Parte 3

Métodos de Testar Vulnerabilidades

Identificando métodos para coleta de “inteligência”

- Prova de Conceito
- Código de Exploração
- Ferramentas Automatizadas
- Versioning

Prova de Conceito

- Usado para demonstrar a existência de uma vulnerabilidade.
- Fornecido por um fornecedor ou um pesquisador em um fórum aberto.
- É uma demonstração do problema através de um bloco pequeno de código que não explora um sistema em benefício do atacante.

Prova de Conceito

- Não é um descrição que mostra ao usuário como reproduzir o problema.
- Usada para identificar a origem do problema e recomendar uma solução.
- Usada para notificar a comunidade de segurança, sobre um problema.

Prova de Conceito

- O objetivo é ganhar tempo entre o momento em que a vulnerabilidade é anunciada e o momento em que usuários maliciosos começam a produzir código para tirar proveito dessa vulnerabilidade e lançar ataques.

Prova de Conceito

- Um fornecedor é beneficiado a fim de que tenha tempo para produzir e lançar um *path* para o problema.

Código de Exploração

- Um programa escrito para tirar vantagem de um problema em alguma parte de um software.

- É elaborado para mostrar mais detalhes de como uma vulnerabilidade pode ser explorada.

Código de Exploração

- Pode ser escrito em várias linguagens.
- Fornece à comunidade de segurança, um programa para demonstrar uma vulnerabilidade, produzir algum ganho para o usuário executando o programa.

Código de Exploração

- Também torna possível o ataque a sistemas por usuários maliciosos.

- Oferece clareza na exploração da vulnerabilidade e motivação para fornecedores produzirem um *patch*.

Código de Exploração

- Lançar uma exploração funcional significa lançar um programa que tire proveito de um problema.

- Fóruns que divulgam detalhes técnicos sobre uma vulnerabilidade e compartilham explorações funcionais são observados por muitos membros, todos com as suas motivações.

Ferramentas Automatizadas

- São pacotes de software criados por fornecedores para permitir testes de segurança (descobertas de vulnerabilidades) de forma automática.
- Geram relatórios sobre lista detalhada dos problemas de vulnerabilidades com o sistema.
- Permitem usuários legítimos realizarem auditorias para controlarem o andamento da proteção do sistema.

Ferramentas Automatizadas

- Usuários maliciosos, com a mesma ferramenta, podem identificar vulnerabilidades em *hosts*.

- Usuários com pouco conhecimento técnico têm a capacidade de identificar e proteger seus hosts.

Ferramentas Automatizadas

- Gratuitamente disponível existe a Nessus (software livre para varredura de vulnerabilidades) da Nessus Project (www.nessus.org).

- De código proprietário temos: CyberCop Segurity Scanner (Network Associates), NetRecon (Symantec), Internet Scanner (Internet Security Systems), IIS-Scan (...), Languard-Scanners (...), Iris (...).

Versioning

- Consiste em identificar versões ou revisões de software que um sistema está usando.
- **Muitos pacotes de software possuem uma versão** (Windows 2000 Professional ou Solaris 8).
- E muitos **pacotes de software incluídos em um software versionalizado**, também incluem uma versão (como uma distribuição LINUX, que é uma coleção de pacotes de software, cada um com sua próprias versões).

Versioning

- O método consiste na observação de uma lista de pacotes de software com versões anunciadas contendo vulnerabilidades, por um fornecedor.
- Um método de realizar Versioning é executar o comando de versão em um pacote de software:
- Como, por exemplo, o comando *uname –a* que fornece a revisão de kernel rodando em uma máquina LINUX.

Versioning

- Outro modo é usar uma ferramenta de gerenciamento de *patch*, provida por um fornecedor para verificar o software pela última versão.

- Por exemplo, executar a ferramenta *showrev –p* em um sistema SUN SOLARIS).

