Paradigmas da codificação dos vírus de computador – uma análise das estruturas arquiteturais internas

*Ivo Fabiano Pereira Simões**

RESUMO

Este ensaio visa apresentar parcialmente os conceitos dos vírus de computador e também uma parte das diversas estratégias de implementação utilizadas para garantir sua sobrevivência nos sistemas infectados. Em virtude de se tratar de um assunto potencialmente perigoso, os exemplos referentes aos vírus de computador serão apresentados com parcimônia e, ainda que funcionais, de maneira simplificada. Atualmente, as empresas de segurança informática e de produtos antivírus passaram ao conceito do Malware[1] *colocando os vírus de computador em uma mesma generalização classificativa com os* Cavalos de Troia[2], Worms[3] *e outras potenciais ameaças computacionais. Neste ensaio vamos nos ater apenas ao estudo dos vírus de computador e seus paradigmas de arquitetura.*

PALAVRAS-CHAVE

Vírus de computador; Sistemas antivírus; Vida artificial.

Introdução

Há bem mais de 20 anos, as viroses computacionais habitam, em escalas variáveis de ocupação, os computadores do mundo. Ainda hoje os chamados vírus de computador são uma realidade em praticamente todos os sistemas informáticos, independente de plataforma operacional ou arquitetura de hardware. Ao longo dos anos, as viroses computacionais evoluíram em tecnologia, paradigmas e complexidade. Semelhantes aos vírus biológicos, os vírus de computador buscam

* Graduado em Tecnologia em Processamento de Dados (Universidade da Cidade); Graduando em Engenharia Elétrica (UERJ); Pós-graduado *Lato Sensu* em Análise, Projeto e Gerência de Sistemas (Universidade da Cidade); Mestrando em Sistemas e Computação – Instituto Militar de Engenharia-IME; Diretor de Tecnologia e Pesquisas da Tachion Tecnologia & Sistemas. Estudioso dos Vírus de computador desde 1999.

[1] *Malware* é o termo genérico com que a indústria de segurança passou a classificar qualquer programa que se entenda como possivelmente perigoso a um sistema computacional.

[2] *Cavalo de Troia* ou *Trojan* é, genericamente falando, um programa que abre um acesso para o sistema onde ele se localiza, tipicamente através de uma porta de rede.

[3] *Worm* ou *Verme* é um programa semelhante ao vírus de computador, porém ele não é capaz de criar uma cópia de si mesmo injetando o seu código em um arquivo hospedeiro. Assemelha-se ao comportamento biológico de um verme parasita.

criar cópias de si mesmos para, com isso, manter sua presença e "sobrevivência" no(s) sistema(s) em que se alocam. Participantes da metodologia de preservação da própria espécie passam a se enquadrar nos requisitos, tidos hoje como mínimos, para o que se considera uma forma de vida, e são anexados ao contexto do recente campo de pesquisa denominado "vida artificial". Muito se evoluiu no campo da virologia computacional. Os vírus modernos são capazes de se automodificar ("Polimorfismo", "Metamorfismo" etc.), alterando sua assinatura de detecção conhecida e/ou estrutura interna; assim, mantêm sua sobrevivência no sistema por muito tempo, garantindo período longo para poderem se replicar internamente e para outros computadores.

Breve resumo sobre a história dos vírus de computador

Desde que apareceu o primeiro vírus de computador "moderno", sob as concepções atuais (1986) [3], até o dia de hoje, calcula-se algo em torno de 100.000 a 150.000 vírus existentes, segundo os catálogos de produtos antivírus mais completos. A quantidade real, tirando as variantes dos vírus e outros artefatos, que na realidade não são vírus (como *Phishing Scans, Worms, Cavalos de Troia,* etc.), pode-se chegar apenas aos 18.000 ou 28.000 vírus. É quase impossível determinar com exatidão esses números, visto que muitos códigos virais não foram difundidos e/ou descobertos.

Existem várias pesquisas que tentam determinar de onde se originou a ideia. Provavelmente o primeiro que teceu uma definição do que até então existia foi o matemático John Von Neumann, ao publicar um artigo em 1949, chamado *Theory and Organization of Complicated Automata* [2], tratando a respeito de "uma porção de código capaz de reproduzir a si mesmo". Já pelo final dos anos 1950, iniciam-se as *Core Wars*[4] (guerras de núcleo), por H. Douglas McIlroy, Victor Vysottsky e Robert Morris Sr., investigadores de inteligência artificial dos laboratórios Bell. Dois programas hostis, escritos em uma linguagem "pseudomontada", chamada *RedCode*, podiam crescer na memória e lutar entre si.

