

ReverseCraft

by gynvael.coldwind//vx

007 - Inline hooks, DLL injection

Strony projektu:

<http://re.coldwind.pl/>
<http://www.uw-team.org/>

Spis treści

Co to są inline hooki?

Jak działają inline hooki?

Po co się używa inline hooków?

Jak w praktyce (manualnie) użyć inline hooków?

Co to jest DLL injection i jak tego używać?

Podsumowanie

***„What would the world be like
without Captain Hook?”***

Kapitan James Hook

Hook

**eng. hook - hak, haczyk, punkt zaczepienia,
obejście**

**eng. to hook - przyczepiać, przymocowywać,
buchnąć, zwinąć, rąbnać, ukraść**

inline hook - jedna z technik hookowania funkcji, polegająca na nadpisaniu początkowego fragmentu funkcji instrukcją skoku

Ale... po co?

- logowanie wywołań funkcji, w celu zrozumienia jak aplikacja działa, wykrycia pewnych wzorców zachowania, lub w innych celach (excp-hook)
- zmiana parametrów wywołania oraz wartości zwracanych przez funkcje, w celu wpłynięcia na przebieg wykonania programu (ansihack)
- emulowanie środowiska (bundlery, hc1)

Jak działają inline hooki?

```
...  
push param2  
push param1  
call func  
add esp, 8  
...
```

```
func:  
push ebp  
mov ebp, esp  
sub esp, 24  
...  
leave  
ret
```

Chcemy móc wpłynąć
na funkcję 'func' i
poznać jej parametry
przy wywołaniach!

Jak działają inline hooki?

```
...  
push param2  
push param1  
call func  
add esp, 8  
...
```

```
func:  
push ebp  
mov ebp, esp  
sub esp, 24  
...  
leave  
ret
```

hook_func

- logowanie parametrów
- zmiana parametrów
- zastąpienie oryginalnej funkcji
- inne

Jak działają inline hooki?

```
...  
push param2  
push param1  
call func  
add esp, 8  
...
```

func:

```
jmp hook_func
```

```
sub esp, 24
```

```
...  
leave  
ret
```

hook_func

- logowanie parametrów
- zmiana parametrów
- zastąpienie oryginalnej funkcji
- inne

Jak działają inline hooki?

```
...  
push param2  
push param1  
call func  
add esp, 8  
...
```

```
func:
```

```
jmp hook_func
```

```
sub esp, 24
```

```
...
```

```
leave
```

```
ret
```

```
hook_func
```

- logowanie parametrów
- zmiana parametrów
- zastąpienie oryginalnej funkcji
- inne

Jak działają inline hooki?

```
...  
push param2  
push param1  
call func  
add esp, 8  
...
```

func:

```
jmp hook_func
```

```
sub esp, 24
```

```
...  
leave  
ret
```

hook_func

- logowanie parametrów
- zmiana parametrów
- zastąpienie oryginalnej funkcji
- inne

hook_func_ret

- logowanie wartości zwracanych
- zmiana wartości zwracanych
- inne

Jak działają inline hooki?

```
...  
push param2  
push param1  
call func  
add esp, 8  
...
```

func:

```
jmp hook_func
```

```
sub esp, 24
```

```
...  
leave  
ret
```

hook_func

- logowanie parametrów
- zmiana parametrów
- zastąpienie oryginalnej funkcji
- zmiana adresu ret

