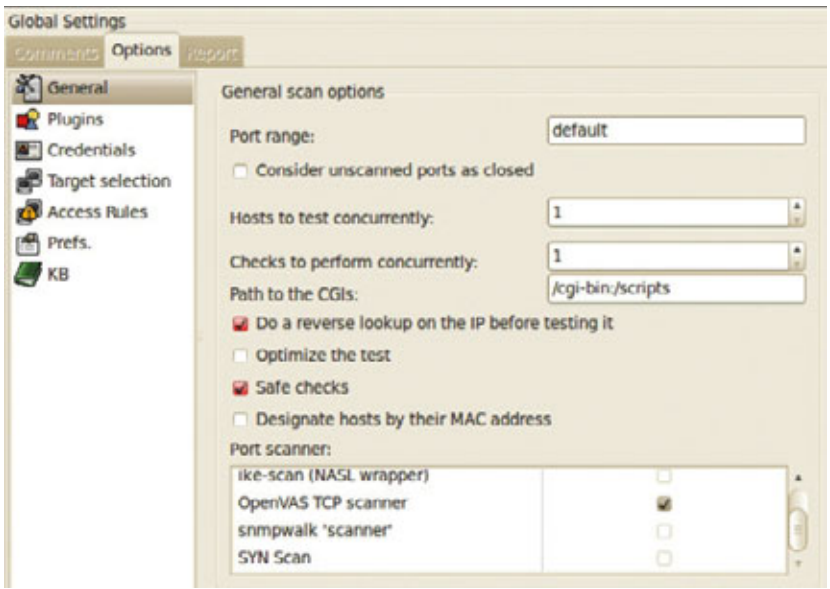

Figura 2 Configurações globais importantes no cliente OpenVAS.

O começo

Antes de você começar a usar o sistema de avaliação de vulnerabilidades, dedique um tempo para conhecer os formatos de dados do OpenVAS e limpar os espaços de memória para garantir que você possui a configuração correta para as suas preferências mais importantes. Isto também ajuda a evitar erros durante o armazenamento de informações confidenciais. Caso esteja trabalhando com o cliente clássico OpenVAS – e eu assumo que você está – a estrutura hierárquica dos alvos de varredura será de grande ajuda, já que você

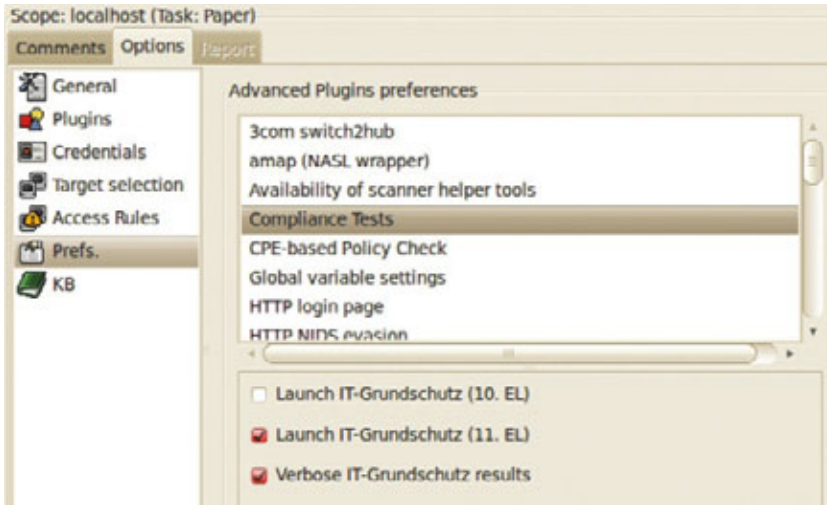

Figura 3 Preferências para os plugins individuais.

