PROPOSTA DE ARQUITETURA E PROTOCOLO PARA A GERAÇÃO AUTOMÁTICA DE ASSINATURAS DE *MALWARES*

Autor: Gustavo André Arrabal de Souza

Orientador: Prof. Dr. Marcos Antônio Cavenaghi

0 - Agenda

- 1- Introdução;
- 2- Introdução aos *Malwares*;
- 3- Proposta de Arquitetura e Protocolo;
- 4- Sistema Operacional *Windows;*
- 5- Conclusões e Trabalhos Futuros;
- Referencias Bibliográficas.

1- Introdução

A. Introduções Gerais;

B. Problema;

C. Objetivo.

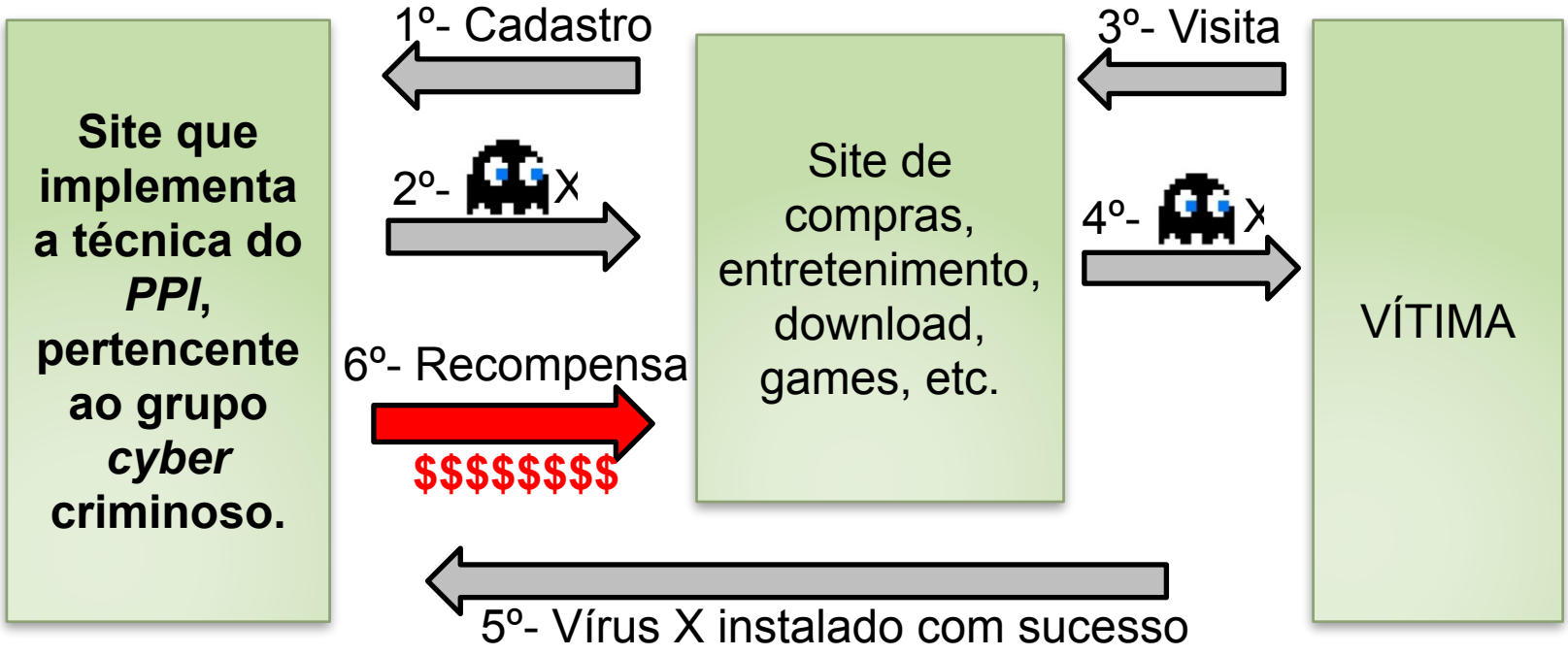
1- Introdução
A. Introduções Gerais

- Aumento do número de *malwares* (*malicious software*) na internet;

- *Kaspersky Lab* detectou e neutralizou mais de 1 bilhão de ameaças somente no segundo trimestre de 2012 (IT... , 2012).

1- Introdução
A. Introduções Gerais

- Disseminação de códigos maliciosos são práticas apoiadas e financiadas por grupos *cyber* criminosos que utilizam de estratégias do tipo *Pay-Per-Install* (STEVENS, 2010);



1- Introdução

A. Introduções Gerais

Figura 1: *Dogma Million*: site que faz pagamentos a quem contribui disseminando os seus *rootkits* (MATROSOV; RODIONOV, 2012).

1- Introdução
A. Introduções Gerais

- Propagação de vírus para criação de *botnets* (COMPUTER. . . , 2012).

1- Introdução

A. Introduções Gerais

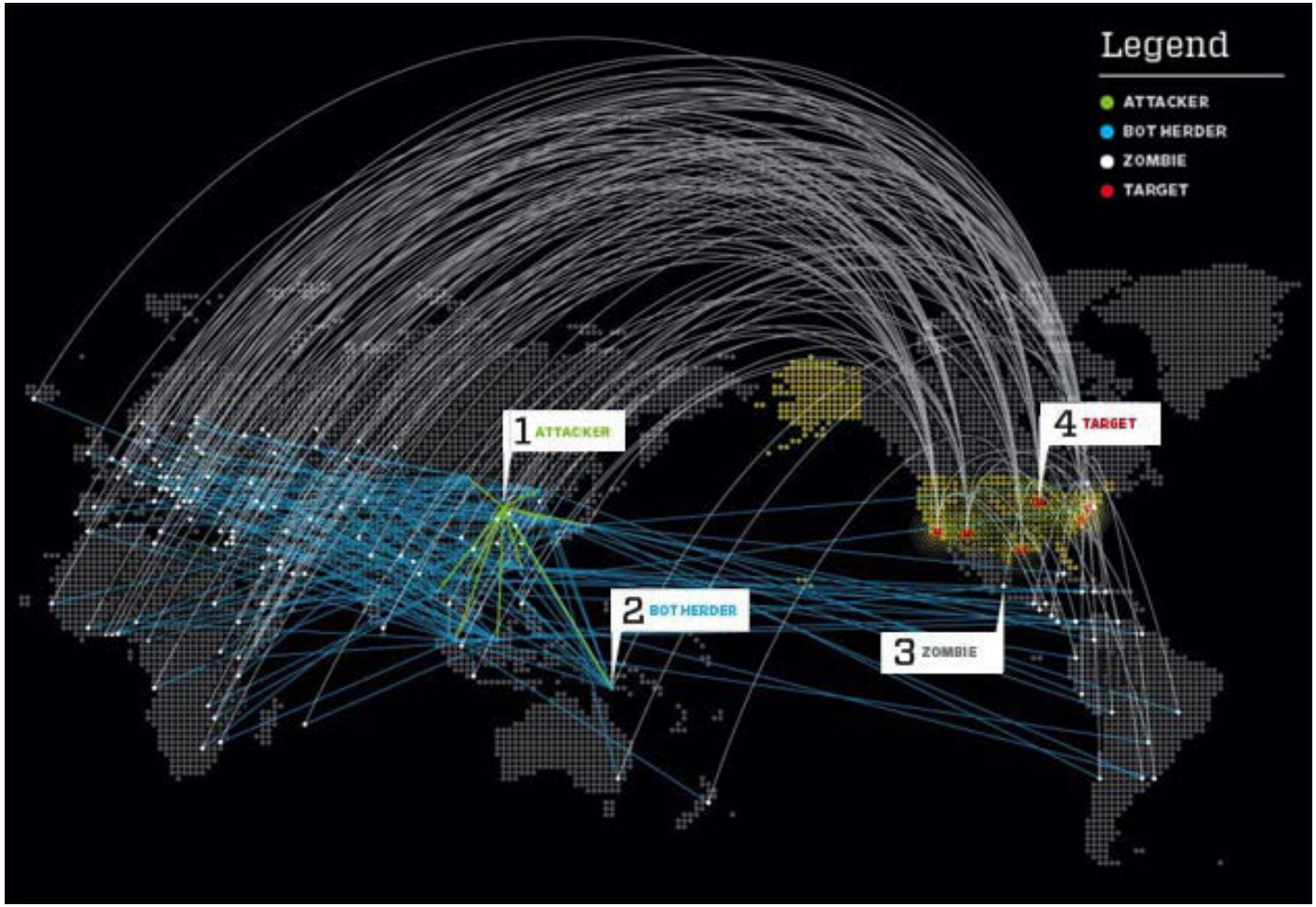


Figura 2: Exemplo de uma rede *Botnet* (SOCIAL. . . , 2012).