hook_func_ret

- logowanie wartości zwracanych
- zmiana wartości zwracanych
- inne

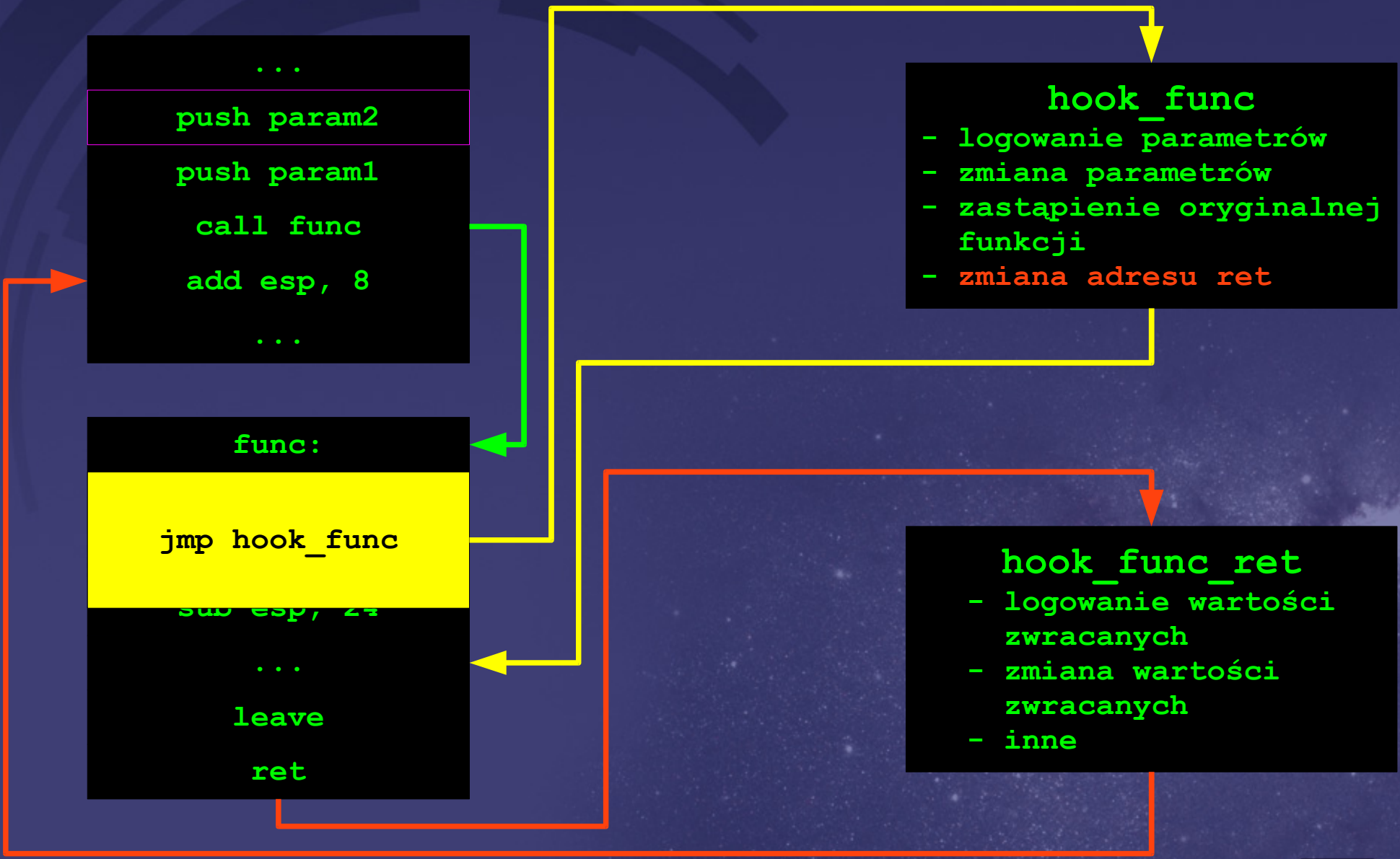
Jak działają inline hooki?

```
...  
push param2  
push param1  
call func  
add esp, 8  
...
```