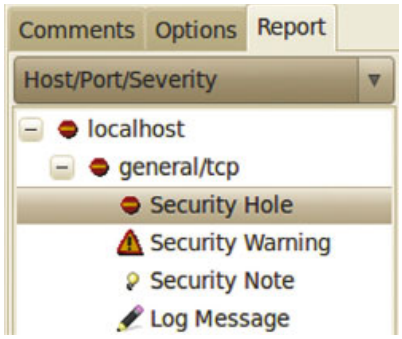

Figura 4 Relatórios de varredura.

pode aplicar algumas de suas preferências para outros alvos.

É possível tomar vantagem de seu conhecimento para modificar a hierarquia de alvos de varredura de forma retrospectiva – para fazer isto, simplesmente mova os arquivos no sistema de arquivos. O arquivo `.openvasrc` é particularmente interessante para cada alvo, já que todas as configurações da varredura estão nele, incluindo as opções que não podem ser configuradas nos clientes. Por exemplo, `log_whole_attack = no` desabilita totalmente o registro de varreduras no servidor central de varredura.

As figuras 2 e **3** mostram as preferências globais que você deve verificar antes da varredura. Por exemplo, se você está testando servidores web virtuais, você deve habilitar a resolução reversa de endereço (reverse lookup) para evitar receber todos os resultados agrupados sob um único endereço IP compartilhado. Inicialmente, a opção Safe checks é bastante desejável, já que as varreduras de DoD não podem estar entre os primeiros testes realizados. Infelizmente, esta opção está desabilitada como padrão. Você também pode desejar reduzir o número de máquinas analisadas em paralelo e o número de testes a serem completados para evitar cargas desnecessárias na rede e nas máquinas alvo.

Ao coordenar suas atividades com as do administrador de rede você deve certificar-se de que nenhum firewall ou sistema de detecção de invasão bloqueie suas varreduras e produza resultados inúteis. Inicialmente, ao menos a varredura *OpenVAS TCP* é uma boa escolha. Quando utilizar o Nmap para varrer as portas você deve verificar as configurações necessárias logo de início, usando o Nmap em um processo à parte e então aplicando os resultados.

Os plugins definem a interface cliente OpenVAS e são configurados em *Prefs* na aba *Options*. É uma boa ideia verificá-los depois de carregar ou atualizar os plugins. Por exemplo, você pode habilitar checagens bastante informativas em sincronia o Departamento Federal Alemão para Segurança da Informação (*BSI IT-Grundschutz*, o escritório de segurança online da Alemanha) [4] como testes de conformidade (*Compliance Tests* – **figura 3**). Este conveniente recurso de configuração de plugin pode ser aproveitada em seu próprio desenvolvimento de plugins.

Listagem 1: Compilando e Executando o OpenVAS

```
01 make depend   # instala os pacotes exigidos
02 make          # compila e instala o OpenVAS
03 make initial  # cria certificados, base de dados, etc.
04 make start    # inicia os processos necessários em background
```

Listagem 2: Uma modificação (override) severa

```
01 <severity_override name="IT Grundschutz M5.064:Secure Shell"
02     host="192.168.178.46"
03     port="general/IT Grundshutz"
04     OID="1.3.6.1.4.1.25623.1.0.895064"
05     severity_from="Security Hole"
06     severity_to="False Positive"
07     active="true">
08     <reason>
09     Backport
10     </reason>
11 </severity_override>
```

Arquivo	Significado
`~/.openvas`	Diretório para os arquivos de configuração dos alvos de varredura.
`~/.openvasrc`	Parâmetros globais de configuração.
`~/.openvas/Scope/Target`	Todas as configurações (.openvasrc) para este alvo de varredura e os resultados da mesma (Report_date). Na primeira vez em que você cria um novo alvo de varredura, o OpenVAS copia a configuração do escopo (Scope) para o alvo (Target).
`~/.openvas/severity_overrides.xml`	O arquivo XML com as modificações que influenciam os resultados da varredura quando avaliando seu efeito.

Tabela 1 Localização dos arquivos de configuração.

