Vulnerabilidades de Software e Formas de Minimizar suas Explorações

Luiz Otávio Duarte[1]
Luiz Gustavo C. Barbato[1,2]
Antonio Montes[1,2]

[1] LAC - Laboratório Associado de Computação e Matemática Aplicada
INPE - Instituto Nacional de Pesquisas Espaciais

[2] CenPRA - Centro de Pesquisas Renato Archer
MCT - Ministério da Ciência e Tecnologia

{duarte, lgbarbato}@lac.inpe.br, antonio.montes@cenpra.gov.br

Introdução

- O aumento no desenvolvimento de novas tecnologias vem sendo acompanhado por um grande número de sistemas sendo atacados;
- Para que a maioria destes ataques possam ser bem sucedidos é necessária a existência de alguma vulnerabilidade;
- Segundo Viega e McGraw, as vulnerabilidades presentes em programas são as mais exploradas. `http://www.buildingsecuresoftware.com`

Motivação

- Dados do ICAT/NIST mostram que no primeiro semestre de 2005, **75% das vulnerabilidades eram remotamente exploráveis**.
- Das vulnerabilidades remotamente exploráveis, **80% estavam ligadas a má codificação do programa**.
- Das vulnerabilidades remotamente exploráveis, **40% estavam ligadas a problemas de extravazamento de *buffers***.

`http://icat.nist.gov`

Objetivo

Mostrar soluções que podem ser utilizadas para **minimizar as explorações**:

- Inibindo certos tipos de ataques;
- Dificultando a exploração das vulnerabilidades;
- Diminuindo as consequências de invasões;
- Restringindo as ações dos invasores;

Vulnerabilidades de Software

Vulnerabilidade em software é um conjunto de condições que podem levar à violação de uma política de segurança explícita ou implícita.

Vulnerabilidades de Software

- Format String
  - Especificações de **como os dados devem ser representados**;
  - Muito utilizadas em funções da família **`printf()` e `syslog()`**;
  - Um atacante que consiga controlar este formatador pode modificar a maneira que o programa é executado.

`http://www.cs.ucsb.edu/~jzhou/security/formats-teso.html`

Vulnerabilidades de Software

- Injections
  - Ocorre quando **comandos são dinamicamente gerados**;
    - *SQL injection, XPath injection, LDAP injection, OS commanding injection, Code injection, XSS, etc ...*
  - Acontecem, na maioria das vezes, devido a não validação de dados de entrada passados ao programa;

`http://www.webappsec.org/projects/threat/classes_of_attack.shtml`

Vulnerabilidades de Software

- Symbolic Link
  - É um *link* que aponta para outro arquivo e **atua como o arquivo apontado**;
  - A vulnerabilidade existe se um usuário puder criar um *link* para um arquivo que não possui permissão para o acesso;
  - A exploração, geralmente, ocorre em programas que não checam se o arquivo que devem abrir é ou não um *link*.

Vulnerabilidades de Software

- Race Condition
  - Ocorrem em **ambientes que suportam multiprogramação**;
  - A vulnerabilidade ocorre se um recurso que será utilizado por outro processo for intencionalmente modificado;
  - É comum que este tipo de vulnerabilidade seja explorado em em arquivos temporários.

Vulnerabilidades de Software

- Buffer Overflow
  - Geralmente ocorre quando **mais dados do que o suportado tentam ser armazenados**;
  - Atualmente é a vulnerabilidade mais explorada em um software;
  - De um modo geral, podem ser exploradas via *smash stack*, *arc injection*, *pointer subterfuge*.

`http://csdl.computer.org/comp/proceedings/discex/2000/0490/02/0490toc.htm`

Métodos de Exploração

- Stack Smash
  - Objetiva **modificar o endereço de retorno**;
  - Faz com que este aponte para um **código executável**;
  - Este código deve ser inserido pelo atacante.

`http://www.phrack.org/phrack/49/P49-14`

Métodos de Exploração

- Arc Injection
  - Também tem por objetivo **modificar o endereço de retorno**;
  - Geralmente, faz com que aponte para um local que contenha alguma função específica, como `system()`;
  - Não existe a necessidade da injeção de código por parte do atacante.

Métodos de Exploração

- Pointer Subterfuge
  - Faz com que um ponteiro **aponte para um lugar arbitrário no PAS**;
  - Não necessita sobrescrever variáveis adjacentes até o endereço de retorno;
  - Não necessita da injeção de código executável.

Formas de Minimizar as Explorações

Segundo Crispin Cowan, o programa sem falhas é infactível para sistemas não triviais.

- Abordagens dependentes do Compilador
- Abordagens dependentes do Sistema
- Abordagens dependentes da Aplicação

Formas de Minimizar as Explorações

- Abordagens dependentes do compilador
  - Dados inseridos em **tempo de compilação** para que o programa seja seguro em **tempo de execução**;
  - Geralmente acopladas ao compilador do sistema;
  - Não eliminam as vulnerabilidades, mas atuam sobre sua exploração.