1- Introdução
A. Introduções Gerais

- Assinatura:
  - Técnica que busca extrair padrões dos *malwares* afim de, posteriormente, conseguir identificá-los. Atualmente existem duas abordagens na extração de assinaturas para vírus (ALAZAB et al., 2010)

1- Introdução
B. Problema

- *Malwares* criados são de difícil detecção por utilizarem técnicas que dificultam sua análise;
- Possuem a habilidade de auto-mutação;
- Dificuldade de criação e implementação de regras em IPSs (*Intrusion Prevention System* - Sistema de Prevenção de Intrusos) e em sistemas de antivírus.

1- Introdução
C. Objetivo

- Propor uma arquitetura física e um protocolo lógico para o auxílio na identificação de intrusões em sistemas computacionais;
- Geração automática de Assinaturas e Regras de IPS de forma híbrida;
- Assinaturas capaz de identificar *malwares* provenientes de sistemas de mutação ou não.

2- Introdução aos *Malwares*

A. *Static Malware;*

B. *Polymorphic Malware;*

C. *Metamorphic Malware:*
   A. *Obfuscation;*
   B. *Anti-VM.*

2- Introdução aos *Malwares*
A. *Static Malware*

- Mais simples entre os demais;
- Fácil detecção.

2- Introdução aos *Malwares*
B. *Polymorphic Malware*

- Tem a habilidade de criar mutações de si mesmo sem alterar sua funcionalidade.

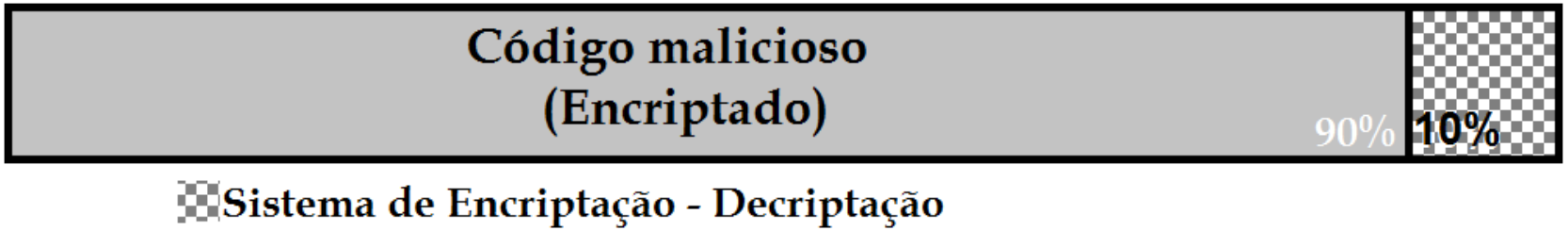


Figura 3: Estrutura de um *Polymorphic Malware* (KAUSHAL; SWADAS; PRAJAPATI, 2012).

2- Introdução aos *Malwares*
B. *Polymorphic Malware*

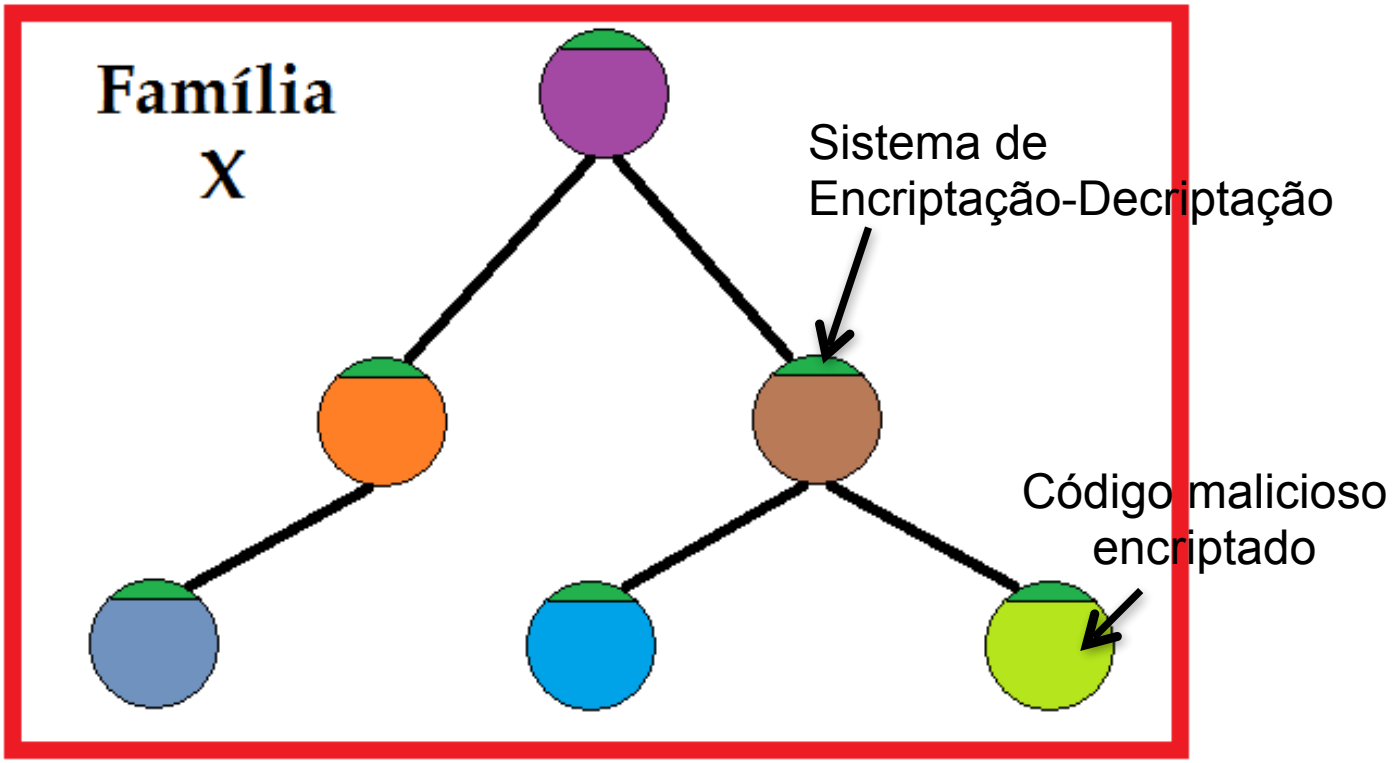


Figura 4: Árvore genealógica de um *Polymorphic Malware*.

2- Introdução aos *Malwares*
C. *Metamorphic Malware*

- Não utiliza técnicas de encriptação;
- Capacidade de criar código totalmente diferente a cada variação, impedindo assim que haja trechos de códigos iguais mantendo sua funcionalidade.

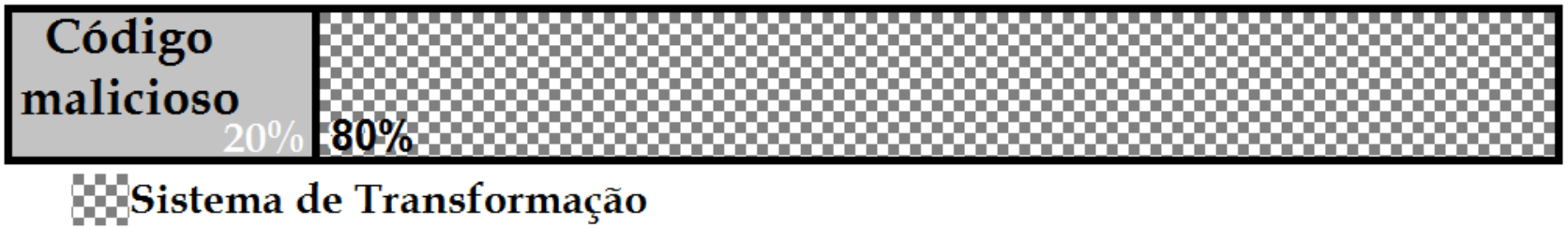


Figura 5: Estrutura de um *Metamorphic Malware* (KAUSHAL; SWADAS; PRAJAPATI, 2012).