Acerte o Alvo

A escolha óbvia de um alvo inicial (*Target selection*), logo após a instalação do cliente ou servidor é o próprio `localhost`, o mesmo servidor onde o OpenVAS foi instalado. A configuração da máquina é conhecida e você pode facilmente verificar suas vulnerabilidades e repetir os testes. Caso o cliente e o servidor sejam executados em máquinas diferentes, você sempre poderá escolher seu próprio sistema cliente como o alvo inicial. Você deve levar em consideração a

rede que conecta as duas máquinas. Também faz sentido configurar sua própria máquina virtual para testes uma vez que pode ter controle total sobre essa máquina (por exemplo, com o VirtualBox [5]), podendo facilmente modificá-la e mantendo vários estados como snapshots.

Após a seleção dos plugins – existem mais de 21.000 atualmente – você pode executar a primeira varredura. Comece pela seleção de uma visão exterior completa do alvo da varredura; ou seja, não forneça nenhuma credencial de login para o sistema alvo já que esta seria a situação inicial ao confrontar um invasor. Dependendo das portas abertas e serviços ativos, a varredura pode levar desde alguns minutos até várias horas, mas você pode interromper o processo a qualquer momento. Após completar ou interromper a varredura você terá um relatório listando os detalhes com base nas portas – ou serviços – para o seu alvo.

Três tipos de informações são exibidas na aba *Report* do cliente OpenVAS (**figura 4**): *Security Hole* aponta as potenciais vulnerabilidades descobertas pela varredura; *Security Note* exibe informações sobre a porta para a avaliação do sistema; *Log Message* contém a informação que é retornada pelo plugin, o que pode ser, de fato, muito útil. Comece verificando a informação geral em *general/tcp*, especialmente os parâmetros agregados para a varredura, tais como configuração, duração e assim por diante.

Vulnerabilidades

Mesmo que você tenha aplicado as últimas atualizações, a primeira varredura tipicamente revela dezenas de vulnerabilidades críticas, principalmente causadas pelo uso incorreto de plugins de varredura que não são muito tolerantes a erros. Por exemplo, as varreduras IT-Grundschutz [4] apenas irão retornar resultados

Figura 5 Modificação de severidade para um falso positivo.

significativos se puderem ter acesso interno ao sistema alvo, o que requer um login. Sem isto os testes falham e, infelizmente, retornam um estado que incrementa o contador de vulnerabilidades críticas de segurança.

Listagem 3: Fragmento do unibwm1.nasl

```
01 if(description)
02 {
03   script_id(9999001);
04   script_version("$Revision: 7732 $")
05   script_tag(name:"risk_factor", value:"None");
06   script_name("UniBwM 1: Plaintext passwords");
07   desc ="
08   Panorama : Esse script testa a existência de senhas
➥ em texto puro para o subversion
09   Fator de risco : Nenhum";
10
11   script_description(desc);
12   script_summary("Plaintext passwords");
13   script_category(ACT_GATHER_INFO);
14   script_copyright("Copyright (C) 2010 Stefan Schwarz");
15   script_family("General");
16   script_dependencies("ssh_authorization.nasl");
17   if(description)
18 {
19   script_id(9999001);
20   script_version("$Revision: 289 $");
21   script_tag(name:"risk_factor", value:"None");
22   script_name("UniBwM 1: Plaintext passwords");
23   desc ="
24   Panorama : Esse script testa a existência de senhas
➥ em texto puro para o subversion
25   Fator de risco : Nenhum";
26
27   script_description(desc);
28   script_summary("Plaintext passwords");
29   script_category(ACT_GATHER_INFO);
30   script_copyright("Copyright (C) 2010 Stefan Schwarz");
31   script_family("General");
32   script_dependencies("ssh_authorization.nasl",
➥ "gather-package-list.nasl");
33   script_add_preference(name:"Launch this script",
➥ type:"checkbox", value:"yes");
34   exit(0);
35 }
```

Figura 6 Propriedades do plugin cliente OpenVAS.

