

# **Universidade Federal do Paraná**

Loirto Alves dos Santos  
Luiz Henrique Pires de Camargo

## **Vírus de computador** **Uma abordagem do código polimórfico**

**Curitiba - PR**

2012

# **Universidade Federal do Paraná**

Loirto Alves dos Santos

Luiz Henrique Pires de Camargo

## **Vírus de computador**

**Uma abordagem do código polimórfico**

Monografia apresentada junto ao curso de Ciência da Computação, do Departamento de Informática, do Setor de Ciências Exatas, como requisito parcial para a obtenção do título de Bacharel.

**Orientador:** Prof. Dr. Bruno Müller Junior

**Curitiba - PR**

2012

*A nossos pais. Sem eles nada disso teria sentido.*

# Agradecimentos

- A Deus
- A nosso esforço e dedicação que, apesar de serem poucos, nos valeram muito.
- Aos professores pela paciência e dedicação

# Resumo

Este trabalho tem por finalidade realizar um estudo sobre alguns algoritmos e técnicas de polimorfismo utilizadas para criar vírus de computador e o quanto elas tornam difícil - e algumas vezes até mesmo impossível - a detecção do código malicioso.

**Palavras-chave:** Vírus, Vírus de computador, Vírus polimórfico, polimorfismo.

# Abstract

This paper aims to conduct a study of some algorithms and techniques used to create polymorphic computer viruses and how they make it difficult - and sometimes even impossible - to detect the malicious code.

**Keywords:** Virus, Computer Virus, Polymorphic virus, Polymorphism.

# Sumário

<b>Agradecimentos</b>	<b>iv</b>
<b>Resumo</b>	<b>v</b>
<b>Abstract</b>	<b>vi</b>
<b>Sumário</b>	<b>vii</b>
<b>1 Introdução</b>	<b>1</b>
<b>2 Revisão bibliográfica</b>	<b>2</b>
2.1 Antivírus . . . . .	2
2.2 História . . . . .	2
2.3 Antivirus e SO . . . . .	2
2.3.1 DOS . . . . .	3
2.3.2 Windows . . . . .	3
2.3.3 Linux . . . . .	3
2.4 Técnicas de detecção . . . . .	4
2.4.1 Virus de pendrive . . . . .	4
2.4.2 Virus de macro . . . . .	4
2.4.3 Virus Polimórficos . . . . .	5

<b>3</b>	<b>Descrição conceitual do trabalho</b>	<b>6</b>
<b>4</b>	<b>Detalhes do trabalho</b>	<b>7</b>
<b>5</b>	<b>Conclusão</b>	<b>8</b>
	<b>Referências Bibliográficas</b>	<b>9</b>
<b>A</b>	<b>Estrutura de Arquivos PE e ELF</b>	<b>11</b>
A.1	Arquivo PE . . . . .	11
A.1.1	Estrutura de arquivo PE. . . . .	11
A.1.2	PE - Cabeçalho . . . . .	12
A.1.3	Tabela de Seções . . . . .	12
A.1.4	Páginas de imagem . . . . .	13
A.1.5	Importação . . . . .	13
A.1.6	Exportação . . . . .	14
A.1.7	Correção . . . . .	14
A.1.8	Recursos . . . . .	14
A.1.9	Debug . . . . .	14
A.2	Arquivo ELF . . . . .	14
A.2.1	A estrutura do arquivo ELF . . . . .	15



## Introdução

Nossa vida moderna é extremamente dependente de computadores: desktops, notebooks, netbooks, PDA, celulares, satélites, veículos, microondas, televisores, gps, bancos, energia elétrica, comunicações ..., enfim, uma gama enorme de exemplos poderiam ser citados. Dentro deste contexto, os vírus de computador (e suas variações) são uma ameaça real à qual todos - direta ou indiretamente - estamos expostos.

## Revisão bibliográfica

### 2.1 Antivírus

Os antivirus são softwares criados para analisar, detectar, eliminar e impedir os virus informáticos ou ao menos diminuir a intensidade do ataque. Foram criados pela necessidade de que os virus impedissem a utilização do sistema. Os virus atuais são mais poderosos, e ainda existem outros não tão fortes que são utilizados como piada ou somente para incomodar, se espalhar pelos computadores sem fazer mal à máquina e sim à paciência do usuário.