Identificando métodos para coleta de “inteligência”

- **Técnicas de Pesquisa Comum**:
  - Whois
    - baseado em serviço de nomes,
    - baseado em serviço de rede.
  - DNS
    - Ferramenta *Dig*
    - Ferramenta nslookup

Whois

- Um banco de dados Whois é uma coleção de informações gratuitamente disponível na Internet.
- Destinada a manter informações de contato para recursos de rede.
- Existem vários: Whois.com, Whois.biz, e o banco da American Registry of Internet Numbers, contendo informações baseadas em serviço de nome (datalhes sobre um domínio) e baseadas em serviço de rede (informações sobre o gerenciamento de redes).

Whois baseado em serviço de nomes

- Fornecem diversos detalhes sobre um domínio.
- Dados são fornecidos para facilitar a comunicação entre proprietários de domínio, no caso de um problema.

Whois baseado em serviço de nomes

- É o método apropriado para tratar os problemas que surgem, embora a tendência hoje pareça ser primeiro incomodar o provedor sobre um problema.

Whois baseado em serviço de nomes

- Nomes de domínio podem ser registrados através de diferentes registrars competindo:

- ENOM é um registrar, uma empresa que faz registros de informações sobre domínios em bancos de dados Whois.

Whois baseado em serviço de nomes

- Em um banco de dados **Whois baseado em serviço de nomes** pode-se ver:

  - informações de contato do proprietário de um domínio, como o nome, telefone, fax, endereço postal, servidores de nome de domínio.

Whois baseado em serviço de nomes

- Esses dados podem proporcionar a um usuário um método para atacar e controlar um domínio.

- Dados fornecidos incluem os servidores de nomes de domínio (DNS).

Whois baseado em serviço de rede

- Fornecem detalhes das informações de gerenciamento de redes.

- Esses dados podem auxiliar o pessoal de rede e de segurança, com informações para localizar um *host*, se um problema ocorrer.

Whois baseado em serviço de rede

- Fornece dados como o provedor de contato dos endereços de rede, e em alguns casos, alguma empresa alugando esse espaço de endereçamento.

Whois baseado em serviço de rede

- Em um banco de dados **Whois baseado em serviço de rede**, pode-se obter:

  - um espaço de endereços de rede (66.38.151.0 à 66.38.151.63)

  - proprietário desse espaço de endereços (por exemplo, uma empresa ou grupo empresarial) e

Whois baseado em serviço de rede

- ou em alguns casos uma empresa ocupando (alugando) esse espaço.

- Essas informações podem dar a um atacante as fronteiras para um possível ataque.

Whois baseado em serviço de rede

- Por exemplo, para obter informações:

  prompt:> whois –h whois.arin.net 66.38.151.10
- ARIN Registration Services

- Usar servidor Whois em rs.internic.net para informação relacionada a registros de domínios (Internet's Network Information Center).

- Usar whois.nic.mil para informação sobre NIPRNET (Internet Protocol Router Network) .

Ferramenta *dig*

- Disponível gratuitamente, distribuído como parte do pacote BIND (Berkeley Internet Name Domain) do Internet Software Consortium.
- Coleta informações dos servidores de DNS.
- Usado para traduzir nomes dos *hosts* em endereços IP, ou vice-versa.

Ferramenta *dig*

- Muitas explorações não possuem a capacidade de traduzir os nomes dos *hosts* e precisam de endereços numéricos para funcionar.

- Usado para para obter informações de versão de servidores DNS.

- Prompt:> dig @pi.cipherpunks.com TXT CHAOS version.bind

- TXT CHAOS version.bind são parâmetros para a busca do *dig*.

Ferramenta *dig*

- Os dados de nomeação do DNS são divididos em zonas.

- Dig pode realizar outros serviços de DNS, como por exemplo, uma transferência de zona, que é uma função do DNS para distribuir registros de serviço de nomes a outros *hosts*.

Ferramenta *dig*

- A base de dados do DNS é distribuída através de uma rede lógica de servidores.
- Cada servidor DNS retém parte da base de nomes, principalmente os dados para o domínio local.

Ferramenta *dig*

- Algumas das consultas são relativas a computadores no domínio local e são resolvidas por servidores dentro do domínio.