Em outras palavras, você precisará desabilitar estes plugins ou fornecer as credenciais necessárias antes de executar o próximo teste. Uma rápida olhada nas estatísticas pode, tipicamente, levar a falsas interpretações, tal como a mensagem *OpenVAS isn't (yet) set up for fully automated testing and evaluation* (O OpenVas não está ainda configurado para testes e avaliações automáticas).

Falsos positivos

Outro problema frequente é que os plugins tenderão a identificar o que eles pensam que são números obsoletos de versões, tipicamente em sistemas com backports [6]. Consertos para vulnerabilidades descobertas em versões mais novas são, frequentemente, fornecidos também para sistemas antigos, mas ainda oficialmente suportados. Mas isto não quer dizer que um backport cuja vulnerabilidade foi resolvida ganhe um novo número de versão, o que confunde o OpenVAS. Uma varredura de segurança não verifica as vulnerabilidades, ela apenas busca por versões com vulnerabilidades conhecidas.

O servidor Ubuntu 8.04 é um exemplo disto: o OpenVAS classifica a versão do OpenSSH utilizada por ele como crítica, mas se você pesquisar por esta versão e as correções aplicadas a ela você descobrirá que `ssh-2.0-openssh_4.7p1 debian-8ubuntu1.2` está atualizada e que todas as correções de segurança foram aplicadas [7]. Os plugins são facilmente enganados por pequenas diferenças como esta, tornando necessária a limpeza manual das mensagens.

Para reduzir totalmente os falsos positivos você pode mudar a opção *Report paranoia* dentro de *Global variable settings* de *Normal* para *Avoid false alarms*. Claro que isto apenas funciona para os plugins que aceitam esta opção e apenas 130 dentre os mais de 21.000 plugins estão entre os que fazem hoje. Dito isto, você pode preferir evitar o grande risco de deixar passar uma vulnerabilidade real.

Você pode utilizar o mecanismo de modificação (*override*) de severidade para alterar a criticidade de uma vulnerabilidade identificada pelo OpenVAS, reclassificando a mensagem gerada pela varredura de *Security Hole para False Positive*. Estas modificações ficam armazenadas globalmente no arquivo `severity_overrides.xml` (**tabela 1**), onde você pode, então, editá-las retrospectivamente (**listagem 2** e **figura 5**) e aplicá-las apropriadamente aos resultados de um relatório.

Estas modificações (*overrides*) são, inicialmente, relativas à máquina coberta pelo relatório – 192.168.178.46 neste exemplo. Caso você deseje estendê-las a um grupo de máquinas, ou todas as máquinas, basta apenas substituir a entrada `host=` para `192.168.178.*`. Mas tenha cuidado: esta mudança irá influenciar toda a regra e irá fazer com que o OpenVAS assuma um falso positivo mesmo que uma vulnerabilidade genuína exista. Em outras palavras, você precisa verificar as regras que você modificou em intervalos regulares.

Visualizações

O OpenVAS distingue as várias visualizações de um objeto alvo para varredura:

1. Visualização a partir da Internet pública;
2. Visualização a partir da rede interna de uma empresa, possível a partir de várias zonas de segurança;
3. Visualização a partir de dentro do próprio objeto alvo, após o login.

As visualizações 1 e 2 diferem apenas em relação à posição ocupada pelo scanner do OpenVAS; não há diferenças sob o ponto de vista do cliente. O cliente OpenVAS pode abrir uma conexão para qualquer número de scanners e utilizá-los de forma paralela. Você deve estar ciente das políticas de segurança na infraestrutura da rede em questão. Mesmo a varredura de seu próprio servidor na zona pública de sua rede corporativa pode tornar-se um desafio.

