ANÁLISE DE *EXPLOITS* PARA APLICAÇÕES LINUX

Jhonatan Ríchard Raphael[1]; Fabrício Sérgio de Paula[2]

[1]Estudante do Curso de Ciência da Computação da UEMS, Unidade Universitária de Dourados; e-mail: 017362@comp.uems.br. Bolsista UEMS.

[2]Professor do Curso de Ciência da Computação da UEMS, Unidade Universitária de Dourados; e-mail: fabricio.paula@gmail.com

Redes de Computadores e Segurança Computacional.

Resumo

Este projeto propôs analisar *exploits* para aplicações que executem sob o sistema operacional Linux. Foram analisadas as ações realizadas por *exploits* durante a fase inicial de execução, de forma a identificar as instruções e chamadas ao sistema operacional mais freqüentes durante ataques. Com essas informações, é possível realizar o projeto de um IDS adequado para detectar ataques a aplicações nesse sistema.

Palavras-chave: Sistema de Detecção de Intrusão. Chamadas ao sistema. Vulnerabilidades.

Introdução

Embora a interconexão de redes de computadores resulte em benefícios salientes, a diversidade proporcionada nesses ambientes contempla também indivíduos dispostos a promover perturbações. Esses indivíduos são movidos por razões ideológicas, psicológicas, financeiras, desejo de vingança e até mesmo por pura diversão (DE PAULA, 2004).

Por outro lado, assuntos relacionados à segurança computacional nem sempre recebem atenção adequada por parte dos desenvolvedores de *software*, revendedores, administradores e consumidores. Como parte dessa distração, um número alto de vulnerabilidades surge constantemente, colocando usuários individuais e instituições inteiras sob risco (PETHIA, 2000; SPAFFORD & GARFINKEL, 1996).

Diversas tecnologias têm sido empregadas visando garantir os quesitos de segurança computacional, incluindo *firewalls*, mecanismos criptográficos e sistemas de detecção de intrusão (NAKAMURA & DE GEUS, 2003). Um sistema de detecção de intrusão (IDS) é

especialmente interessante porque permite identificar a ocorrência de ataques em um computador ou rede de computadores. Devido às suas características, o IDS tornou-se uma tecnologia essencial para administrar a segurança de corporações.

Um IDS atuando no nível de pacotes é bastante útil para identificar ataques. Entretanto, sua capacidade pode ser limitada, especialmente ao lidar com tráfego cifrado. Apesar de, nesse nível, haver essa limitação, um ataque com sucesso irá produzir comportamento inadequado em processos do sistema operacional atacado. Por essa razão, a identificação de ataques no nível de aplicação pode ser mais precisa, embora em alguns casos possa ser mais tardia.

Este trabalho propôs analisar *exploits*, que são ataques projetados para explorar fraquezas em aplicações de modo a comprometer o sistema atacado (NORTHCUTT et al., 2001). A análise de *exploits* para aplicações Linux[1] possibilita entender qual o comportamento inicial de processos atacados nesse tipo de sistema. O conhecimento desse comportamento é necessário para um posterior projeto de IDS adequado para lidar com ataques no nível de aplicação da pilha de protocolos TCP/IP[2].

Material e Métodos

Foram realizados estudos e experiências práticas envolvendo a execução de aplicações em máquinas virtuais ou ambientes confinados através de *chroot* (SPAFFORD & GARFINKEL, 1996) e o monitoramento da execução de aplicações. As fontes de referências bibliográficas incluem artigos disponíveis na Internet, e o acesso às bibliotecas da UEMS e UFGD. As atividades práticas foram realizadas através do uso de máquinas virtuais e ferramentas de monitoramento como *strace* e *ltrace* e a biblioteca *ptrace*.

Este projeto contou com o acesso aos recursos computacionais da UEMS, que inclui um laboratório com 02 computadores dedicados. Foi utilizado o sistema operacional Linux.

Resultados e Discussão

A biblioteca ou chamada ao sistema *ptrace()* oferece uma forma de monitorar determinado processo em execução e alterá-lo. Pode ser útil para criar sistemas de depuração e rastreamento de chamadas ao sistema. Segue um exemplo de código abaixo de como capturar as instruções e chamadas ao sistema write de um processo.