Formas de Minimizar as Explorações

- Abordagens dependentes do Compilador
  - **StackGuard**
- Objetivo é o programa finalizar ao ocorrer um ataque a um *buffer overflow*;
- Utiliza-se de canários $\rightarrow$ randômicos, NULL terminator, XOR randômico;
- Eficiente contra ataques de *smash stack*.

`http://www.cse.ogi.edu/DISC/projects/immunix/StackGuard`

Formas de Minimizar as Explorações

- Abordagens dependentes do Compilador
  - **SSP - Antigo ProPolice**
- Como o StackGuard, possui o mesmo objetivo e um canário randômico, chamado de guardião;
- Reordenamento das variáveis e argumentos das funções;
- Aborda ataques de *stack smash*, *arc injection*, *pointer subterfuge*.

`http://www.trl.ibm.com/projects/security/ssp`

Formas de Minimizar as Explorações

- Abordagens dependentes do Compilador
  - **StackShield**
- Objetivo é o programa finalizar ao ocorrer um ataque a um *buffer overflow*;
- Armazena os endereços de retorno em um local dito "seguro";
- Eficiente contra ataques de *stack smash*.

`http://www.angelfire.com/sk/stackshield`

Formas de Minimizar as Explorações

- Abordagens dependentes do Compilador
  - **PointGuard**
- Objetivo é o programa finalizar ao ocorrer um ataque do tipo *pointer subterfuge*;
- Trabalha fazendo um **XOR criptográfico** entre o valor do ponteiro e um valor randômico;
- Não possui uma versão disponível publicamente.

`http://www.ece.cmu.edu/~adrian/630-f03/readings/cowan-pointguard.pdf`

Formas de Minimizar as Explorações

- Abordagens dependentes do Compilador
  - O SSP ou StackGuard + PointGuard mostram-se eficazes $\rightarrow$ *stack smash*, *pointer subterfuge* e *arc injection*;
  - Entretanto, somente a PointGuard trata problemas de *heap smash*. Se a `libc` fosse compilada com o PointGuard.

Formas de Minimizar as Explorações

- Abordagens dependentes do Sistema

As abordagens dependentes do sistema tem como objetivo proteger o sistema inteiramente **em tempo de execução**. Para isso, faz com que as bibliotecas e o `kernel` do sistema sejam fortificados.

Dentre as ferramentas atualmente disponíveis, as que realmente tratam vulnerabilidades de *buffer overflow* são as chamadas de **bloqueadoras de comportamento**.

Formas de Minimizar as Explorações

- Abordagens dependentes do Sistema
  - **Libsafe**

- Previne *buffer overflows*.
- É uma espécie de invólucro para a biblioteca padrão do `C`.
- Intercepta e checa os limites dos argumentos antes de passar para as funções da bilbioteca padrão do `C`.
- É transparente para os processos protegidos pois é carregada automaticamente (`/etc/ld.so.preload` ou LD_PRELOAD).

`http://www.research.avayalabs.com/project/libsafe/index.html`

Formas de Minimizar as Explorações

- Abordagens dependentes do Sistema
  - W^X
- W xor X (OR não pode ser pois permitiria os 2 juntos)
- OpenBSD
- Páginas executáveis não podem ter permissão de escrita e vice-versa.
- Leitura e Escrita = Dados, Leitura e Execução = Instruções

`http://www.openbsd.org/security.html`
`http://www.bytelabs.org/download/material/os-security04.pdf`

Formas de Minimizar as Explorações

- Abordagens dependentes do Sistema
  - **Openwall**

- É um patch para o `kernel` de sistemas `Linux`.
- Implementa *stack* não executável.
- Usuários não podem criar links para arquivos que eles não possuem permissões de leitura e escrita.

`http://www.openwall.com/linux/README.shtml`

Formas de Minimizar as Explorações

- Abordagens dependentes do Sistema
  - **PaX**

- Proteções por menor privilégio para páginas de memória.
- SEGMEXEC, PAGEEXEC (NX/XD bit)
- Restrições `mprotect()` (impede PROT_WRITE e PROT_EXEC)
- Base `mmap()` randômica (códigos dinamicamente alocados são mapeados em diferentes posições a cada execução)

`http://pax.grsecurity.net`

Formas de Minimizar as Explorações

- Abordagens dependentes do Sistema
  - **Exec Shield**
- Desenvolvido pela equipe Red Hat
- Red Hat Enterprise Linux

`http://www.redhat.com/f/pdf/rhel/WHP0006US_Execshield.pdf`

Formas de Minimizar as Explorações

- Abordagens dependentes do Sistema
  - **Windows DEP**
- Data Execution Prevention (DEP)
- Windows XP Service Pack 2

`http://www.microsoft.com/technet/prodtechnol/winxppro/maintain/sp2mempr.mspx`

Formas de Minimizar as Explorações

- Abordagens dependentes do Sistema
  - **RaceGuard**

- É um patch para o Linux para não permitir que *race conditions* ocorram na criação de arquivos temporários.
- Se `stat()` falhar (arquivo não existe), o nome do arquivo é armazenado em um cache.
- Agora se uma subsequente `open()` com o mesmo nome de arquivo detectar que o arquivo existe, o **ataque** é detectado e a aplicação é abortada.