Dentro do alvo

Varreduras do OpenVAS que têm a habilidade de efetuar um login na máquina alvo e analisá-la a partir de seu interior produzem resultados muito melhores. O login não deve ser o de uma conta administrativa (*root*); uma conta não privilegiada irá, tipicamente, revelar informações suficientes sobre o sistema e suas vulnerabilidades.

Os pares usuário-senha nos sistemas alvos são algo que você desejará proteger; infelizmente o OpenVAS não faz um bom trabalho neste caso. Ao invés disto, ele armazena as senhas sem criptografia, em modo texto, na sua base de dados SQLite, tornando-as acessíveis ao menos ao administrador do servidor de varreduras, que pode não ser necessariamente o administrador de todos os sistemas alvo. Um simples comando SQL como

```
sqlite3 tasks.db 'select * from
➥ lsc_credentials;'
```

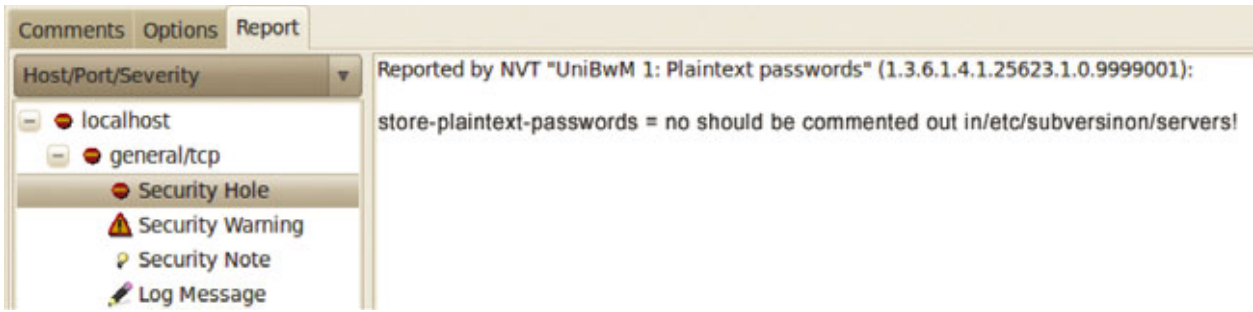

Figura 7 O cliente OpenVAS exibindo os resultados de varredura coletados pelo plugin exemplo.

revela toda esta informação confidencial a qualquer um com acesso à base de dados. A criptografia das credenciais antes de seu armazenamento não é uma solução, uma vez que estes dados seriam decriptados durante a varredura e, para fazer isto, você precisaria armazenar uma chave de criptografia em algum lugar no apliativo

O cliente clássico OpenVAS revela alguma fraqueza aqui; ele armazena as credenciais de acesso em seu arquivo de configuração local, `.openvasrc` (**tabela 1**). Mesmo que você tenha sido avisado sobre isto, a alternativa de armazenar a palavra-chave para sua chave privada é igualmente assustadora. Uma solução funcional seria armazenar os dados criptografados combinados a uma senha que você não precise armazenar na aplicação, ou possivelmente usar o suporte a smartcard. Esta fraqueza fornece um amplo escopo de desenvolvimento para as versões posteriores do OpenVAS.

Segurança de chaves

Chaves assimétricas fornecem uma solução para a distribuição das credenciais de segurança por múltiplos sistemas mitigando, assim, o problema com um repositório centralizado. A chave pública deve residir no sistema alvo. A conta deve ter sua senha desabilitada, de forma que a chave privada no cliente seja a única forma de acessá-lo. Como você pode apenas armazenar a palavra-chave às claras, novamente, muitos administradores desejarão evitar esta abordagem. A única forma de proteger a chave privada é mantê-la em uma memória segura (por exemplo, um pendrive USB) ou criptografar a própria chave utilizando TrueCrypt, PGP ou Encrypt-FS.