2- Introdução aos *Malwares*
C. *Metamorphic Malware*

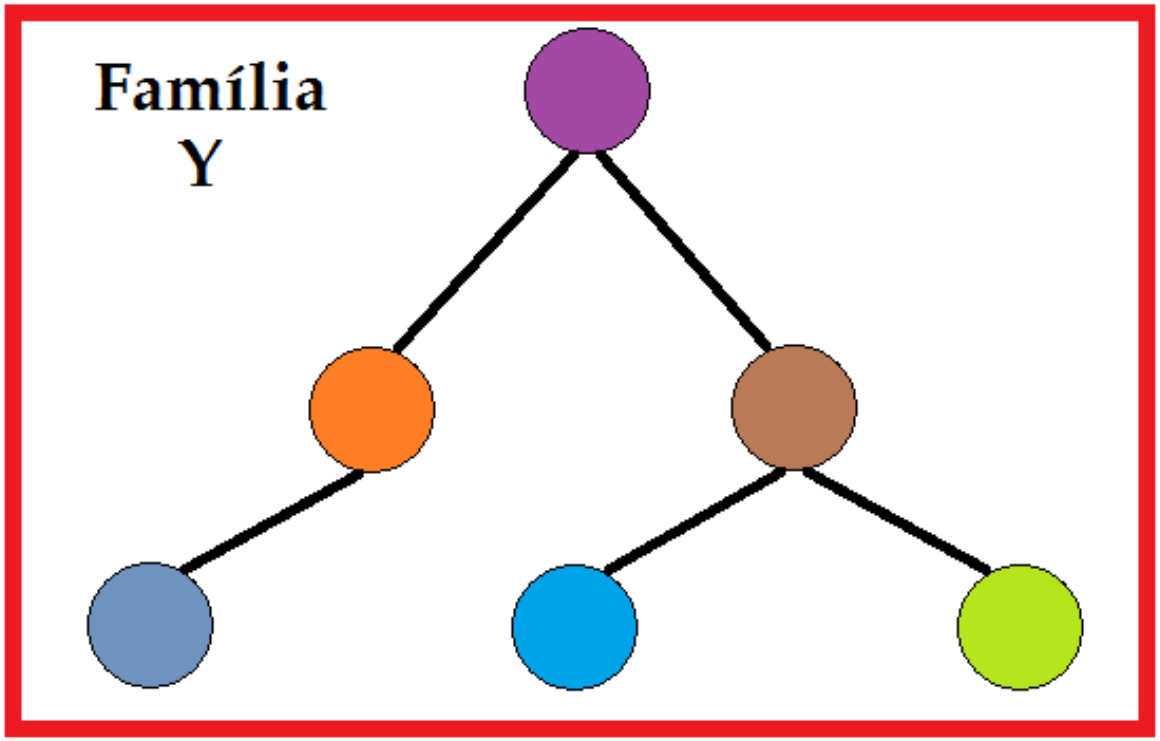


Figura 6: Árvore genealógica de um *Metamorphic Malware.*

2- Introdução aos *Malwares*
C. *Metamorphic Malware*

- Sistemas de Transformação:
  - *Anti-Debugging;*
  - *Anti-Disassembly;*
  - *Obsfucation;*
  - *Anti-VM.*

2- Introdução aos *Malwares*
C. *Metamorphic Malware*
A. *Obfuscation*

- *NOP Sequence (CHRISTODORESCU; JHA, 2003);*
- *Instruction Substitution (BRANCO; BARBOSA; NETO, 2012);*
- *Register Reassignment (CHRISTODORESCU; JHA, 2003);*
- *Code Transposition (YOU; YIM, 2010).*

2- Introdução aos *Malwares*
C. *Metamorphic Malware*
A. *Obfuscation*

```
xor eax,eax
inc eax
push ebx
...
```

NOP Sequence

```
xor eax,eax
NOP
NOP
inc eax
NOP
NOP
NOP
push ebx
NOP
...
```

```
xor eax,eax
mov ebp, esp
test esi, esi
...
```

Instruction Substitution, Register Reassignment

```
sub ebx,ebx
push esp
pop ebp
xor esi, esi
...
```

```
xor eax,eax
inc eax
push ebx
...
```

Code Transposition

```
        jmp .first
.second:
        push ebx
        jmp .continuation
.first:
        xor eax,eax
        inc eax
        jmp .second
.continuation:
        …
```

2- Introdução aos *Malwares*
C. *Metamorphic Malware*
B. *Anti-VM*

- *Malwares* se autodestroem quando percebem que estão sendo executados em *VMs*;
- Seus autores pressupõe que quando seus vírus estão em tal ambiente é porque tal ameaça esta sob análise de especialistas.
- Ex.: Executar um comando a nível de *kernel*.