Exemplos:

Programa IMP

Nome: Imp
Comportamento: Cria uma cópia de si mesmo na próxima posição de memória e move para a próxima instrução a ser executada.
Código:
imp: MOV imp, imp+1

Programa DWARF

Nome: "DWARF" (pigmeu)
Comportamento: Coloca uma bomba "0" em cada 5 posições de memória, para tentar "zerar" os dados e instruções dos programas inimigos. Como esse programa é composto de apenas 4 instruções, ele fica em um intervalo, sem ser bombardeado pelo "0".
Código:
1001: DAT -1
1002: ADD #5 -1
1003: MOV #0 @-2
1004: JMP –2

Em 1970, Bob Thomas criou um programa denominado *Creeper* (repetidor), que viajava pelas

[4] http://www.corewar.info/

redes e era usado pelos controladores de tráfego aéreo para ceder o controle de uma aeronave de um terminal para outro.

No início dos anos 1980, John Shock e Jon Hupp, do centro de investigação da Xerox, disseminou um "programa-worm" para troca de mensagens e tarefas automáticas durante a noite, mas que acabou trabalhando de forma incontrolada, e tiveram que eliminá-lo. Nesse mesmo ano, estudantes de Ciência da Computação escreveram para o *AppleII,* um programa que se autorreproduz.

Em 1983, Ken Thompson recebia o prêmio Alan Turing e surpreendia a todos com um discurso baseado nas *Core Wars*, em que estimulava a todos os usuários e programadores a experimentar essas "criaturas lógicas", o que causou rebuliço no alto escalão da computação da época.

Em 1984 e nos anos seguintes, apareceu na prestigiada revista americana *Scientific American* uma série de artigos de A. K. Dewney, a qual revelava ao grande público a existência e as características dos *Core Wars*.

Em 1985, um estudante da universidade da Califórnia do Sul, chamado Fred Cohen [1], completava sua tese sobre programas "autoduplicadores" (iniciada em 1983). Foi, na realidade, o orientador de sua tese quem sugeriu o nome de "vírus de computador". Havia publicado um artigo na "IFIPsec 84" (*International Information Security Conference*), intitulado *Computer Víruses. Theory and experiments*, em que estabelecia uma definição acadêmica para os vírus de computador como "um programa que pode infectar a outros programas, incluindo uma cópia possivelmente evoluída de si mesmo".

Pode-se dizer que foi no ano de 1986 que apareceram os primeiros vírus no sentido que os conhecemos hoje. Os vírus para *Apple II,* especialmente o *Elk Cloner*, e os ensaios acadêmicos de Cohen tinham outras considerações.

O conceito verdadeiro de "vírus de computador"

O vírus de computador é um programa que se assemelha a um vírus biológico. Inicialmente, o termo "vírus" (do latim: veneno) foi utilizado como sinônimo de veneno e se referia a agentes de natureza desconhecida que provocavam diversas doenças. Vírus de computador são programas capazes de se autorreplicar e se anexar a outros arquivos (infecção), de forma que, quando um arquivo infectado é acionado, o código do vírus é executado juntamente e busca realizar nova infecção.

Os sintomas de uma infecção por um vírus de computador em um sistema variam de acordo com a tecnologia usada pelo vírus bem como pelo seu tipo. Muitos dos vírus mais recentes simplesmente não apresentam sinais de infecção nem possuem características de manifestação perceptível a fim de se manterem despercebidos o maior tempo possível.

Quando os vírus de computador foram criados, os seus implementadores colocaram como característica básica apresentar comportamentos similares aos de vírus biológicos. Muitos dos programadores que criaram os populares vírus não tinham ideia de que, um dia, programas dessa categoria teriam quesitos básicos para ser considerados criaturas com vida artificial. Realizando um paralelo entre as características dos vírus biológicos e a sua correspondência com os vírus de computador temos:

– *Reprodução*: Os vírus de computador se reproduzem a partir da sucessiva cópia de seu

código, podendo com isso infectar outros programas ou outros computadores.

– *Crossing-over:* Alguns vírus de computador sofisticados conseguem recombinar características entre si para produzir novos tipos de vírus.