O importante é que o cliente OpenVAS apenas atualiza a credencial de acesso modificada quando ele varre o objeto alvo. Mesmo que o cliente apague as credenciais de acesso antes de encerrar o programa, elas ainda estarão disponíveis no arquivo de configuração. Assim, os administradores dos sistemas a serem sondados podem ter acesso às credenciais, mesmo que a chave privada esteja armazenada de forma central.

Uma vez que a chave pública correspondente está localizada no sistema alvo (em `~./ssh/authorized_keys2` no Linux), os administradores podem, eles mesmos, controlar este processo. Apesar de ter a chave pública e privada, o acesso ao sistema alvo ainda é negado. Um plugin especial para o OpenVAS pode restringir o acesso a uma única varredura através da mudança da localização do arquivo com as chaves após a varredura com o seguinte comando,

```
pread(cmd: "/bin/mv", argv:
➥ make_list ("mv","~/.ssh/
➥ authorized_keys2", "~/.ssh/
➥ authorized_keys2.bak"));
```

que é equivalente a permitir o acesso à varredura apenas uma vez.

O processo para a criação dos pares de chaves é similar ao processo usado no OpenSSH [8]. Entretanto, ao contrário do que diz a documentação do OpenVAS, apenas chaves do tipo DSA funcionam hoje, não RSA [9]. Em outras palavras, não é uma boa ideia usar as ferramentas de criação de credenciais existentes no pacote interno (em *Extras/LSC Credentials Manager*).

O método requer apenas a chave pública para a conta desejada (por exemplo `sshovas` em `~/.ssh/authorized_keys2`). Depois disto, todos os sistemas nos quais você precisa efetuar a varredura interna devem estar acessíveis através do comando

```
ssh -i key-directory/
➥ sshovas_dsa.p8 sshovas@target
```

Para testar, habilite o plugin SSH Authorization. Após a varredura você deve observar a confirmação de que as verificações de segurança estão habilitadas em *SSH/Security Node*.

Como alguns plugins são escritos de forma dependente da linguagem de programação usada – o plugin para a varredura do VMWare reconhece apenas a cadeia de caracteres de retorno `command not found` – faz sentido usar o ambiente na língua inglesa, por exemplo configurando `LANG=us` em `~/.profile`.

plugin apara varreduras de segurança

Ainda que o OpenVAS tenha, atualmente, cerca de 21.000 plugins, pode ser que você precise desen-

volver o seu para que ele esteja em conformidade com a política de segurança de sua empresa. Você precisará usar a linguagem NASL (a linguagem de script do Nessus, Nessus Attack Scripting Language) para isso. Vários manuais estão disponíveis para ajudá-lo a aprender esta linguagem [10] [11] e você pode usar os plugins existentes como modelo. Um artigo em uma das edições prévias da *Linux Magazine* [12] descreve a estrutura dos plugins NASL. O código fonte para o exemplo discutido aqui está disponível para download [13] também com uma tradução para o inglês [14].

O exemplo implementa um requisito de segurança que previne o armazenamento aberto em texto de senhas em servidores Subversion [15]. O script deve procurar por configurações clientes que não previnam, explicitamente, o armazenamento não criptografado de senhas. Para permitir que isto aconteça, o script verifica os arquivos de configuração global do Subversion em `/etc/subversion` para conferir se as diretivas `store-passwords = no` ou `store-plain-text-passwords = no` existem. Esta diretiva não pode estar desabilitada, o que, infelizmente, é o caso na instalação padrão.

Os detalhes do cabeçalho obrigatório do plugin (descrição), que são consultados pelos clientes (**listagem 3**), permitem a você selecionar os plugins relevantes para a sua varredura e configurar suas preferências. As instruções em `script_dependencies` garantem que o OpenVAS irá executar os pluguins requeridos e acessar os resultados e tarefas preliminares (neste caso, um login via SSH e a lista de pacotes instalados).