`http://www.cse.ogi.edu/sysl/readings/papers/RaceGuard.pdf`

Formas de Minimizar as Explorações

- Abordagens dependentes do Sistema
  - **Systrace**
- Permite controlar as chamadas de sistemas que as aplicações podem acessar no sistema
- É uma ferramenta que permite que o administrador especifique quais recursos um dado processo pode acessar.
- Apesar de não eliminar ou prevenir um ataque contra um *buffer overflow*, esta abordagem permite que menos privilégios sejam dados ao atacante.

`http://www.citi.umich.edu/u/provos/systrace`

Formas de Minimizar as Explorações

- Abordagens dependentes do Sistema
  - **Grsecurity MAC**
- Patch para o Linux
- MAC - Mandatory Access Control
- Permite restringir o acesso a arquivos, recursos, sockets, etc, de TODOS os usuários, baseado em um sistema de ACL (Access Control List)
- *"... aumenta a dificuldade do comprometimento com sucesso do sistema."*

`http://www.grsecurity.net/gracldoc.pdf`

Formas de Minimizar as Explorações

- Abordagens dependentes da Aplicação
- Realizam o que é chamado de **auditoria de segurança sofware**.
- Esta auditoria pode ser realizada de maneira **estática** ou **dinâmica**.
- A **análise estática** visa procurar erros de codificação da aplicação inspecionando seu código fonte.
- A **análise dinâmica** busca encontrar problemas de segurança durante a execução do programa.

Formas de Minimizar as Explorações

- Abordagens dependentes da Aplicação
  - Analisadores Estáticos

- **Flawfinder**: `http://www.dwheeler.com/flawfinder`
- **ITS4**: `http://www.cigital.com/its4`
- **PScan**: `http://www.striker.ottawa.on.ca/~aland/pscan`
- **RATS**: `http://www.securesw.com/rats`
- **Splint**: `http://www.splint.org`

Formas de Minimizar as Explorações

- Abordagens dependentes da Aplicação
  - Analisadores Dinâmicos

WebScarab Project

- É um framework, escrito em Java, para analisar aplicações que usam protocolos de comunicação HTTP e HTTPS.

`http://www.owasp.org/software/webscarab.html`

Formas de Minimizar as Explorações

- Abordagens dependentes da Aplicação
  - Analisadores Dinâmicos

O projeto .NET OWASP é a coleção de projetos focados no tema Asp.Net de segurança. Projetos atuais podem ser usados para ajudar a identificar e testar aplicações Web.

`http://www.owasp.net`

Formas de Minimizar as Explorações

- Abordagens dependentes da Aplicação
  - Analisadores Dinâmicos
- **SAMSHE** (Security Analyser for Microsoft Shared Hosting Environment) - Projetada para adiministradores e usuários finais. Esta ferramenta identifica vulnerabilidades comuns no IIS (Internet Information Server) 5.0 ou 6.0 executando Asp.Net ou código ASP.

Formas de Minimizar as Explorações

- Abordagens dependentes da Aplicação
  - Analisadores Dinâmicos
- **ANSA** (Asp.Net Security Analyser) e **ANBS** (Asp.Net Baseline Security) - Estes projetos contém ferramentas que ajudam a identificar vulnerabilidades comuns disponíveis no Asp.Net e para configurar seguramente o servidor eles a executam.

Formas de Minimizar as Explorações

- Abordagens dependentes da Aplicação
  - Analisadores Dinâmicos
- **C# Spider** - Parte inicial do projeto que construirá um scanner de segurança para Web para temas de segurança conhecidos usando o formato OASIS WAS e scanner de segurança para SQL Injection e Cross Site Scripting.

Considerações Finais

- Instalar e configurar esses recursos demanda um certo tempo.
- As soluções apresentadas não deixam os sistemas seguros.
- Existem formas de driblar essas soluções.

  Exemplo: `http://www.phrack.org/phrack/56/p56-0x05`
- Minimizar as explorações é uma maneira de diminuir os riscos.
- **Desenvolvimento de Software Seguro sempre foi, ainda é e sempre será a melhor abordagem.**

Mensagem Final

Buffers **NÃO** podem ser **ESTOURADOS**. Podem ser **EXTRAVAZADOS**.

(GTS 01.2005)