### 2.2 História

O primeiro antivirus foi criado em 1988 por Denny Yanuar Ramdhani. Era uma vacina ao virus Brain, um virus de boot, além de remover o virus imunizava o sistema contra uma nova infecção. A forma de desinfectar era remover as entradas do virus no pc e já bloqueava estas fraquezas para impedir um novo ataque. Ainda em 1988 um virus foi projetado para infectar com a "ajuda" da BBS, neste John McAfee, desenvolveu o VirusScan, primeira vacina para o virus.

### 2.3 Antivirus e SO

Por enquanto existe uma dependência dos virus para com os sistemas operacionais, pois afetam o modo em que o executável interage com o sistema, e pedidos

especiais são feitos pelo próprio SO e cada qual o faz de forma diferente, ou seja um vírus que funciona em windows nunca funcionaria em linux, só se fossem chamadas suas APIs, como feito pelo wine no sistema linux, e mesmo assim não teria todo o potencial de infecção, já que é preparado para a estrutura do sistema para o qual foi projetado.

### **2.3.1 DOS**

No sistema DOS o anti-virus não funciona em "tempo real", somente como scanner, normalmente era colocado no boot do sistema para varrer o sistema em busca de novas infecções, e outras verificações somente se chamado pelo usuário. Sendo infectado no meio de uma tarefa o vírus já se propagou e danificou diversas áreas e somente será percebido na nova execução do antivírus.

### **2.3.2 Windows**

Já no windows o antivírus protege as principais formas de ataque, para este sistema. continua a utilizar o scanner, como no DOS. Ganhou a função de monitoramento, com diversas ferramentas para encontrar padrões de vírus. A cada executável aberto há esta verificação, o que compromete o desempenho do computador. A cada período pré-determinado há uma varredura sobre os arquivos do sistema para verificar arquivos infectados, remove o vírus e tenta manter a integridade do arquivo. Se encontra um padrão de infecção mas ainda não existe "vacina" para remoção diversos sistemas de proteção utilizam a ferramenta de "quarentena", ou seja mantém o arquivo infectado em um espaço que não pode ser "alcançado" pelo usuário até que possa restaurar o arquivo, ou ao menos conheça o vírus.

### **2.3.3 Linux**

Não são muito populares neste sistema. Por enquanto não há uma grande preocupação, nem pela parte de usuários e nem pela parte de desenvolvedores. O que existe hoje são alguns sistemas que detectam vírus para windows pelo linux, para fazer uma manutenção do sistema. E mesmo assim não são tão "potentes" quanto os de windows, não há muita preocupação em desenvolvê-los.

## 2.4 Técnicas de detecção

São diversas as técnicas de detecção dentre elas: Heurística: Que significa descobrir. Estuda o comportamento, estrutura e características para analisar se é perigoso ao sistema ou inofensivo. Emulação: Abre o arquivo em uma virtualização do sistema, e analisa os efeitos sobre o sistema. Arquivo monitorado: Mantém um arquivo no sistema e o monitora, se ele modificar alguma característica é porque o sistema foi infectado. E então o antivírus toma as precauções necessárias. Assinatura do vírus: Com um trecho de código do vírus tem-se sua assinatura, quando tenta detectar o vírus busca-as para analisar se já não existe dentro do banco de dados do antivírus. Temos o falso positivo, o antivírus com base no comportamento do arquivo o considera infectado, o que dificulta para usuários comuns identificarem as anomalias e utilizar com segurança o sistema.

### 2.4.1 Virus de pendrive

No sistema operacional windows eles se utilizam do arquivo autorun.inf para se autoexecutar e infectar a máquina. sua limpeza é simples, existem alguns antivírus que alteram o conteúdo do autorun e tiram a permissão de gravação do arquivo, e alguns usuários criam um diretório com o nome autorun.inf e isso impede de criar o tal arquivo. os vírus em si funcionam de forma interessante, temos por exemplo o conficker q após infectar o pc ele passa a infectar td pendrive q nel for utilizado, assim como enquanto conectado a internet ele baixa diversos outros vírus e com isso acaba com o sistema e arquivos do usuário. sua prevenção é simples e sua remoção é complicada. ou seja se todos fossem informados de como o vírus funciona a prevenção seria óbvia e este tipo de vírus seria obsoleto.