– *Latência:* Alguns vírus de computador permanecem inertes até que as condições ambientais lhes sejam favoráveis (ex.: os vírus que se manifestam em dias marcados como o Sexta-feira 13, Michelangelo etc.).

– *Alimentação:* Os vírus de computador retiram energia e informações (dados) do hospedeiro.

– *Mimetismo:* Alguns vírus de computador, conhecidos como vírus furtivos (técnicas *stelth*), tentam evitar a sua detecção, forjando informações, fazendo com que o sistema fique com a mesma aparência de antes da infecção.

– *Mutação:* Muitos vírus de computador se modificam cada vez que duplicam. É importante ressaltar que essa modificação, notada na mutação, pode acontecer não intencionalmente, isto é, ao ser transferido de um computador a outro, por exemplo, algumas partes do vírus podem sofrer alterações.

Tecnologias relacionadas a vírus de computador

As tecnologias virais são usadas para criar vírus mais complexos e com maiores capacidades. As principais aplicações e utilizações dessas tecnologias são:

– Prolongar o tempo para escrever código antiviral (demorar a se criar uma cura);

– Aumentar e dificultar o tempo de detecção de objetos infectados;

– Manter o vírus ativo e replicando-se pelo maior tempo possível;

– Restringir ao máximo a aquisição de conhecimento (*Know How*) viral por parte das empresas antivírus (ex.: *antidebug,* autodestruição).

a) Métodos Anti-Heurísticos

Anti-Heurística é a tecnologia usada pelas viroses computacionais para burlar a proteção dada pelos métodos heurísticos e emuladores dos sistemas antivírus. Diversas técnicas são utilizadas nesse intento.

Os sistemas antivírus tentam pseudoexecutar e/ou emular o código do arquivo suspeito na busca de indícios característicos de viroses computacionais. A primeira forma de se tentar evitar isso é usar funções não documentadas, as quais provavelmente os emuladores dos antivírus não "sabem pseudoexecutar". As funções não necessariamente têm de estar fazendo parte do algoritmo funcional do vírus (vide *antidebug*) podendo apenas estar em algum lugar do código viral para iludir a heurística.

Outro método, mais antigo, porém funcional com diversos produtos atuais, é "emular" o término do código viral com uma instrução de salto incondicional ("JMP") para o final do arquivo ou o próprio início, mas que, na verdade, não salte para o destino especificado. Com isso, muitos emuladores heurísticos param de analisar o arquivo. Importante é ressaltar que essa técnica se torna mais funcional se colocada antes da rotina de codificação ("encriptação") do vírus, pois toda codificação ("encriptação") é tida como "suspeita" pelos sistemas antivírus. Exemplo:

Emulação anti-heurística de término

```
CALL AV
FORA4AC:
   MOV AH,4AC; 4AC em AX termina a exe-
      cução na interrupção 21h
      INT 21H
      JMP SEGUE
AV:
   MOV BYTE PTR [FORA4AC+1],2CH
   RET
SEGUE:
...
```

Outra técnica (bem mais antiga) é de "estourar" o tempo de verificação heurística do arquivo. Tendo-se em conta que alguns dispositivos heurísticos dos antivírus possuem um tempo máximo de verificação e pseudoexecução de código, utiliza-se um grande laço de repetição na tentativa de que esse tempo passe e então o mecanismo de verificação heurística pare a sua verificação. Por exemplo:

Emulação anti-heurística de "estouro de tempo"

```
MOV CX,0FFFFH; valor para o loop (Máximo
      num reg de 16 bits)
ANTI_1:
      JMP     ANTI_2   ; salto incondicional p/
      anti_2
      MOV   AX,4c00H ; isso "terminaria" o pro-
      grama
      CALL    int 21H  ; chamada do Sist Op
ANTI_2:
     LOOP    ANTI_1   ; loop anti_1
```

b) Residência em memória

Residência em memória é a técnica que permite ao vírus de computador permanecer alocado em um espaço de memória RAM. Essa tecnologia permite diversos benefícios para a infecção viral. Como exemplo, um vírus de computador que infecte no evento de abertura de arquivos e que esteja residente em memória pode infectar inúmeros arquivos ao ser executada uma varredura por um *software* antivírus, por exemplo.