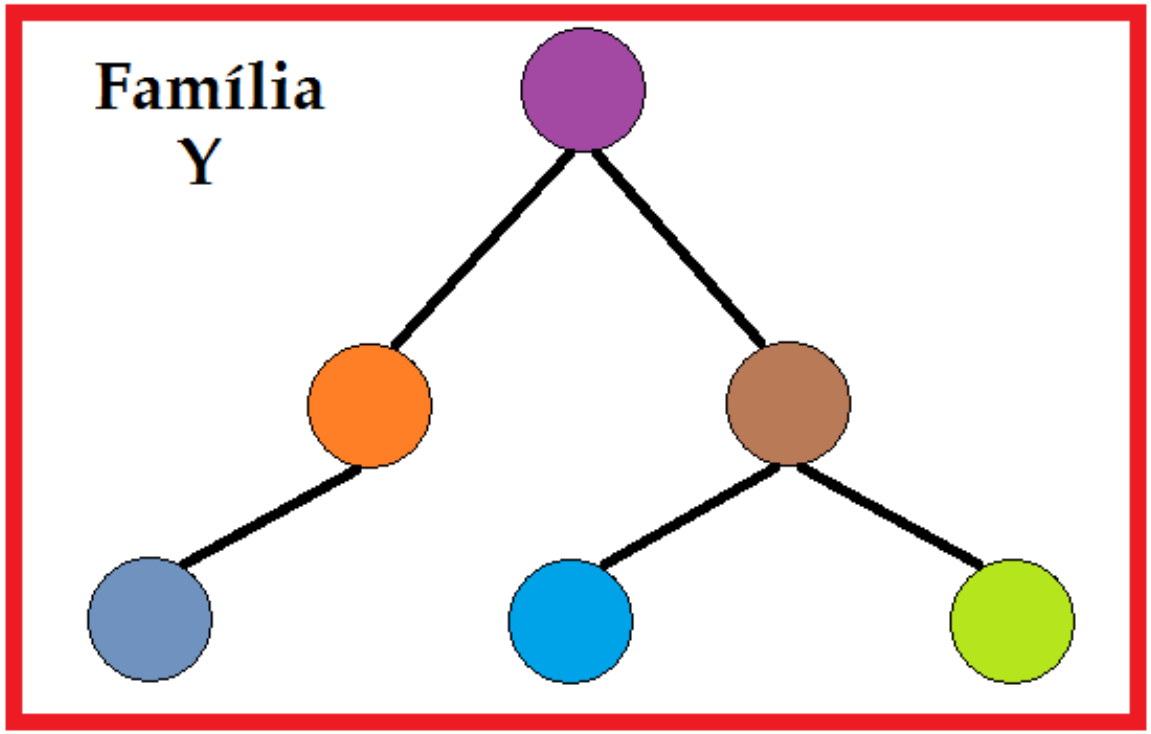
3- Proposta de Arquitetura e Protocolo

A. Arquitetura;

B. Protocolo.

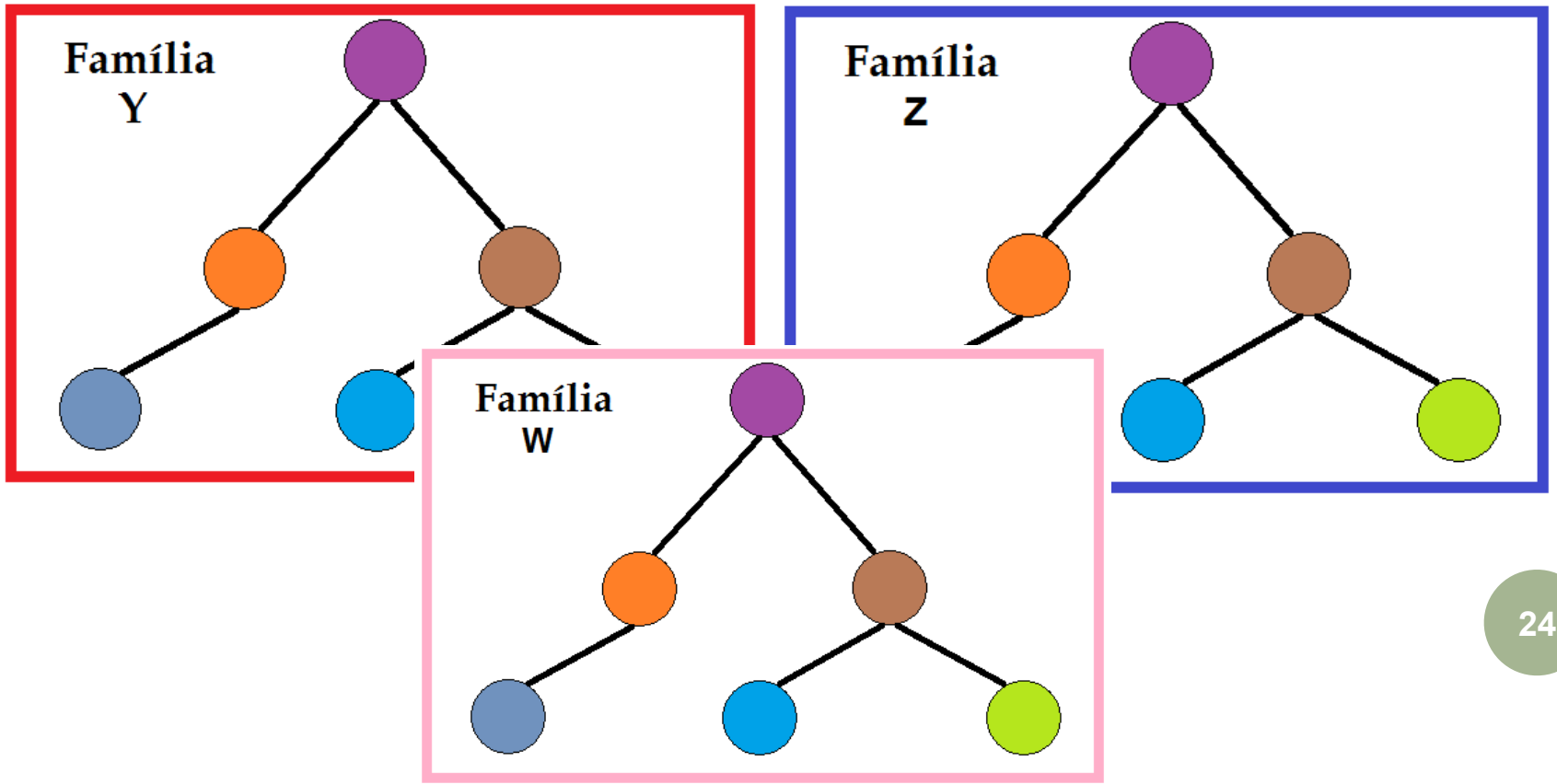
3- Proposta de Arquitetura e Protocolo

- **O que ocorre:** Análise do código afim de se criar uma assinatura.



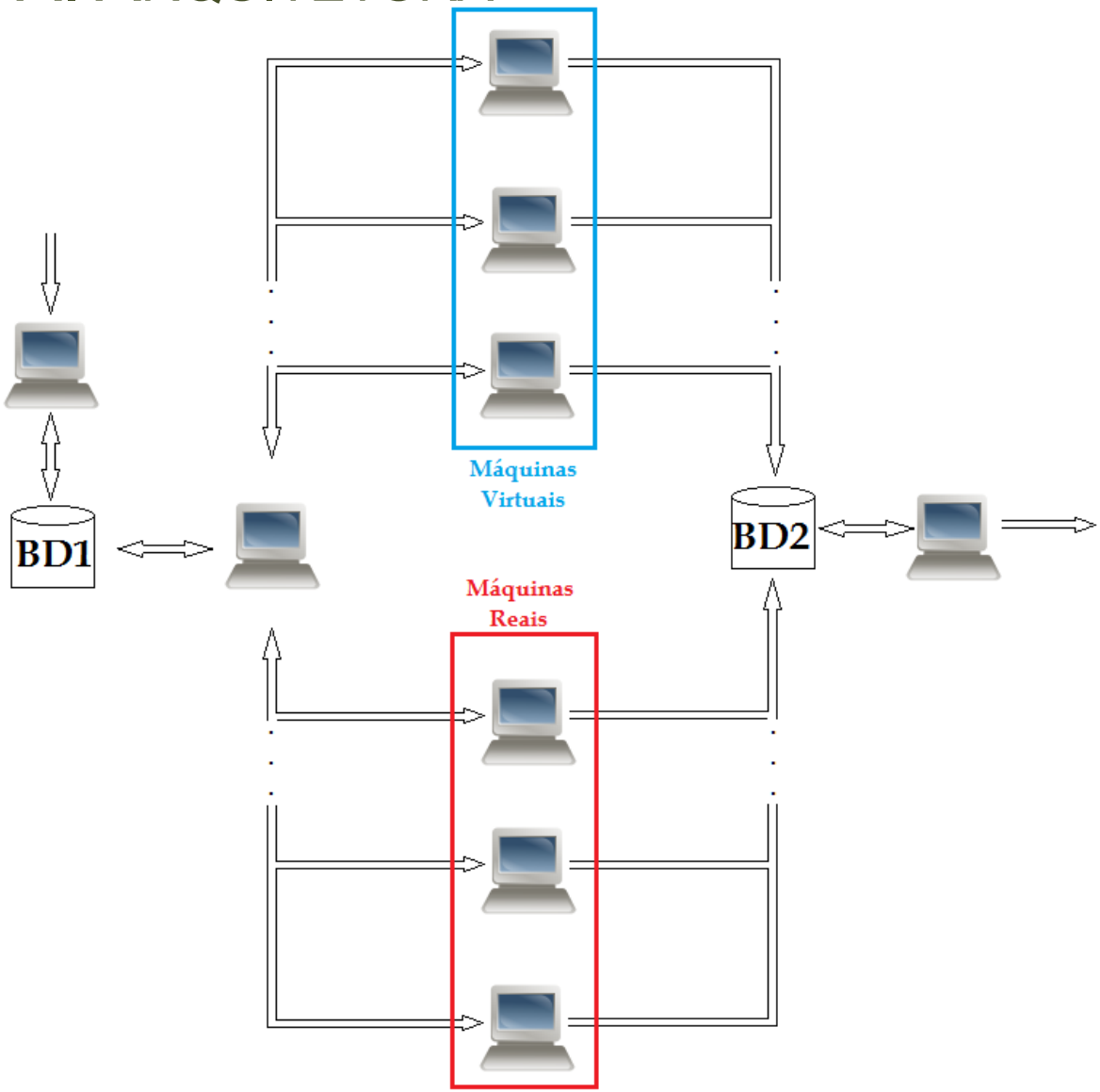
3- Proposta de Arquitetura e Protocolo

- **Proposta:** Criação de uma arquitetura de análise híbrida, ou seja, baseada no comportamento e código do *malware*.