---

1 Detalhes do sistema operacional Linux podem ser encontrados em http://www.kernel.org.
2 Detalhes sobre a pilha de protocolos TCP/IP podem ser encontrados em (Stevens, 2000).

```
child = fork();
if(child == 0) {
        ptrace(PTRACE_TRACEME, 0, NULL, NULL);
        execl("/home/jhonatan/Desktop/desenvolvendo/exploit", "exploit", NULL);
}
else {
        int status;

        union u {
                long val;
                char chars[long_size];
        }data;

        struct user_regs_struct regs;
        int start = 0;
        long ins;

        while(1) {
                wait(&status);
                if(WIFEXITED(status))
                        break;
                ptrace(PTRACE_GETREGS,child, NULL, &regs);
                if(start == 1) {
                        ins = ptrace(PTRACE_PEEKTEXT,child, regs.eip,NULL);
                        printf("EIP: %ld Instruction ""executed: %lx\n",regs.eip, ins);
                }
                if(regs.orig_eax == SYS_write) {
                        start = 1;
                        ptrace(PTRACE_SINGLESTEP, child,NULL, NULL);
                }
                else
                        ptrace(PTRACE_SYSCALL, child,NULL, NULL);
        }
```

Figura 1 – Captura de instruções e chamadas ao sistema write de um processo.

Foram coletados *exploits* para aplicações Linux de qualquer natureza. O repositório utilizado encontra-se no seguinte endereço *web*: http://www.exploit-db.com/. O banco de dados divide-se nas seguintes categorias: remoto, local, *web*, *dos* e *shellcode*. Alguns exemplos a seguir:

- *ProFTPD 1.3.3c compromised source remote root Trojan*. Tipo de ataque: remoto.
- *Printoxx Local Buffer Overflow*. Tipo de ataque: local.
- *ProFTPd 1.3.0 mod_ctrls Local Stack Overflow*. Tipo de ataque: local.

Apesar da chamada ao sistema *ptrace* auxiliar no rastreamento dos *exploits*, é necessário algum método para executá-los corretamente, simulando algum tipo de ataque. Um exemplo de código C que executa um *shellcode* abaixo:

```
1  char shellcode[] =
2    "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
3    "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
4    "\x80\xe8\xdc\xff\xff\xff/bin/sh";
5
6  main()
7  {
8          void (*fp) (void);
9          fp = (void *)shellcode;
10         fp();
11 }
```

Figura 2 – Execução de *shellcode* com ponteiro para função

O rastreamento de chamadas ao sistema dos *exploits* foi executado por um programa desenvolvido utilizando a biblioteca *ptrace*. Os *shellcodes* foram executados por um programa auxiliar que utiliza o método de corrupção de memória. Segue o código abaixo:

```
1  char shellcode[] =
2                  "\x6a\x0b\x58\x99\x52"
3                  "\x68\x61\x61\x61\x61"
4                  "\x89\xe1\x52\x6a\x74"
5                  "\x68\x2f\x77\x67\x65"
6                  "\x68\x2f\x62\x69\x6e"
7                  "\x68\x2f\x75\x73\x72"
8                  "\x89\xe3\x52\x51\x53"
9                  "\x89\xe1\xcd\x80\x40"
10                 "\xcd\x80";
11
12 void main() {
13         int *ret;
14         ret = (int *)&ret + 2;
15         (*ret) = (int)shellcode;
16 }
```

Figura 3 – Execução de *shellcode* com ponteiro para função

Os *exploits* foram executados em um processador Intel Pentium T4400 com 3GB de memória RAM, rodando o sistema operacional Linux Ubuntu 11.04 i386, *kernel* 2.6.38-8. Foram utilizados o monitor de máquinas virtuais Virtual Box e o compilador GCC 4.5.2.

As chamadas ao sistema implementadas foram divididas em três grupos:

- **Arquivos, diretórios e links**: abertura, criação, remoção, truncamento e alteração de atributos (*pathname*, proprietário, grupo e permissões);
- **Processos**: criação e execução;
- **Comunicação**: envio de sinais para processos, estabelecimento e aceitação de conexões TCP, e envio de recebimento de tráfego UDP.