A maioria dos *software* antivírus executa uma verificação no vetor de memória principal em busca de viroses residentes. Isso provocou o surgimento das técnicas de invisibilidade em memória. A técnica de residência em memória atualmente é usada (quando é usada) por períodos curtos a fim de deixar o vírus o mais despercebido possível, visto que o monitoramento de memória e os algoritmos heurísticos dos *software* antivírus evoluíram bastante em *performance* e acurácia.

c) Retrovirosidade

Os retrovírus são viroses computacionais que usam técnicas para anular a ação e/ou desativar os mecanismos dos sistemas antivírus. Partem do princípio que, se o antivírus não funcionar, mas parecer que está funcionado, ter-se-á um ambiente favorável por muito mais tempo para que as replicações ocorram, e a propagação do vírus seja garantida.

As técnicas de "retrovirosidade" tentam desativar da memória os mecanismos de verificação em tempo real, adulterar arquivos do antivírus que sejam responsáveis por alertas e/ou afins, bem como remover o carregamento do sistema antivírus durante a inicialização do sistema operacional. Obviamente que quanto mais sutis forem as ações, mais garantidos serão os resultados.

Abaixo é apresentado um antigo exemplo de código demonstrando uma rotina com esse objetivo. Essa rotina foi retirada a partir de um programa criador de viroses computacionais (N.R.L.G - *Nuke Randomic life Generator*). A rotina visa remover da memória o antigo antivírus VSAFE:

Exemplo de código "retroviral"

```
ANTI_V:
MOV AX,0FA01H;REMOVE VSAFE DS MEMORIA
MOV DX,5945H
INT 21H
RET
```

Existem relatos de vírus que foram lançados em conjunto, apenas com um pequeno tempo de

diferença entre os dias, em que o primeiro afetava os sistemas antivírus, "preparando" o caminho para o segundo se propagar. Esse relato é conhecido no meio da comunidade de "Vx"[5], mas o autor deste trabalho não encontrou qualquer documentação ou código a respeito desses vírus.

d) Encriptação

A "encriptação" é o método pelo qual os codificadores virais permitem aos vírus "esconder" o seu código, dificultando a desmontagem (*unassembled*) do mesmo e, com isso, facilitando o seu estudo para uma "vacina" ou afim. A "encriptação" viral se divide basicamente em dois tipos: a de chave fixa e a de chave variável.

A "encriptação" com chave fixa não é mais usada e há tempos é tida como um verdadeiro "pecado" no meio dos "Vxs", pois facilita grandemente a descoberta da decodificação ("desencriptação") de código e, com isso, a depuração do espécime viral. A chave variável pode ser conseguida, por exemplo, lendo-se um valor qualquer da CPU e atribuindo esse valor (que se torna aleatório para cada infecção) como chave de criptografia.

Por exemplo:

Escolhe-se um método de geração de um valor para a chave:

Código de geração de chave criptográfica

```
IN AL,4: pega um valor randômico do "clock"
MOV BYTE PTR [DECR], AL: salva o valor como própria chave
```

Depois disso, aplica-se a chave conseguida a uma rotina de criptografia:

Código de "encriptação"

```
ENCRIPT:
LOADSB : carrega um byte
XOR AL,BYTE PTR [DECR] ;faz uma op "xor" do
    byte com própria chave
 STOSB :armazena o byte
 LOOP ENCRIPT:loop
 RET:retorna para a chamadora
 DECRIPT DB 0: chave de decriptação valor 0
```

Na verdade, praticamente qualquer algoritmo de criptografia poderia ser usado, e uma criptografia com base no operador XOR pode tornar-se bastante robusta em um vírus de computador quando sequenciada ou complementada com a precedência de outras operações matemáticas.

e) Polimorfismo

O polimorfismo é uma técnica aplicada já há algum tempo e se tornou padrão para todos os tipos de vírus computacionais, sejam eles para DOS, *boot*, macro, Windows, Linux ou multiplataformas (Windows-Linux)[6], pois se tornou uma condição para sobrevivência da entidade artificial, sem a qual se torna uma presa fácil para os sistemas antivírus.

Os vírus polimórficos executam uma criptografia "mutante" do próprio código com uma chave aleatória e com combinações de vários algoritmos de codificação/decodificação ("encriptação/desencriptação"), objetivando a modificação do próprio código executável. Com a aplicação dessa polimorfia, os mesmos passam a não poder ser detectados pela técnica de assinaturas, pois o seu código é modificável.

[5] "VX" é uma abreviatura usada para designar a comunidade mundial de codificadores de vírus.