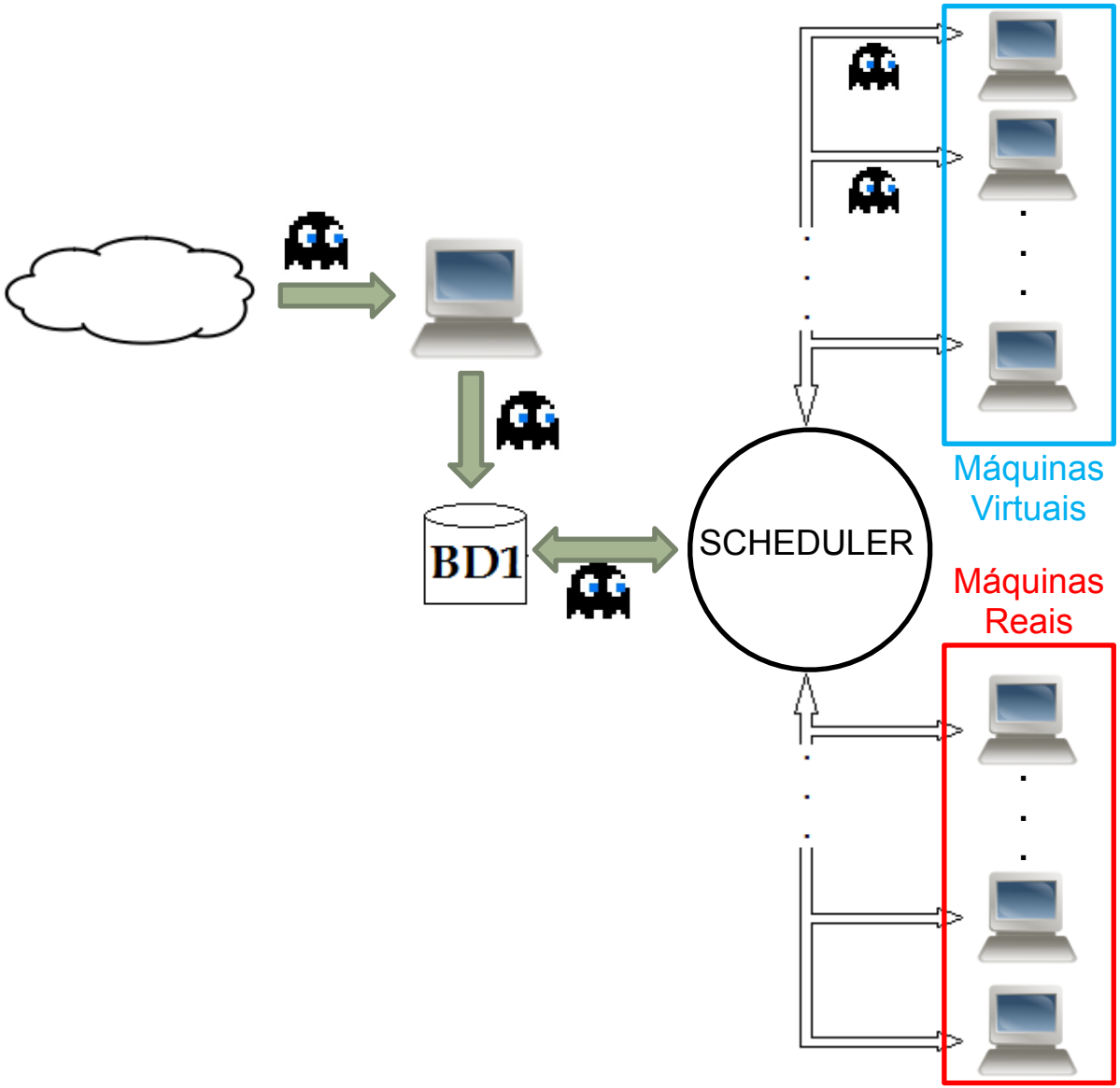


3- Proposta de Arquitetura e Protocolo

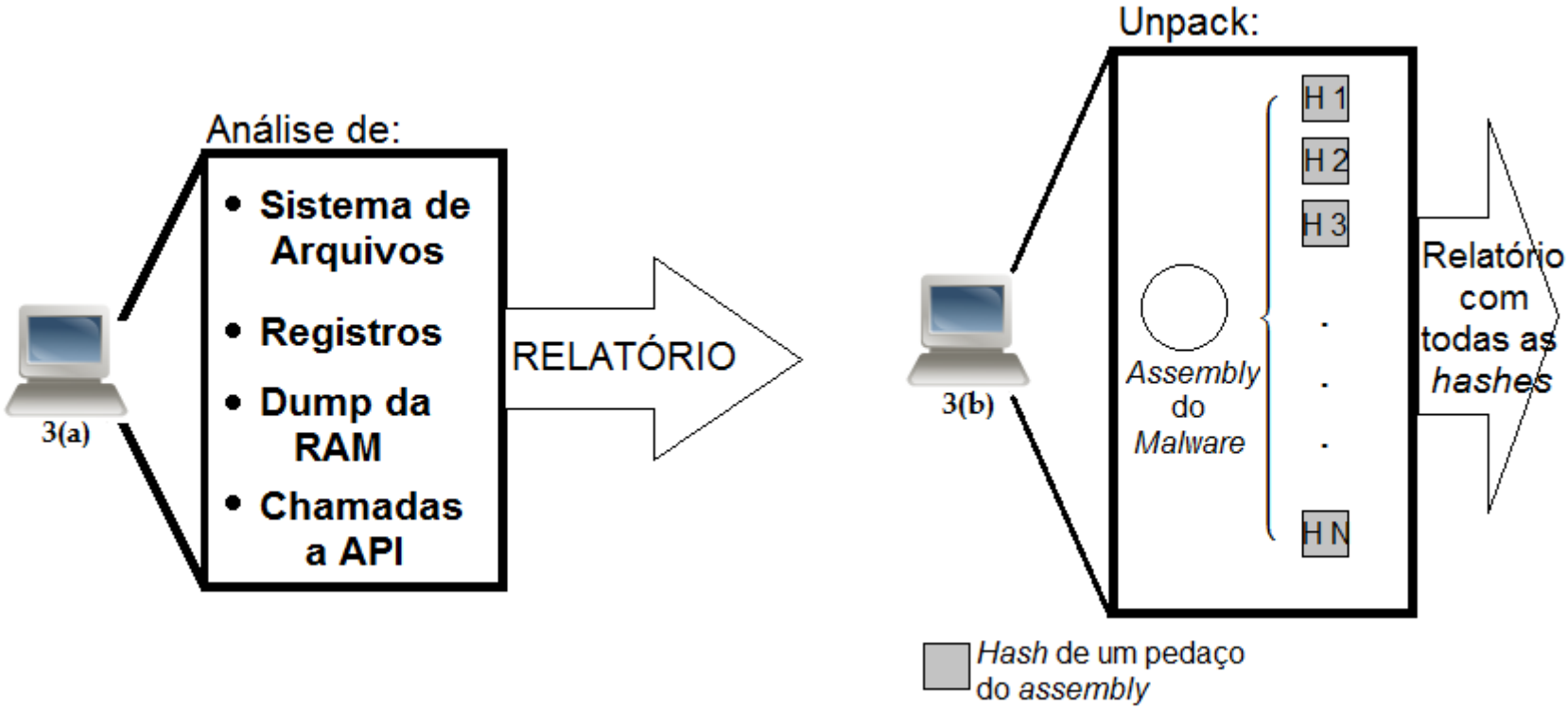
A. Arquitetura



3- Proposta de Arquitetura e Protocolo
B. Protocolo



3- Proposta de Arquitetura e Protocolo
B. Protocolo

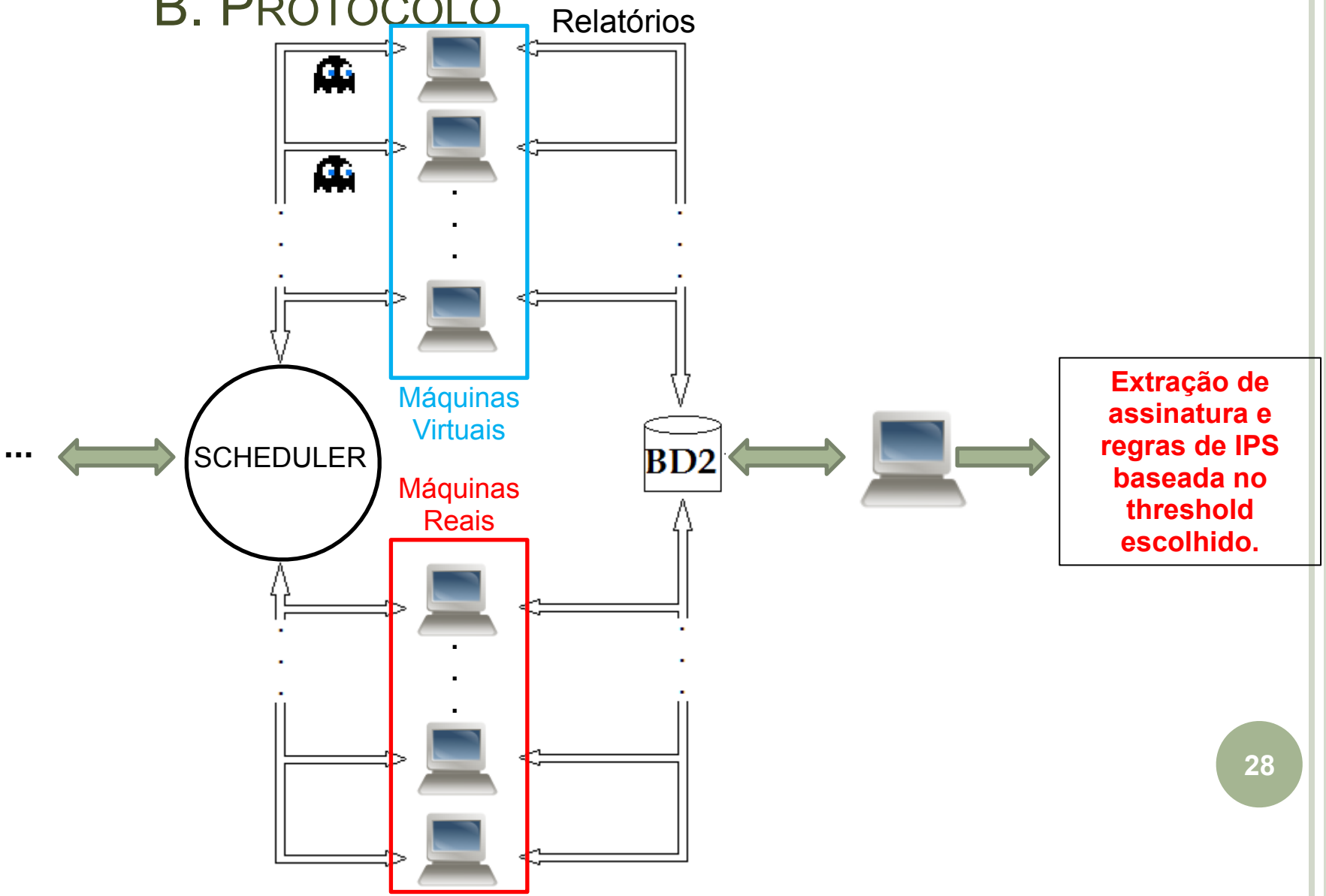


(a): Análise do comportamento do *malware.* **(b):** Análise do código do *malware.*

Figura 7: Subetapas do protocolo proposto.

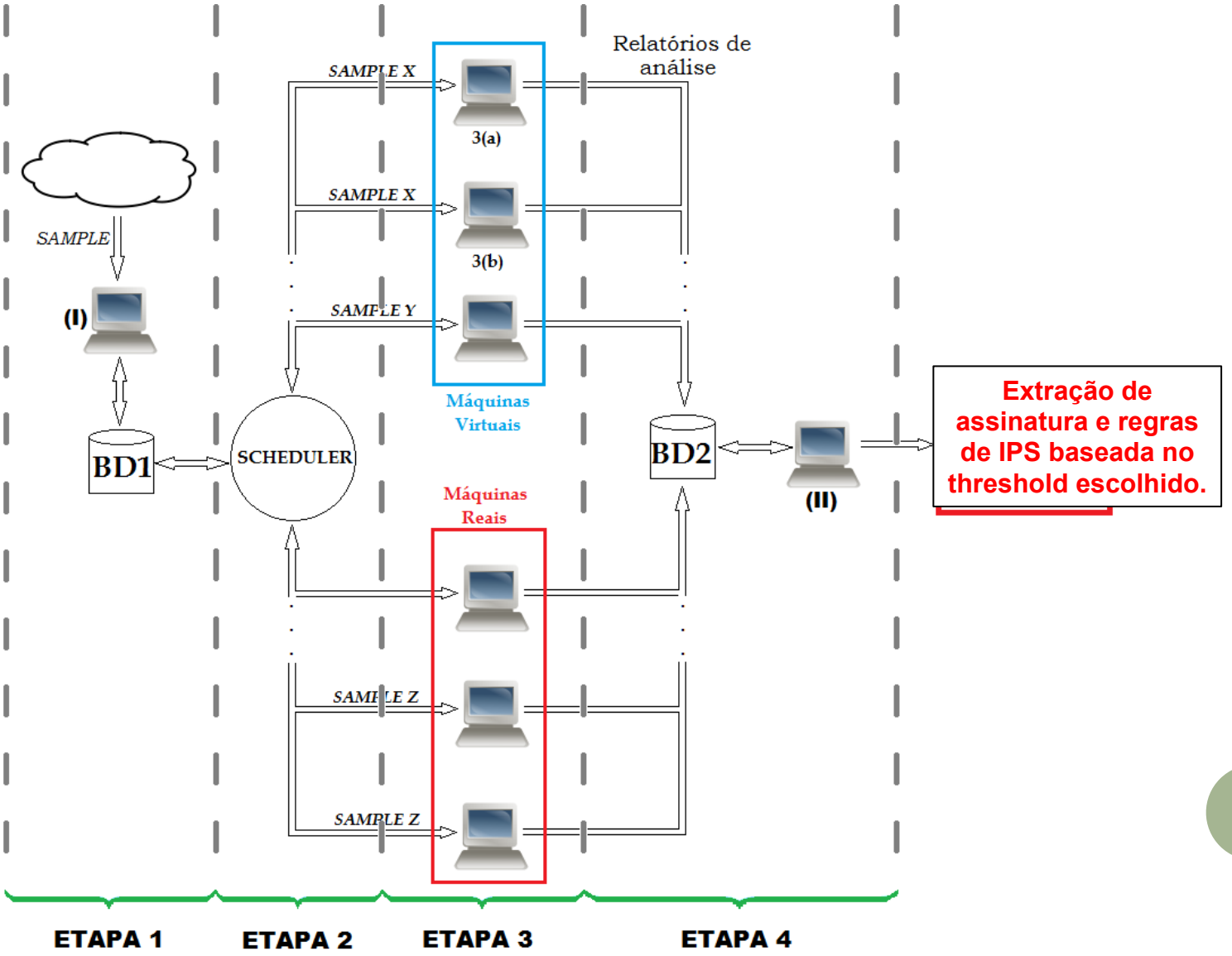
3- Proposta de Arquitetura e Protocolo

B. Protocolo