Foram rastreados 17 *shellcodes* de *exploits* locais:

Tabela 1 – Chamadas ao sistema executadas pelos *exploits* locais divididas em grupos

Chamadas ao sistema	**Quantidade**	**Porcentagem**
Arquivos, diretórios e links	67	50,00%
Processos	61	45,52%
Comunicação	6	4,48%
Total	134	100,00%

As chamadas ao sistema mais freqüentes dos *exploits* locais foram clone, close, write e execve com 18, 14, 13 e 13 vezes respectivamente.

Foram rastreados 23 *shellcodes* de *exploits* remotos:

Tabela 2 – Chamadas ao sistema executadas pelos *exploits* remotos divididas em grupos

Chamadas ao sistema	**Quantidade**	**Porcentagem**
Arquivos, diretórios e links	58	40,85%
Processos	51	35,92%
Comunicação	33	23,24%
Total	142	100,00%

As chamadas ao sistema mais freqüentes dos *exploits* remotos foram execve e dup2, com 14 e 13 vezes respectivamente.

Segue os dados de rastreio dos 40 *exploits* rastreados:

Tabela 3 – Chamadas ao sistema executadas por todos os *exploits* divididas em grupos

Chamadas ao sistema	**Quantidade**	**Porcentagem**
Arquivos, diretórios e links	125	45,29%
Processos	112	40,58%
Comunicação	39	14,13%
Total	276	100,00%

As chamadas ao sistema mais freqüentes envolvendo todos os *exploits* rastreados foram execve e close, com 27 e 25 vezes respectivamente.

Conclusões

Conclui-se que o desenvolvimento deste projeto atingiu o objetivo proposto, identificando as principais e mais freqüentes chamadas ao sistema de *exploits* para aplicações Linux.

O foco principal do uso de máquinas virtuais foi usufruir as vantagens da virtualização através das mesmas. Dessa maneira, é possível gerar um ambiente seguro para testes e simulação da execução de *exploits*, sem causar danos ao sistema anfitrião ou outros sistemas na rede local. Embora a máquina necessite do sistema real para sua inicialização, há uma independência da mesma com as máquinas virtuais nela criadas, tornando o sistema seguro. A infecção por vírus em uma máquina virtual não afeta a máquina real.

Depois de realizado um estudo referente à biblioteca *ptrace*, foi desenvolvido um programa para rastreamento de processos, de acordo com as necessidades deste projeto. Este programa foi essencial para o levantamento de informações a respeito dos *exploits* analisados.

Em relação aos métodos de execução dos *exploits*, foram desenvolvidos pequenos exemplos de corrupção de memória e *buffer overflow* que possam simular ataque, além do programa auxiliar na execução dos *exploits*.

Em relação aos resultados do monitoramento, pode-se concluir que códigos maliciosos de *exploits* executam chamadas ao sistema referentes a processo na mesma proporção que as referentes a arquivos, diretórios e links. Nota-se também que houve um grande uso da chamada ao sistema execve, que tem a função de executar um arquivo; e da chamada ao sistema dup2, que duplica um arquivo já existente.

Para informações mais detalhadas, o relatório científico está disponibilizado no link: http://websrv.comp.uems.br/~fabricio/artigos/relat-ic-jhonatan-2010-2011.pdf

Agradecimentos

Agradeço ao Fabrício, pelos conselhos e orientação. Agradeço à minha família, pela paciência e incentivo. Agradeço à UEMS pelo apoio financeiro e às pessoas que contribuíram para a realização desse trabalho de alguma maneira.

Referências

DE PAULA, F. 2004. Uma arquitetura de segurança computacional inspirada no sistema imunológico. Tese (Doutorado em Ciência da Computação) – Universidade Estadual de Campinas, 153p.

NAKAMURA, E., DE GEUS, P. 2003. **Segurança de redes em ambientes cooperativos**. São Paulo-SP, Ed. Futura, 488p.

NORTHCUTT, S., COOPER, M., FEARNOW, M., FREDERICK, K. 2001. **Intrusion signatures and analysis**. Thousand Oaks-CA, Ed. New Riders Publishing, 448p.

PETHIA, R. 2000. CERT Coordination Center. **Internet security issues**. Disponível em: http://www.cert.org/congressional_testimony/Pethia_testimony25May00.html (último acesso em 23/06/2009).

SPAFFORD, G., GARFINKEL, S. 1996. **Practical Unix & Internet security**. Sebastopol, Ed. O'Reilly Media, 1004p.