[6] Desmitificando um errôneo pensamento geral, existem vírus para Linux, Unix e para as mais diversas plataformas de S.O. conhecidos.

Fragmento de código para polimorfismo

```
MOV AL,[BP+VALUE]
MOV [BP+VALUE1],AL
MOV AH,40h
LEA dx,[BP+DEL1]
MOV CX,DEL2 - DEL1
INT 21h
LEA SI,[BP+C_START]
LEA DI,[BP+VIRUS_END]
MOV CX,VIRUS_END - C_START
CALL CRYPT1
JMP WRITE
CRYPT: ; Encriptação
LODSB
Push CX
NOP
MOV CL,4
ROL AL,CL
NOP
NEG AL
ROL AL,CL
NOP
POP CX
STOSB
NOP
LOOP CRYPT
RET
NOP
WRITE:
```

Alguns vírus polimórficos mais complexos utilizam algoritmos avançados para a geração do seu código decodificador ("desencriptador"); as instruções anteriores, que passam de um arquivo infectado para outro, podem ser misturas com instruções que não modificam nada (ex.: NOP, STI, CLC etc). O carregamento e a modificação de chaves e outros parâmetros de criptografia também são criados usando blocos aleatórios, que podem ter quase todas as instruções dos processadores Intel (ex.: ADD, SUB, XOR, OR, SHR, SHL, PUSH, POP etc.), com todos os modos de endereçamento possíveis.

f) Metamorfismo

Existe muito pouca documentação a respeito da tecnologia de metamorfismo viral. Metamorfismo nos vírus de computador significa que seu vírus contém alguns dados criptografados, usados para gerar novas cópias polimórficas. A nova cópia polimórfica será executada depois de sua criação, que poderá conter instruções de diversos tamanhos, pois ocorre um "polimorfismo da cópia polimórfica".

g) Permutação

Os vírus permutantes realizam "trocas de posição" de blocos do próprio código. Eles são capazes de alterar a ordem de execução das rotinas internas de forma a ficarem diferentes, mas com um mesmo funcionamento. Vírus permutantes não contêm dados criptografados; eles usam uma cópia atual para gerar novas cópias polimórficas e sempre conseguem obter o tamanho das próprias instruções. Isso pode ser feito complementando todas as instruções para que estas sempre tenham um mesmo tamanho fixo (pode-se, por exemplo, complementar com instruções "NOP" até se atingir o tamanho necessário para se igualar com o tamanho dos outros blocos).

h) Invisibilidade

Entende-se por invisibilidade (técnicas *STELTH*), em vírus computacionais, a arte de implementar um vírus de computador que procura esconder sua presença e/ou ação do sistema como um todo

(S.O., antivírus e mesmo do usuário). Quando um arquivo é criado, modificado ou apagado, são geradas modificações diversas nas informações dos arquivos afetados; se alguma ação é executada no sistema, ocorrerão alterações na memória principal, e serão realizadas chamadas ao sistema operacional, solicitando interrupções, APIs e correlatos. Os vírus com tecnologia de invisibilidade procuram "apagar rastros de suas ações" como, por exemplo, retornando a hora e data do arquivo antes da infecção, o que impede, por observação explícita, a verificação de uma alteração no arquivo.

Código para restaurar a data e hora de um arquivo

```
    MOV  ax,5700h ; pega hora e data
    INT 21H ;
    MOV hora,dx ; salva a hora na variável
    "HORA"
    MOV data,cx; e salva a data na variável
    "DATA"
- >aqui entrariam ações de infecção etc.
    MOV ax,5701h ; restaura MOV dx,hora
    ; hora
    MOV cx,data ; data
    INT 21H ;
```

Como foi demonstrado, bastaria serem guardados os valores iniciais de data e hora para se ter uma "invisibilidade" no diretório. Complementa-se também restaurando os atributos originais do arquivo (que podem ter sido removidos para permitir a gravação no mesmo). Outra forma de invisibilidade no diretório seria a escolha aleatória de nomes parecidos com os de arquivos do sistema operacional (que geralmente são desconhecidos da maioria dos usuários comuns) em locais pouco acessados pelo usuário (como a pasta *system* do Windows) e também a alteração do atributo para "oculto" (*hidden*).