- Contudo, cada servidor registra os nomes dos domínios e endereços de outros servidores de nomes, de modo que consultas para fora do domínio podem ser resolvidas.

Ferramenta *dig*

- Efetuando manualmente uma transferência de zona, um atacante pode ganhar informações valiosas sobre sistemas e endereços gerenciados por um servidor de nomes DNS.

Ferramenta *nslookup*

- nslookup é acrônimo de Name Service Lookup.
- Gratuita no Internet Software Consortium.
- Procura informações nos hosts através do DNS e retorna informações sobre um domínio.
- Prompt:> nslookup

Identificando métodos para coleta de “inteligência”

- Nmap
- Indexação Web

Network Mapper

- Para um atacante ganhar acesso a um host, ele precisa ser lançado contra um serviço rodando no host.

- O serviço precisa ser vulnerável a um problema que permitirá ao atacante ganhar acesso.

Network Mapper

- É possível descobrir que serviços o host utiliza, através de alguns métodos de coleta de “inteligência”.

- Também é possível sondar portas em um host, com utilitários como *netcat* para saber se uma conexão pode ser feita com o serviço.

Network Mapper

- O processo de coletar informações sobre serviços disponíveis em um host é simplificado por ferramentas como o Network Mapper (Nmap).

- O Nmap identifica os serviços abertos, públicos e os serviços que estão sendo filtrados por TCP Wrappers ou firewalls.

Network Mapper

- Prompt:> nmap –sS –O derivative.cipherpunks.com
- Os flags -sS e -O instruem o Nmap a conduzir um scan SYN no host e identificar o sistema operacional através de respostas IP recebidas.

Indexação Web

- Conhecida como *spidering.*
- Programas automatizados para vasculhar *sites* (conjunto de páginas) e indexar os dados de links de páginas desses *sites* para um banco de dados:
- spiders, crawlers, agents, ...
- Esses programas visitam determinados *sites*, seguem links em páginas, e registram os dados desses links de cada página “visitada”.

Indexação Web

- spiders, crawlers, agents, ...

- Esses programas visitam determinados *sites*, seguem links em páginas, e registram os dados desses links de cada página “visitada”.

Indexação Web

- Os dados obtidos (referentes aos links) são indexados e referenciados em um banco de dados relacional, e associados ao mecanismo de busca.

- Quando um usuário visita um portal (Yahoo, WebCrawler, Google, AltaVista, Miner, ...), a pesquisa de palavras-chave retornará vários links para uma página indexada.

Indexação Web

- Mas, o que acontece quando informações vitais contidas em um *site* não são armazenadas com controle de acesso apropriado?

Indexação Web

- Como os dados dos *sites* estão em arquivos, e se esses arquivos não estão suficientemente protegidos, isso pode permitir que um atacante ganhe acesso às informações do *site* e obtenha "inteligência", através do mecanismo de busca.

Extração de Páginas Web

- Ferramentas como a *wget*, podem ser usadas para extrair recursivamente todas as páginas de um site.

- O processo é bem simples. Com wget :
- prompt:> wget –m –x http://www.mrchal.com

Extração de Páginas Web

- O flag -m (mirror) diz a *wget* para baixar todos os arquivos dentro do site www.mrhal.com seguindo os links.

- O flag -x preserva a estrutura de diretório do *site* quando for baixado.

Extração de Páginas Web

- Este tipo de ferramenta pode permitir a um atacante espelhar um site.

- Um outro programa, chamado *grep* http://www.gnu.org/software/grep/grep.html
- permite que um atacante procure links que possam ser de interesse.

Problemas de Pesquisa de Vulnerabilidades

- Existem várias maneiras de se abordar um problema.
- Qual delas escolhemos depende dos recursos disponíveis e da metodologia com que estamos mais acostumados.

Recursos

- No caso de pesquisa de vulnerabilidades: os recursos são:
  - código,
  - ferramentas,
  - tempo.

Métodos

- Pesquisa de Código-Fonte
- Engenharia Reversa
- Caixa-Preta

Vulnerabilidade

- Uma vulnerabilidade é um problema, explorável ou não, estando ela em um sistema operacional, um banco de dados, na lógica de uma aplicação ou numa interface Web.

