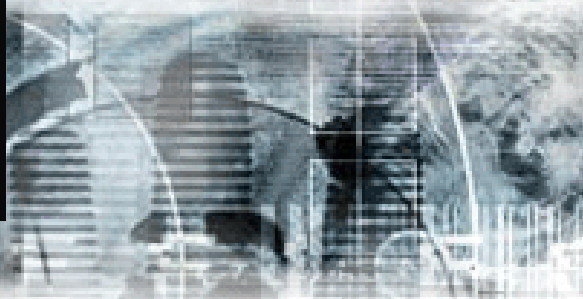




[www.h2hc.com.br](http://www.h2hc.com.br)



# Ataques Polimorficos

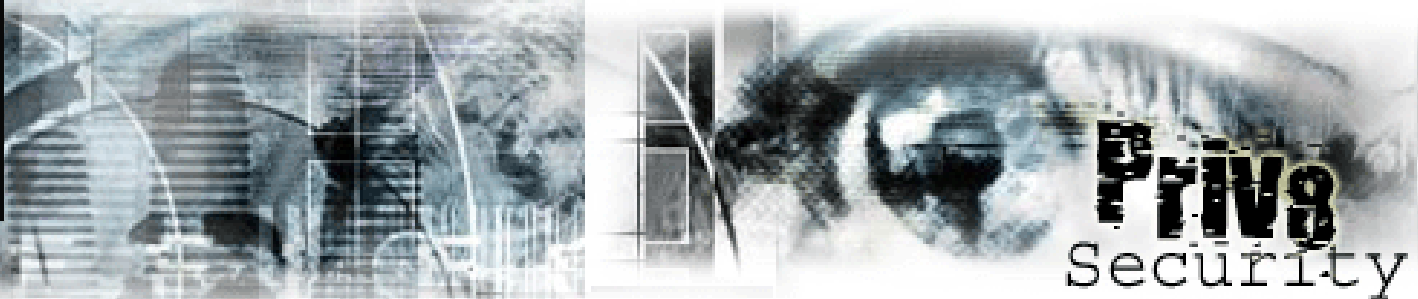
Rodrigo Rubira Branco

[rodrigo@firewalls.com.br](mailto:rodrigo@firewalls.com.br)

[bsddaemon@bsddaemon.org](mailto:bsddaemon@bsddaemon.org)



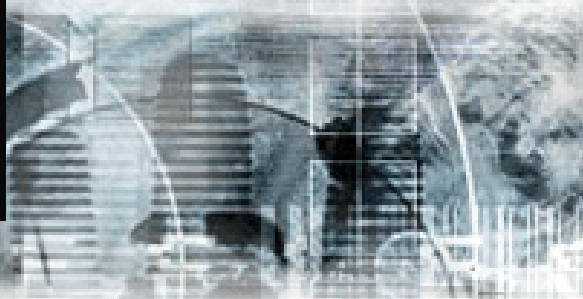
www.h2hc.com.br



- Resumo: Ha alguns anos, uma nova ideia foi inserida no mundo dos virus: Codigos polimorficos. Esta ideia consistia em codigos que possuiam condicoes de se auto-modificar, gerando um novo codigo quando era executado. Fazendo isto, era possivel passar por assinaturas de virus que possuiam simples pattner match. No mundo dos ataques (mais especificamente nos shellcodes) esta ideia voltou a mente apos artigo de Rix na phrack. Iremos verificar como isto funciona.



[www.h2hc.com.br](http://www.h2hc.com.br)



## DIFERENCIAIS

"Iremos demonstrar como os ataques polimorficos funcionam, e como um projeto de automatizacao torna isto ainda mais facil. Veremos um exemplo de codigo polimorfico passando por detectores de intrusos."



[www.h2hc.com.br](http://www.h2hc.com.br)

# Funcionamento

**Priv8**  
Security

-----  
**call decryptor**

-----  
**shellcode**

-----  
**decryptor**

-----  
**jmp shellcode**  
-----



[www.h2hc.com.br](http://www.h2hc.com.br)

# Funcionamento

**Priv8**  
Security

**O decoder fica responsavel por reverter o processo utilizado na codificacao do shellcode.**

**Este processo pode ser, por exemplo:**

- ADD**
- SUB**
- XOR**
- SHIFT**
- Encryption (like DES)**



[www.h2hc.com.br](http://www.h2hc.com.br)

# Decoder

**Priv8**  
Security

Quando efetuamos um call, o endereço da próxima atenção (no nosso caso, o endereço do nosso shellcode) será colocado na stack. Então, o decoder pode facilmente pegar este endereço do shellcode, fazendo um pop para um registrador qualquer. Então, tudo que sobra a fazer será manipular os bytes do shellcode e então dar um jump para este.