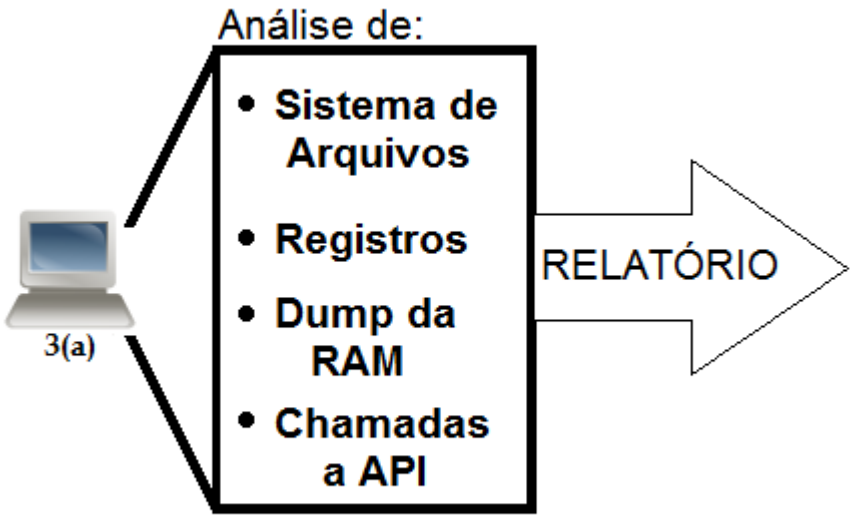


3- Proposta de Arquitetura e Protocolo

B. Protocolo



3- Proposta de Arquitetura e Protocolo
B. Protocolo



4- Sistema Operacional *Windows*

A. Sistema de Arquivos *NTFS*;

B. Registros do Sistema;

C. *Dump* da Memória RAM;

D. API do *Windows.*

4- Sistema Operacional *Windows*
A. Sistemas de Arquivos *NTFS*

- *Master File Table* (MFT) do Sistema NTFS (NTFS. . . , 2012);
- Armazenada no setor de boot do HD;
- Todo arquivo e diretório no HD possui entrada na MFT;