Pesquisa, Metodologia

- Pesquisa é o processo de se reunir informações que podem levar ou não à descoberta de uma vulnerabilidade.

- Metodologia é o conjunto de métodos recomendados, que podem ser usados para pesquisar vulnerabilidades.

Amplitude de Métodos e Ferramentas

- Do entusiasta de segurança doméstico ao auditor de código corporativo, os métodos e ferramentas são os mesmos.
- Métodos variando de deduções a métodos científicos, ...
- Ferramentas variando de editores hexa a desassembladores de código, ...

Escolha do Método

- Pesquisadores menos experientes podem preferir um método mais organizado de pesquisar vulnerabilidades.
- Pesquisadores mais experientes em programação podem confiar mais no seu sentimento.
- A escolha é uma questão pessoal.

Escolha do Método

- Tipos de código diferentes exigem métodos de pesquisa diferentes : manipular código binário requer um método muito diferente de manipular código-fonte.

Observação

- Existem diversos esquemas de organização diferentes que são usados pelos pesquisadores na comunidade de segurança ao pesquisar vulnerabilidades.

Observação

- Esses esquemas são variados.
- Alguns pesquisadores confiam em auditorias metódicas, com organização pré-estabelecida, realizadas parte-por-parte.
- Ou outros métodos consistentes, mas com uma organização subjetiva.

Ferramentas

- Existem gratuitamente disponíveis diversos pacotes de rastreamento de vulnerabilidade e de rastreamento de auditoria de software.

- *Bugzilla* oferece diversos recursos e uma interface. Ver em http://bugzilla.mozilla.org/

Pesquisa de Código-Fonte

- O código-fonte pode ser escrito em qualquer linguagem: C, Perl, Java, C++, ASP, PHP, ...

- A pesquisa começa geralmente com a procura de funções propensas a erro.

Descoberta por diferença

- Método usado para se descobrir as vulnerabilidades de um pacote.

- Tipo de pesquisa utilizado quando um fornecedor corrige uma vulnerabilidade em um pacote, mas não disponibiliza detalhes sobre o problema.

Descoberta por diferença

- O método determina se um arquivo-fonte foi alterado, e se foi, que partes do arquivo foram alteradas de uma versão para outra.

- Utilitário usado neste tipo de pesquisa: *diff*
- http://www.gnu.org/software/diffutils/diffutils.html

Descoberta por diferença

- *diff* é distribuído com a maioria dos sistemas operacionais UNIX.

- Também disponível para outras plataformas através, por exemplo, da Free Software Foundation.

Descoberta por diferença

- Usada em arquivos-fonte para mostrar as diferenças exatas entre as bases-fontes.
- Ver exemplo, livro Ryan Russel, pag. 84-85.
- Fornecedor provê um *patch* que seja um diff entre duas bases-fonte, como em FreeBSD. Ver exemplo, por Ryan Russel, pag. 85.

Pesquisa por Código-Fonte

- Com a GNU License e movimentos de fontes aberta (open source), a opção de se obter o código-fonte é mais viável.

- Mas, nem todos os fornecedores abraçaram esse movimento e muitos pacotes de software permanecem com fonte-fechada.

Funções Propensas a Erro

- Funções das próprias linguagens.
- Na linguagem C, temos *strcpy, sprintf* ou *mktemp.*
- Essas funções C são normalmente exploradas ou usadas impropriamente para realizar atividades maliciosas.

Funções Propensas a Erro

- O usos das funções *strcpy, sprintf* pode resultar em transbordamento de buffer (*buffer overflow*), devido à falta de verificação de limite (o que é armazenado é maior do que a área reservada) que .

Funções Propensas a Erro

- Exemplos: livro Ryan Russel, pag.88-91:
- *strcpy (c, b);*
- *sprintf (c, “%s”, b);*
- *strcat (c, b);*
- *gets (c);*

Funções Propensas a Erro

- A função *mktemp* pode resultar em *race conditions* (condições de corrida) exploráveis como substituição de arquivos e elevação de privilégios. Ver no exemplo, livro Ryan Russel, pag. 93.