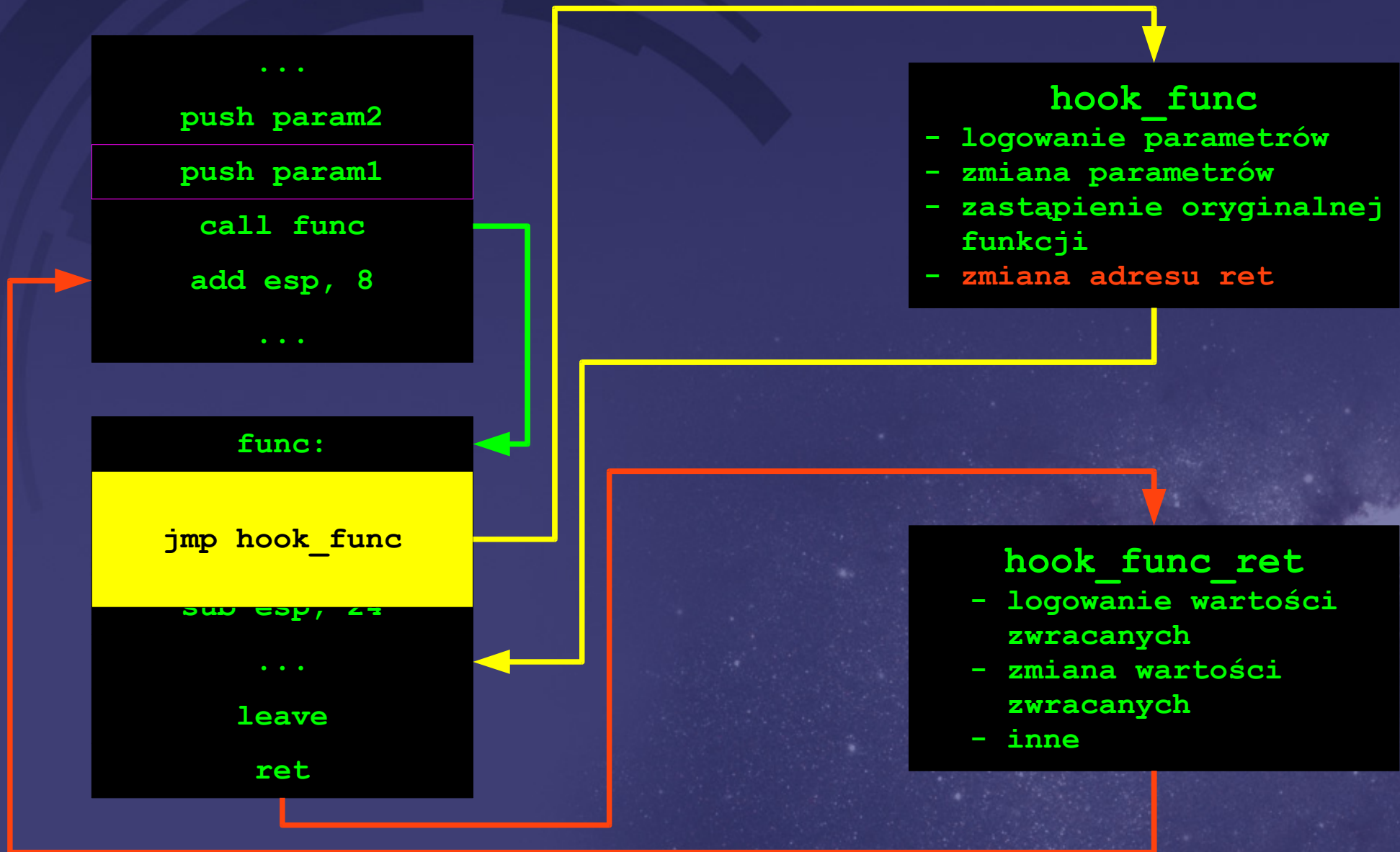
```
func:  
jmp hook_func  
sub esp, 24  
...  
leave  
ret
```

```
hook_func  
- logowanie parametrów  
- zmiana parametrów  
- zastąpienie oryginalnej funkcji  
- zmiana adresu ret
```