4- Sistema Operacional *Windows*
A. Sistemas de Arquivos *NTFS*

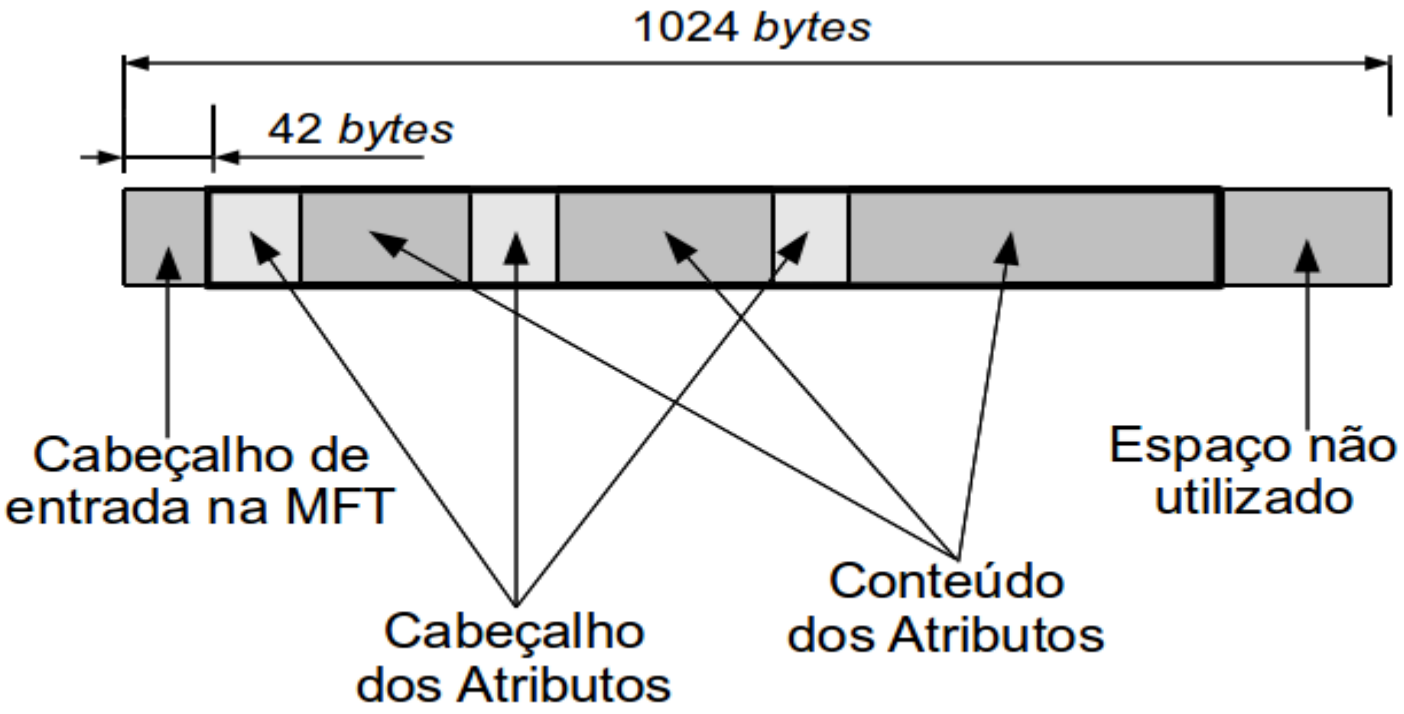


Figura 8: Estrutura da entrada na MFT.

4- Sistema Operacional *Windows*
B. Registros do Sistema

- Banco de dados hierárquico central (WINDOWS. . ., 2008);

- Contém toda informação necessária para configurar o sistema;

Tabela 1: Registros do sistema e seu conteúdo.

Chave/Hive	Informação
Legacy/System	Armazena quando um determinado arquivo foi executado e seu *timestamp*
Run/Software	Armazena quais arquivos irão ser executados sem a ciência do usuário (seja no *boot* ou quando alguma ação específica ocorrer).
Software Hive	Analisar as DLL's do sistema.

4- Sistema Operacional *Windows*
C. Dump da Memória RAM

- Arquivo binário que irá conter tudo que está alocado na memória RAM no momento da captura;

- Alguns *malwares* se alocam na RAM e se autodestroem do disco.

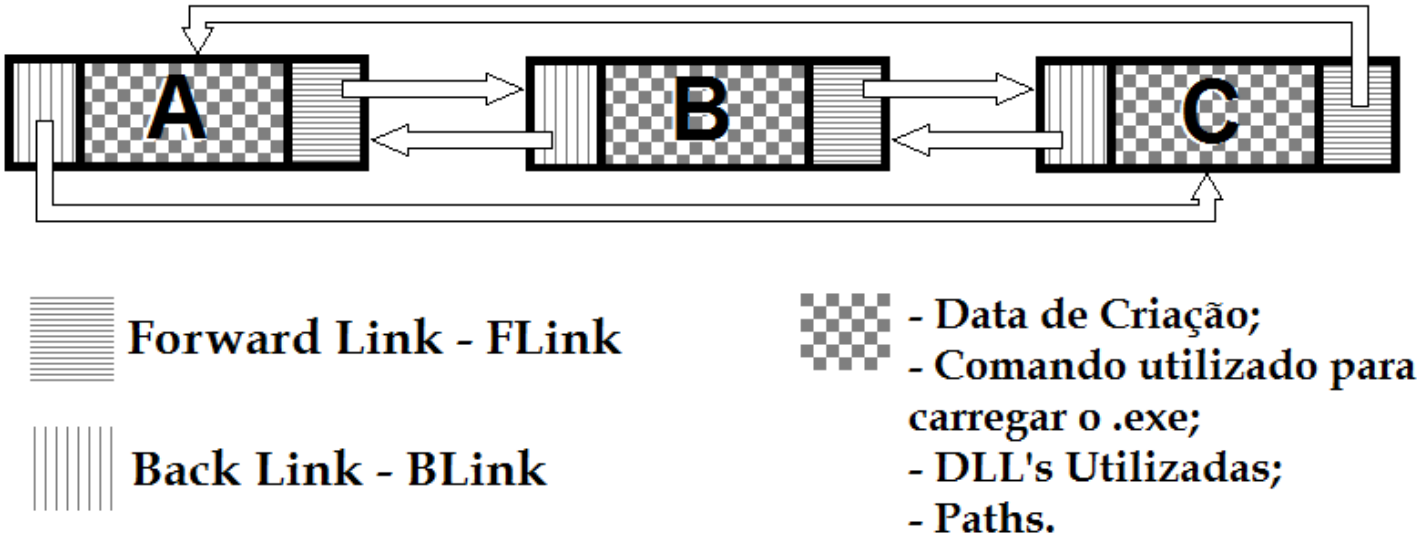


Figura 9: Disposição, realizada pelo *Windows*, dos *Eprocess* na memória.

4- Sistema Operacional *Windows*
C. Dump da Memória RAM

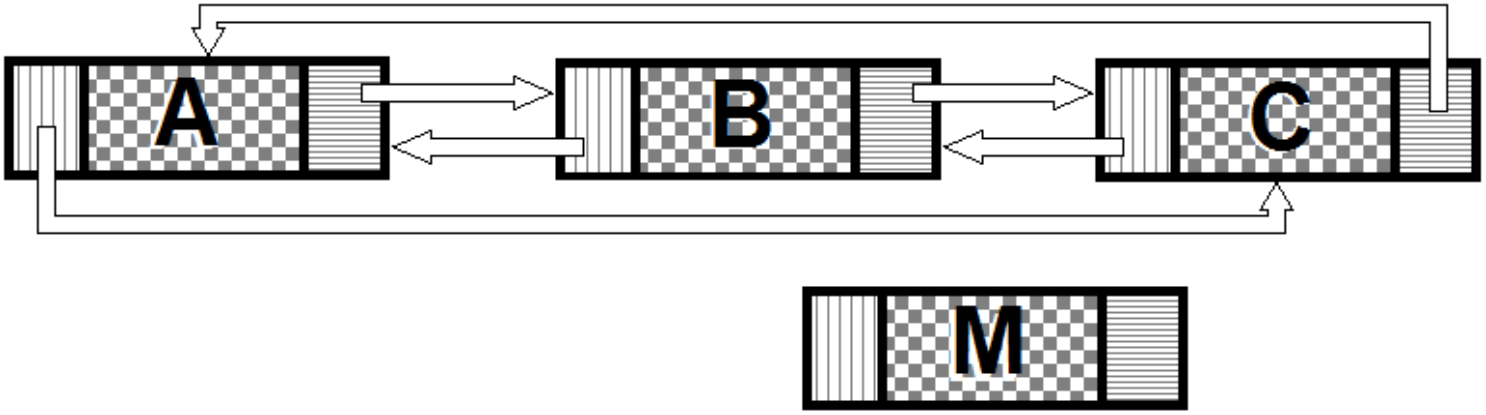


Figura 10: Bloco *Eprocess* pertencente a um *malware* fora da lista circular.