### 2.4.2 Virus de macro

Os vírus de macro são utilizados dentro de, aparentemente, inofensíveis arquivos estilo "office" são scripts executados automaticamente para facilitar a visualização dos arquivos e fazer eles executarem o que teriam de executar, os criadores de vírus aproveitam que macros tem poder de execução e infectam os arquivos colocando dentro a macro código malicioso que o usuário previamente nem notará, e após execução do arquivo já estará infectado e infectará outros. A maior praga disso está nas apre-

sentações de slides, como foi muito difundido por e-mails para passar imagens com animações. O vírus se instala dentro destes arquivos e o usuário desconhece que por trás de tudo que está visualizando um vírus acabou de se instalar em sua máquina.

### **2.4.3 Vírus Polimórficos**

Ainda não existe uma forma eficaz para se detectar este tipo de vírus, eles não tem um padrão a ser identificado. O que se faz é criar um arquivo de vítima e este fica sempre sendo monitorado, mas o bom vírus polimórfico já está residente em memória e faz o sistema "ver" o arquivo como inalterado e com isso não há mais nada a ser feito. seria uma limpeza manual, sem o auxílio de outra máquina seria inviável, enquanto o vírus se infecta o usuário tentaria localizá-lo e deletá-lo uma guerra perdida.

# Capítulo 3

## Descrição conceitual do trabalho

# Capítulo 4

## Detalhes do trabalho

# Capítulo 5

## Conclusão



# Referências Bibliográficas

- [1] P. Szor. *The Art of Computer Virus Research and Defense* (Addison Wesley Professional, 2005).
- [2] Wikipedia. Computer virus — Wikipedia, The Free Encyclopedia (2012). [Online; acessado em 11-Abr-2012], URL [http://en.wikipedia.org/w/index.php?title=Computer\\_virus&oldid=486444876](http://en.wikipedia.org/w/index.php?title=Computer_virus&oldid=486444876).

## **Apêndices**

# Estrutura de Arquivos PE e ELF

## A.1 Arquivo PE

O formato de arquivo PE (Portable Executable Format File) é o último utilizado para plataforma Microsoft.

### A.1.1 Estrutura de arquivo PE.

DOS 2 - Cabeçalho EXE compatível	Seção DOS 2.0 (para compatibilidade com DOS somente)
Não utilizado	
OEM - Identificador	
OEM - Info	
Offset para cabeçalho PE	
DOS 2.0 Stub Program & Reloc. Table	
Não utilizado	
PE - Cabeçalho	Palavras limitadas a 8 bytes
Tabela de seções	
Image Pages <ul style="list-style-type: none"> <li>· Info de Importação</li> <li>· Info de Exportação</li> <li>· Info de correção</li> <li>· Info de recursos</li> <li>· Info de debug</li> </ul>	

### A.1.2 PE - Cabeçalho

Temos no cabeçalho uma estrutura dividida em campos com palavras de 4 bytes, enfatizamos alguns deles abaixo:

Tipo de CPU: o campo informa qual o tipo de CPU para a qual o executável foi projetado.

Número de Seções: o campo informa o número de entradas na tabela de seções.

Marca de Tempo/Data: Armazena a data de criação ou modificação do arquivo.

Flags: Bits para informar qual o tipo de arquivo ou quando há erros em sua estrutura.

LMAJOR/LMINOR: maior e menor versão do linkador para o executável.

Seção de alinhamento: O valor de alinhamento das seções. Deve ser múltiplo de 2 dentre 512 e 256M. O valor padrão é 64K.

OS MAJOR/MINOR = Versões limitantes (maior e menor) do sistema operacional.

Tamanho da Imagem: Tamanho virtual da imagem, contando todos os cabeçalhos. E o tamanho total deve ser múltiplo da seção de alinhamento.

Tamanho do Cabeçalho: Tamanho total do cabeçalho. O tamanho combinado de cabeçalho do DOS, cabeçalho do PE e a tabela de seções.