[www.h2hc.com.br](http://www.h2hc.com.br)

**Finalizando**

**Priv8**  
Security

**Precisamos apenas concatenar o shellcode no final de nosso decoder, modificando tambem o parametro utilizado para repetir o procedimento de decoder com o tamanho de nosso shellcode. Precisamos prever zero-bytes e outros caracteres especiais para evitar problemas.**



[www.h2hc.com.br](http://www.h2hc.com.br)

# Dificuldades

**Priv8**  
Security

- Evitar BADChars (\r \t \0)
- Gerar códigos alfanuméricos para os decoders
  - Eliminar constantes em decoders
  - Desenvolver novos decoders
- Manter decoders para múltiplas plataformas
  - Inserir instruções “do nothing”
  - Otimizar o tamanho do decoder





[www.h2hc.com.br](http://www.h2hc.com.br)

# SCMorphism

**Priv8**  
Security

A intenção não está apenas em codificar o shellcode original com um valor randômico e colocar o decoder na frente do mesmo. A ferramenta também pode utilizar um decoder randômicamente gerado.

SCMorphism possui os seguintes decoders: Xor/Add/Sub/Shuffle/Bit Shift {LEFT|Right}

Existe a possibilidade de se inserir quaisquer combinações destes decoders em double-pattern decoders (e mais recentemente, em N-pattern decoders)

Decoders são divididos em partes (e códigos do nothing são usados), seguindo-se a ideia da ferramenta s-poly de Zillion@safemode.org



www.h2hc.com.br

# SCMorphism

Priv8  
Security

## Usage:

- f <shellcode file> <var name> <shellcode length>--> If it is .c ord . pl specify VAR Name and shellcode length
- t <type: 0 add, 1 sub, 2 xor, 3 byte shift>. If not specified, using random values
- n <number>. Value used in the xor, add or sub operation
- e --> Execute the generated shellcode
- s --> Stdout form: -s <file name> to a file, if not gived use console
- b <badchar1,badchar2,badcharn> --> Exclude this char from shellcode (\n \r and \0 are always excluded)
- c <archfile>,<osfile>,<typefile>
- o --> Optimize option. Be careful, it removes "do nothing" instructions and can be detected by IDS (the decoder are static in this case)-m --> Open Menu to choose shellcode to use. Used if you dont have the shellcode
- v --> Version and greetz information
- h



[www.h2hc.com.br](http://www.h2hc.com.br)

# SCMorphism

**Priv8**  
Security

**Gracas as instrucoes do nothing e aos decoders randomicos, fica impossivel escrever assinaturas IDS para a ferramenta (ou teriamos milhares de falsos positivos).**

**Quando voce chama uma funcao de codificacao, esta ira utilizar um numero randomico para ser usado (que pode ser escolhido pelo usuario).**

**Badchars podem ser escolhidos pelo usuario (e a ferramenta ja implementa os mais importantes).**



[www.h2hc.com.br](http://www.h2hc.com.br)

# SCMorphism

**Priv8**  
Security

**A ferramenta possui diversas instrucoes do nothing e um algoritmo para escolher instrucoes validas para serem utilizadas.**

**Tambem possui integracao total com a ferramenta ADMMutate em termos de se utilizar da estrutura de NOPs oferecida (principalmente como parte de se utilizar a ferramenta para outras plataformas diferentes de intel)**



www.h2hc.com.br

# REFERENCIAS



<http://cs.southwesternadventist.edu/>

<http://rosec.org>

<http://m00.void.ru>

[http://www.infosecwriters.com/text\\_resources/pdf/basics\\_of\\_sh](http://www.infosecwriters.com/text_resources/pdf/basics_of_sh)

<http://www.securityfocus.com/infocus/1768>

<http://www.priv8security.com>

<http://www.intel.com/design/Pentium4/documentation.htm#pag>

<http://www.phrack.org>

<http://www.bsdaemon.org>

<http://www.firewalls.com.br>



[www.h2hc.com.br](http://www.h2hc.com.br)

**FIM! Sera mesmo?**

**Priv8**  
Security

**Duvidas?!?**

**Rodrigo Rubira Branco**

**[rodrigo@firewalls.com.br](mailto:rodrigo@firewalls.com.br)**

**[bsddaemon@bsddaemon.org](mailto:bsddaemon@bsddaemon.org)**