- Memória RAM é um recurso finito;
- No seu limite o *Windows* aloca os processos com menor uso em arquivos (*page file*) no HD (RAM. . . , 2012);

4- Sistema Operacional *Windows*
D. API do *Windows*

- Sistema de interface de programação a nível de usuário para a família de sistemas operacionais *Windows* (RUSSINOVICH; SOLOMON; IONESCU, 2012);

- Conjunto de funções que permitem que aplicações explorem os recursos que o *Windows* oferece (OVERVIEW. . . , 2010);

5- Conclusões e Trabalhos Futuros

A. Conclusões;

B. Trabalhos Futuros.

5- Conclusões e Trabalhos Futuros
A. Conclusões

- *Obfuscation* alteram o código mas mantém a sua funcionalidade (seu comportamento);
- *Anti-debugging, anti-disassembly* e *anti-VM* não interferem na análise comportamental;
- Vantagem na abordagem de extração da assinatura baseada em comportamento e em análise do código dos *malwares*;

5- Conclusões e Trabalhos Futuros
A. Conclusões

- A arquitetura proposta se destaca por possuir os seguintes diferenciais:
  - Criação da assinatura baseada de forma híbrida;
  - Quatro parâmetros de análise para se traçar o perfil comportamental do *malware*:
    - Análise do sistema de arquivos;
    - Análise de modificação dos registros do *Windows*;
    - Análise do *Dump* da RAM;
    - Análise das chamadas a API.

5- Conclusões e Trabalhos Futuros
A. Conclusões

- Assinatura capaz de identificar toda uma família de *malwares*;
- Extração de regras que podem ser implementadas em sistemas IPS;
- Capacidade de oferecer extração, tanto da assinatura quanto das regras para implementação em IPS baseada em *threshold*.

5- Conclusões e Trabalhos Futuros
A. Conclusões

- Limitações:

```
push    0               ; dwThreadId
push    0               ; lpModuleName
call    ds:GetModuleHandleA
push    eax             ; hmod
push    offset _main_function ; lpfn
push    WH_MOUSE_LL     ; idHook
call    ds:SetWindowsHookExA
```

```
thread_network_tasks proc near          ; DATA XREF: function_Launch+7F↓o
                push    ebx
                mov     ebx, eax
                push    300000          ; dwMilliseconds
                call    Sleep
                mov     eax, ebx
                call    DecryptCode___m

loop:                                   ; CODE XREF: thread_network_tasks+28↓j
                push    1200000         ; dwMilliseconds
                call    Sleep
                call    ModifyRegistry___m
                call    network_main
                jmp     short loop
thread_network_tasks endp
```

Figura 11: Trecho de código de um *Malware* onde exemplifica novas técnicas utilizadas pelos criadores para burlar sistemas automatizados.

5- Conclusões e Trabalhos Futuros
B. Trabalhos Futuros

- Implementação da arquitetura e protocolo;
- Inclusão do parâmetro “Fluxo de dados da rede”, para análise;
- Criação de um sistema de otimização a ser implementado nos bancos de dados.

Extra
Fluxo da Rede

- Dos Pacotes recolhidos são extraídos:
  - Endereço de IP (remetente e destinatário);
  - Porta TCP usada (remetente e destinatário);
  - Conteúdo do Pacote.

EXTRA
PROTOCOLO

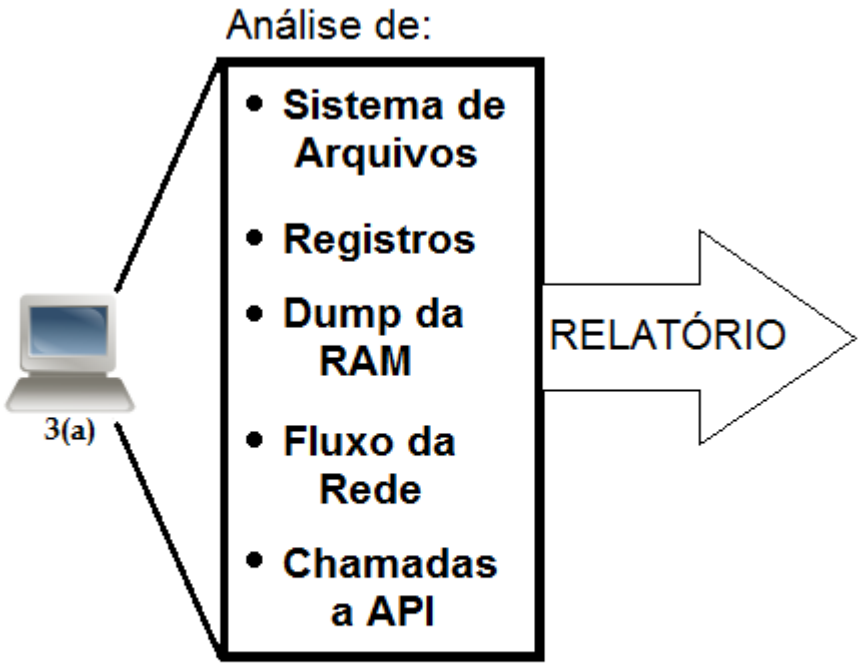


Figura 12: Análise do comportamento do *malware* baseada em 5 parâmetros.

Referencias Bibliográficas

- ALAZAB, M. et al. Malware detection based on structural and behavioural features of api calls. 2010.
- BRANCO, R. R.; BARBOSA, G. N.; NETO, P. D. Scientific but not academical overview of malware anti-debugging, anti-disassembly and anti-vm technologies. 2012.
- COMPUTER Threats FAQ. Kaspersky, 2012. Disponível em: <http://www.kaspersky.com/threats_faq#ddos>.
- CHRISTODORESCU, M.; JHA, S. Static analysis of executablesto detect malicious patterns. 2003. FERRIE, P. Anti-unpacker tricks. 2010.
- IT Threat Evolution: Q2 2012. SecuritList, 2012. Disponível em: <http://www.securelist.com/en/analysis/204792239/IT_Threat_Evolution_Q2_2012>.
- KAUSHAL, K.; SWADAS, P.; PRAJAPATI, N. Metamorphic malware detection using statistical analysis.IJSCE, v. 2, p. 49–53, 2012.

Referencias Bibliográficas

- KRUEGEL, C. et al. Polymorphic worm detection using structural information of executables. 2005.
- MATROSOV, A.; RODIONOV, E. Account of an investigation into a cybercrime group. 2012.
- NTFS File Attributes. NTFS.com, 2012. Disponível em: <http://www.ntfs.com/ntfs-files-types.htm>.
- OVERVIEW of the Windows API. Microsoft, 2010. Disponível em: <http://msdn.microsoft.com/pt-br/library/aa383723.aspx>.
- RAM, virtual memory, pagefile, and memory management in Windows. Microsoft, 2012. Disponível em:<http://support.microsoft.com/kb/2160852>.
- RUSSINOVICH, M. E.; SOLOMON, D. A.; IONESCU, A.Windows Internals, Part 1: Covering Windows Server 2008 R2 and Windows 7. 6th. ed. [S.l.]: Microsoft Press, 2012. 2–4 p.

Referencias Bibliográficas

- SOCIAL Engineering Yourself A BotNet. SocialEngineer.org, 2012. Disponível em: <http://www.social-engineer.org/social-engineering/social-engineering-yourself-a-botnet/>.

- STEVENS, K.The Underground Economy of the Pay-Per-Install (PPI) Business. SecureWorks Counter Threat Unit, 2010. Disponível em:<http://www.blackhat.com/presentations/bh-dc- 10/Stevens_Kevin/BlackHat-DC-2010-Stevens-Underground-wp.pdf>.

- WINDOWS registry information for advanced users. Microsoft Support, 2008. Disponível em: <http://support.microsoft.com/kb/256986>.

- YASON, M. V. The art of unpacking. 2007.

- YOU, I.; YIM, K. Malware obfuscation techniques: A brief survey. 2010.

DÚVIDAS?