FILE CHECKSUM: Checksum do arquivo em si, é setado como 0 pelo linkador.

Flags de DLL: Indica qual o tipo de leitura que deve ser feita, processos de inicialização e terminação de leitura e de threads.

Tamanho reservado da pilha: tamanho de pilha reservado ao programa, o valor real é o valor efetivo, se o valor reservado não tiver no sistema ele será paginado.

Tamanho efetivo da pilha: tamanho efetivo.

Tamanho Reservado da HEAP: Tamanho reservado a HEAP.

Tamanho efetivo da HEAP: Valor efetivo para a HEAP.

### A.1.3 Tabela de Seções

O número de entradas da tabela de seções é dado pelo campo de número de seções que está no cabeçalho. As entradas se iniciam em 1. Segue imediatamente

o cabeçalho do PE. A ordem de dados e memória é selecionado pelo ligador. Os endereços virtuais para as seções são confirmados pelo ligador de forma crescente e adjacente, e devem ser múltiplos da Seção de alinhamento, que também é fornecida no cabeçalho do PE. Abaixo alguns de uma seção nesta tabela, divididos em palavras de 8 bytes:

Nome da Seção: Campo com 8 bytes nulos para representar o nome da seção em ASCII.

Tamanho virtual: O tamanho virtual é o alocado quando a seção é lida.

Tamanho físico: O tamanho de dados inicializado no arquivo para a seção. É múltiplo do campo de alinhamento do arquivo do cabeçalho do PE e deve ser menor ou igual ao tamanho virtual.

Offset físico: Offset para apontar a primeira página da seção. É relativo ao início do arquivo executável.

Flags da seção: Flags para sinalizar se a seção é de código, se está inicializada ou não, se deve ser armazenada, compartilhada, paginável, de leitura ou para escrita.

#### **A.1.4 Páginas de imagem**

A página de imagens contém todos os dados inicializados e todas as seções. As seções são ordenadas pelo endereço virtual reservado a elas. O Offset que aponta para a primeira página é especificado na tabela de seções como visto na subseção acima. Cada seção inicia com um múltiplo da seção de alinhamento.

#### **A.1.5 Importação**

A informação de importação inicia com uma tabela de diretórios de importação que descreve a parte principal da informação de importação. A tabela de diretórios de importação contém informação de endereços que são utilizados nas referências de correção para pontos de entrada com uma DLL. A tabela de diretórios de importação consiste de um vetor de entradas de diretórios, uma entrada para cada referência a DLL. A última entrada é nula o que indica o fim da tabela de diretórios.

### A.1.6 Exportação

A informação de exportação inicia com a tabela de diretórios de exportação que descreve a parte principal da informação de exportação. A tabela de diretórios de exportação contém informação de endereços que são utilizados nas referências de correção para os pontos de entrada desta imagem.

### A.1.7 Correção

A tabela de correção contém todas as entradas de correção da imagem. O tamanho total de dados de correção no cabeçalho é o número de bytes na tabela de correção. A tabela de correção é dividida em blocos de correção. Cada bloco representa as correções para uma página de 4K bytes. Correções que são resolvidas pelo ligador necessitam ser processadas pelo carregador, a menos que a imagem não possa ser carregada na Base de imagens especificada no cabeçalho do PE.

### A.1.8 Recursos

Recursos são indexados por uma árvore binária ordenada. O design como um todo pode chegar a  $2^{31}$  níveis, entretanto, NT utiliza somente 3 níveis: o mais alto com o *tipo*, no subsequente *nome*, depois a *língua*.

### A.1.9 Debug

A informação de debug é definido por um debugador que não é controlado pelo PE ou pelo ligador. Somente é definido pelo PE os dados da tabela de diretório de debug.

## A.2 Arquivo ELF

Explicação

### A.2.1 A estrutura do arquivo ELF

Arquivo Realocável	Arquivo Carregável
Cabeçalho ELF	Cabeçalho ELF
Tabela do cabeçalho do programa (opcional)	Tabela do cabeçalho do programa
seção 1	Segmento 1
seção 2	
...	Segmento 2
seção n	...
Tabela de cabeçalho de seção	Tabela de cabeçalho de seção(opcional)