Alguns vírus residentes em memória ficam monitorando chamadas referentes à memória e, ao interceptarem alguma verificação, realocam-se para um setor já verificado ou para o disco rígido, "escondendo-se" da função de verificação por parte do antivírus. A essa formidável e engenhosa técnica se dá o nome de "invisibilidade em memória". Atualmente, com o monitoramento de eventos do sistema operacional em operações com arquivos no disco rígido, a invisibilidade só é possível se for realizada uma nova cópia do arquivo viral com um polimorfismo ou metamorfismo para um outro local.

i)"Antidebug" (Antidepuração)

As técnicas "*antidebug*" tem por finalidade evitar ou pelo menos dificultar ao máximo que um analista de vírus ou afim consiga desmontar (unassembled) o vírus. Quanto mais tempo a estrutura do vírus ficar oculta, mais tempo ele terá para sobreviver nos sistemas e poder se propagar, visto que ainda não se terá uma vacina ou método de desinfecção.

Basicamente a técnica consiste em utilizar operações "confusas" para "sujar" o algoritmo ou escrever uma rotina que simule a invocação de uma interrupção quando, na verdade, se quer apenas lançar confusão. A própria técnica de encriptação é utilitária no intento do "*antidebug*", pois dificultará o trabalho do analista na "dissecação" do vírus.

Um exemplo simples, para demonstração, seria quando se deve utilizar variáveis para marcar se algo é verdadeiro ou falso (tipicamente FLAGS, sejam elas *booleanas* ou multivaloradas). Quando alguém observa algo como uma

comparação com "Y" ou "N", rapidamente atenta para o que se trata:

Exemplo de técnica *antidebug*

```
CMP BYTE PTR [FLAG],'Y'
...
MOV BYTE PTR [xxxx],'N'
...
```

A rotina seguinte tem um código-fonte que continua sendo entendido pelo programador, mas quando se depura é muito difícil de se dar conta que esse "84H" é para uma comparação "booleana" (SIM/NÃO).

Outro exemplo de técnica "*antidebug*"

```
YES  EQU 84H
NO    EQU 27H
...
CMP BYTE PTR [FLAG],YES
...
```

Objetiva-se muito que seja usada a estratégia do "menos óbvio possível" tendo-se em conta que quem desmonta o programa (unassembled) não sabe que o código do vírus está com embustes e capturas de "APIs" ou "interrupções" ASM dissimuladas. Logo essa tarefa, além de consumir muito mais tempo, provoca cansaço ao analista, que facilmente poderá cometer erros e aumentará "escalavelmente" o tempo do processo de estudo do vírus como um todo, podendo incrementar-se de dias para semanas o período de estudo e solução para o vírus.

j)"Autodestruição"

A "autodestruição" é uma característica muito pouco encontrada nas viroses computacionais. Essa técnica é muito rara e consiste no fato de conseguir apagar suas próprias cópias depois de um tempo prefixado ou no caso de sua remoção do diretório corrente. A "autodestruição" objetiva principalmente não permitir a depuração e o estudo do vírus, impedindo que uma vacina ou afim seja criada. Apesar de ser uma técnica muitíssimo interessante, não foi adotada na comunidade "Vx" por não ser tão eficiente quanto a encriptação e o *antidebug* bem como por gerar vírus maiores e mais difíceis de serem implementados.

Já se ouviu bastante falar no meio da comunidade codificadora viral sobre alguns vírus capazes de apagar parte do seu próprio código quando percebem uma ação ou sua abertura por parte de algum programa (tipicamente uma depuração). Doravante a implementação de um vírus de computador com tal característica funcional não é trivial e deve ser apenas uma lenda temática.

Conclusões

A disseminação do conhecimento tecnológico dos vírus de computador pode permitir que diversos artefatos sejam implementados para proteção e resguardo de qualquer ameaça virtual, sem que haja a dependência de uma empresa específica ou mesmo do lançamento de uma correção para uma virose computacional qualquer. Espera-se que este ensaio possa ser um real incentivo de estudo e pesquisa nesse universo formidável dos vírus e antivírus.

Referências

[1] COHEN, Fred "Computer Viruses – Theory and Experiments". Disponível em: http://www.all.net/books/virus/index.html. Acesso em: 1º de março de 2010.

[2] NEUMANN, John von. Theory and Organization of Complicated Automata. In: A. W. Burks (Ed.) *Theory of Self-Reproducing Automata [by] John von Neumann*. Urbana 1949. (pp. 29-87).

[3] SIMÕES, Ivo Fabiano Pereira. *Projeto DA VINCI - Viroses Computacionais e Vida Artificia*l 2004.