A especificação de um ID único (`script_id`) irá causar, inicialmente, um problema já que o OpenVAS não possui, atualmente, um esquema unificado de numeração. Hoje nenhum plugin utiliza o novo ID OpenVAS, `script_oid`, introduzido na versão 1.0.3; ao invés disto eles ainda usam a numeração simples com `script_id`. Dito isto, o OpenVAS converte, internamente, o antigo esquema para o novo (**figura 6**).

Para encontrar um intervalo numérico que sirva para seus plugins, você pode usar o Awk para descobrir os números já utilizados no diretório de plugins:

```
find . -type f -print | xargs
➥ fgrep script_id | awk -F '[()]'
➥ '{print $2}' | sort -n
[...]
2000201
9000001
9999991
9999992
```

Depois de fazer isto, eu decidi usar o número 9999001 para o exemplo. O uso de `script_add_preference` cria uma caixa de checagem abaixo das opções *Prefs* do cliente OpenVAS para que o novo plugin seja habilitado.

Quadro 1: Versões, pacotes e componentes

Os nomes dos componentes do OpenVAS 4 são um tanto confusos. Por exemplo, o número de versão está apenas relacionado com as bibliotecas (atualmente 4.0.5); os componentes scanner (3.2.4) e manager (2.0.4) usam outros números [16]. Os repositórios disponibilizados pela maioria das distribuições Linux atuais ainda parecem manter os binários antigos; o Ubuntu 11.04 ainda oferece o OpenVAS2. A comunidade, entretanto, oferece pacotes mais recentes; por exemplo, o openSUSE Build Service [17] mantém pacotes para Debian e Ubuntu.

Os pacotes do OpenVAS 4 não mais mantêm o cliente OpenVAS nativo apreciado por muitos usuários por ser mais fácil de usar e manter dados locais de configuração. Caso você use seu gerenciador de pacotes para instalar o cliente, você terá como resultado uma versão obsoleta. Ao invés de comprar da Greenbone um aplicativo de varredura completamente suportado [18] em sua versão atualizada, os usuários interessados podem baixar os códigos fonte mais recentes a partir do repositório [19] e compilar as ferramentas que necessitam.

Se você gosta de trabalhar com a versão em seu estado da arte, você deve considerar obter o código através do Subversion. É relativamente fácil compilar o pacote inteiro. Eu escrevi um Makefile para isto e o testei no Ubuntu [13]. O arquivo cuida do download e atualização do código, instala os pacotes necessários e cria e instala a versão atual e suas atualizações (**listagem 1**). Após completar estes passos, o comando `make up` atualiza o software sempre que você desejar.

Quadro 2: Varreduras de provedores

Provedores de serviços de hospedagem estabelecidos estão plenamente cientes do problema de varreduras de rede e desenvolveram (em alguns casos, muito variadas) estratégias de prevenção. Grandes provedores identificaram os padrões típicos de varredura com bastante confiabilidade no tráfego de rede e automaticamente desconectam, momentaneamente, o alvo de varredura.

A experiência mostra que ferramentas populares ainda retornam bons resultados quando usadas na Internet [20]. O OpenVAS oferece escolhas de varredura de portas que você pode configurar suficientemente bem na opção *Prefs*. Caso isto não funcione, você pode usar um sistema externo de varredura de portas e passar seus resultados ao OpenVAS. Isto torna supérflua a varredura de portas conhecidas nos servidores web de sua empresa e você pode desabilitá-la definindo padrões de varredura estática de portas no OpenVAS.

O sistema alvo

A próxima parte do plugin abre uma conexão protegida por SSH ao sistema alvo. Infelizmente, a troca de informação genérica não funciona em um teste local já que os plugins falham em comunicar-se através da base de conhecimento interna. Em outras palavras, você precisará configurar a conexão SSH explicitamente para os testes locais.