Race Conditions

- São erros de programação comuns definidos como uma situação onde alguém pode adulterar um programa para um determinado evento.

Race Conditions

- Eventos podem ser desde bloquear a memória para evitar que outro processo altere os dados, em um segmento compartilhado, até a criação de um arquivo no sistema de arquivos.

Race Conditions

- Exemplo: Livro Ryan Russel, pag.93,

1. Caso “race condition” ocorre entre a verificação da existência de um nome de arquivo e a criação do arquivo.

2. A limitação da quantidade máxima de nomes que o arquivo pode usar pode resultar em um ataque de link simbólico.

Bugs de Validação de Entrada

- Esse tipo de problema pode resulatar em vulnerabilidades de string de formato.

- Uma vulnerabilidade de string de formato consiste em passar vários especificadores de string, como:
  - %i%i%i%i ou,
  - %n%n%n%n, ...

Bugs de Validação de Entrada

- ... ... para um programa, possivelmente, resultando na execução do código.

- Exemplo: Livro Ryan Russel, pag. 91-92.

Bugs de Validação de Entrada

- A inexistência de validação de entrada, pode permitir que um usuário explore programas como:
  - executáveis de raiz *setuid* (com privilégio de usuário root)
  - *interfaces de* aplicações Web,
  - execução de comandos SQL arbitrários, na comunicação entre um front-end Web e um back-end de banco de dados SQL,

  fazendo-os funcionar mal pela passagem de parâmetros inadequados para eles.

Bugs de Validação de Entrada

- executáveis de raiz *setuid*,
  - conduzem a exploração para executar código com privilégio de usuário root.

Bugs de Validação de Entrada

- *interfaces de* aplicações Web,
  - isso pode permitir que alguém execute comandos arbitrários em um sistema com os privilégios de usuário Web,

  - permitir a um usuário vincular um *shell* a uma porta arbitrária no sistema e ganhar acesso local com permissões do processo HTTP.

Bugs de Validação de Entrada

- execução de comandos SQL arbitrários, na comunicação entre um *front-end Web* e um *back-end de banco de dados SQL:*

  - o usuário pode ver tabelas arbitrárias;
  - executar funções dentro do banco de dados;
  - baixar tabelas.

Técnicas de Engenharia Reversa

- Localizar vulnerabilidades em programa de fonte fechada.
- Pode ser realizada com diversas ferramentas, conforme o sistema operacional.
- Em geral, começa-se com algumas ferramentas de monitoramento sistema operacional para determinar que tipos de arquivos e outros recursos o programa acessa.

Técnicas de Engenharia Reversa

- Uma exceção é se o programa for um programa de rede, caso em que se pode ir direto ao *sniffing*.

- Fonte de ferramentas para Windows é www.sysinternals.com .

Técnicas de Engenharia Reversa

- Ferramentas de interesse:
  - *FileMon*
  - *RegMon*

- Mais sobre essas ferramentas no Cap. 5 sobre **Diferenciação**.

Técnicas de Engenharia Reversa

- *FileMon* e *RegMon* permitirão monitorar um programa em execução, para ver que arquivos estão sendo acessados, se o programa está lendo ou gravando, onde no arquivo ele está e que outros arquivos ele está procurando.

Técnicas de Engenharia Reversa

- Existem três versões de quase todas as ferramentas da SysInternals, e a maioria vem com código-fonte.

- www.Winternals.com vende ferramentas para Windows com mais funcionalidade incluída.

Técnicas de Engenharia Reversa

- A maioria das distribuições UNIX vem com um conjunto de ferramentas equivalentes.

- Em Rosetta Stone, http://bhami.com/rosetta.html , existe uma grande quantidade de ferramentas de rastreamento.

Técnicas de Engenharia Reversa

- Ferramentas UNIX: *trace*, *strace*, *ktrace* e *truss.*

- Ver exemplo, com strace no Linux Red Hat 6.2, pag. 95-100.