Você não pode usar a conexão existente através de `ssh-authorization.nasl` porque `ssh_login_or_reuse_connection` não irá retornar um socket válido. Infelizmente, você precisa armazernar os dados de conexão às claras – e removê-los novamente depois de completar o plugin. Novamente, você pode avaliar a opção `Launch this script` do plugin.

Para investigar o sistema alvo, o plugin pode enviar comandos Shell arbitrários através de `ssh_cmd` e investigar a sua saída. Funções internas como `egrep` também estão disponíveis [16]. A saída do plugin – ou seja, a classificação da vulnerabilidade – é tratada por `security_note`, `security_warning`, ou `security_hole`.

Os plugins NASL são armazenados em `Install-Directory/lib/openvas/plugins` ou em seus subdiretórios. O teste da sintaxe e lógica do seu plugin deve ser feito, primeiramente, fora do ambiente OpenVAS. O comando

```
openvas-nasl -X -i .. unibwm1.nasl
```

trata disto. Você precisa da opção `-X` aqui porque o plugin não está assinado. A opção `-i` aponta o arquivos para os scripts dependentes no diretório mencionado.

Após completar o teste com sucesso, você pode oferecer o plugin para seus clientes. Para isto, você precisa atualizar o serviço de varredura `openvassd` enviando a ele o sinal `kill -HUP`. De agora em diante, qualquer cliente que abra uma nova conexão para o scanner irá receber a atualização do plugin. Depois de selecionar o plugin e executar a varredura no sistema alvo, os resultados aparecerão no relatório (**figura 7**). ■

Mais informações

[1] OpenVAS software - Clientes, serviços e protocolos: http://www.openvas.org/about-software.html

[2] Greenbone: Gestão de Vulnerabilidade com o GSM: http://greenbone.net/learningcenter/vm_with_gsm.html

[3] Greenbone Desktop Suite para Windows: http://www.greenbone.net/technology/gds.html

[4] IT-Grundschutz: https://www.bsi.bund.de/EN/Topics/ITGrundschutz/itgrundschutz_node.html

[5] VirtualBox: http://www.virtualbox.org/

[6] Backports no desenvolvimento de software: http://en.wikipedia.org/wiki/Backport

[7] Gestão de patches no Ubuntu usando o OpenSSH como exemplo: http://packages.ubuntu.com/hardy/openssh-server

[8] Executando testes locais de segurança: http://www.openvas.org/performing_lsc.html

[9] Erros SSH durante a varredura quando da utilização de "pares de chaves público-privada" por JohnBradley: http://wald.intevation.org/tracker/index.php?func=detail&aid=1502&group_id=29&atid=220

[10] "The NASL2 reference manual" por Michel Arboi: http://michel.arboi.free.nasl2ref/fr/nasl2_reference.pdf

[11] "Writing Nasl Scripts" (sumário) por Hemil Shah: http://www.infosecwriters.com/text_resources/NASL_HShah.pdf

[12] "O farejador de vulnerabilidades OpenVAS" por Geoff Gallitz e Tim Brown, Linux Magazine, Junho de 2010, pp. 34-39

[13] Makefile e código fonte: https://subversion.unibw.de/public/openvas

[14] unibwm1.nasl com tradução para o inglês: http://www.linux-magazine.com/Resources/Article-Code

[15] Subversion: http://subversion.apache.org/

[16] Pacotes OpenVAS: http://www.openvas.org/install-packages.html

[17] openSUSE Build Service: http://en.opensuse.org/Build_Service

[18] Greenbone Networks: http://greenbone.net/

[19] Repositório OpenVAS: https://svn.wald.intevation.org/svn/openvas/trunk/

[20] Nmap: "Scanning the Internet" por Gordon Lyon (Fyodor), apresentação feita na Black Hat e Defcon 2008: http://insecure.org/presentations/BHDC08/

Gostou do artigo?

Queremos ouvir sua opinião. Fale conosco em cartas@linuxmagazine.com.br

Este artigo no nosso site: http://lnm.com.br/article/5885