- A tarefa do *strace* (e das outras ferramentas) é mostrar chamadas de sistema (kernel) e seus parâmetros.

Técnicas de Engenharia Reversa

- Pode-se aprender muito sobre como um programa rastreado funciona, usando-se ferramentas de rastreamento.

- Os exemplos em no livro Ryan Russel,

  pag. 95-100, demonstram o que você pode aprender pedindo ao sistema operacional para lhe dizer o que ele está fazendo na execução de um programa sendo rastreado.

Técnicas de Engenharia Reversa

- Para um programa complexo estaríamos procurando arquivos em /tmp de nome fixo, leitura de arquivos graváveis por qualquer pessoa, quaisquer chamadas *exec*, e assim por diante.

Pesquisa de Código Binário

- Rastrear a execução de um programa.
- Ferramentas para executar essa tarefa:
  - *truss* com o Solaris (SUN),
  - *strace* com o LINUX
- Rastrear um programa envolve como a execução do programa interage com o sistema operacional.

Pesquisa de Código Binário

- As variáveis de ambiente extraídas pelo programa podem ser reveladas com *flags* da ferramenta de rastreamento.

- Endereços de memória podem ser revelados.

Pesquisa de Código Binário

- Rastrear um programa através de sua execução pode revelar informações sobre vulnerabilidades em certos pontos do programa.

- O uso do rastreamento pode determinar onde e quando uma vulnerabilidade ocorre.

Depuradores

- *Debuggers*
- Método de se pesquisar vulnerabilidades dentro de um programa, enquanto ele é executado.
- Várias implementações de *debuggers*.
- GNU Debugger (GDB).
- Controlam o fluxo de execução enquanto o programa roda.

Depuradores

- O programa inteiro pode ser depurado, ou partes dele.

- Exibem informações como: registradores, endereços de memória e outras que podem levar à descoberta de um problema explorável.

Depuradores

- Param em certos pontos na execução, mudam variáveis, e alteram dinamicamente o código de máquina em alguns casos.

Desassembladores

- Obtém código binário e o transforma em alguma linguagem de mais alto nível, geralmente em uma linguagem Assembly.

- É preciso ser capaz de se ler o código em Assembly.

Desassembladores

- Um “desassemblador” para Windows é o *IDA Pro* (DataRescue).

- “Desassembla código para várias famílias de processadores, inclusive o Java Virtual Machine.

Desassembladores

- Versão de avaliação limitada está em www.datarescue.com/idabase/ida.htm

- SoftICE (Numega), www.compware.com/products/numega/drivercentral

Desassembladores

- Outras ferramentas relacionadas:

  - *nm* e objdump

  da coleção GNU *binutils*.

- *objdump* manipula arquivos binários: exibe cabeçalhos, símbolos em arquivo binário, e “desassembla” em código *assembly*.
- *nm* é similar e permite ver símbolos referenciados por um arquivo binário.

Auditoria baseada em Regras

- Documentos de projeto são usados:
  - diagramas de engenharia de software,
  - folhas de informações,
  - especificações como em RFCs.

- A auditoria é feita através da verificação da conformidade do código com sua especificação no projeto.

Auditoria baseada em Regras

- Examinando a especificação de um protocolo é possível testar serviços comparando-os com sua especificação.

- Além disso, aplicar tipos de ataques conhecidos (*buffer overflow* ou strings de formato) em certas partes da implementação, pode conduzir a verificação da vulnerabilidade.

Sniffers

- Uso de *sniffers* como ferramentas de pesquisa de vulnerabilidades.
- Podem ser usados em redes para monitorar a interatividade entre os usuários na rede e os serviços providos por sistemas.
- Isso possibilita a observação e a representação de tendências que ocorrem no uso de pacotes que provêem serviços.

Sniffers

- Trabalham em conjunto com a auditoria baseada em regras.

- Podem ser usados na pesquisa de vulnerabilidades de interfaces Web (bugs de validação de entrada) ou em protocolos de rede não especificados por padrão público, mas comumente usados.