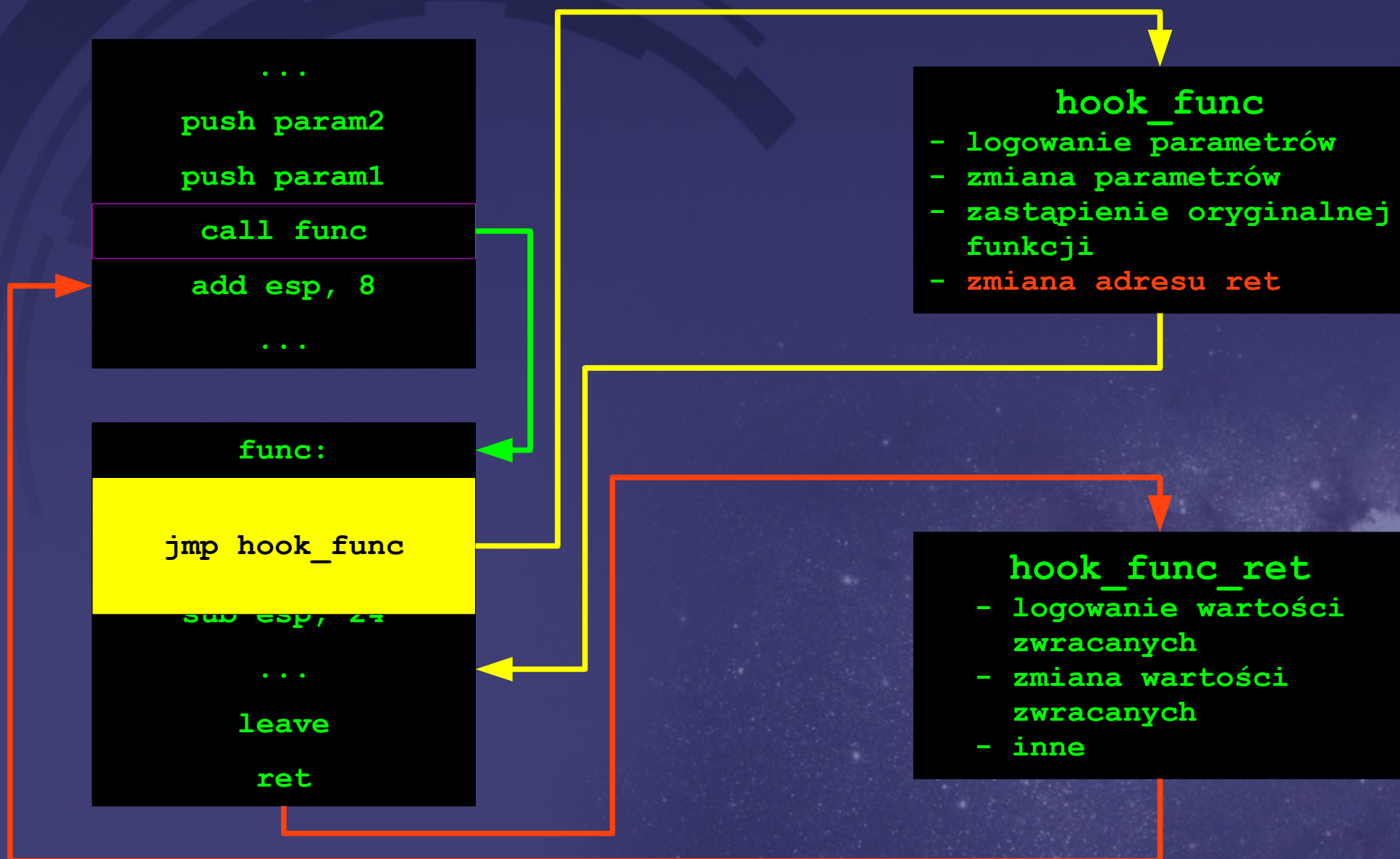
```
hook_func_ret  
- logowanie wartości zwracanych  
- zmiana wartości zwracanych  
- inne
```



Jak działają inline hooki?



Jak działają inline hooki?



Jak działają inline hooki?

```
...  
push param2  
push param1  
call func  
add esp, 8  
...
```

func:

```
jmp hook_func
```

```
sub esp, 24
```

```
...  
leave  
ret
```

hook_func

- logowanie parametrów
- zmiana parametrów
- zastąpienie oryginalnej funkcji
- zmiana adresu ret

hook_func_ret

- logowanie wartości zwracanych
- zmiana wartości zwracanych
- inne

Jak działają inline hooki?

```
...  
push param2  
push param1  
call func  
add esp, 8  
...
```

func:

```
jmp hook_func
```

```
sub esp, 24
```

```
...  
leave
```

```
ret
```

hook_func

- logowanie parametrów
- zmiana parametrów
- zastąpienie oryginalnej funkcji
- zmiana adresu ret

hook_func_ret

- logowanie wartości zwracanych
- zmiana wartości zwracanych
- inne

Jak działają inline hooki?

```
...  
push param2  
push param1  
call func  
add esp, 8  
...
```

func:

```
jmp hook_func
```

```
sub esp, 24
```

```
...
```

```
leave
```

```
ret
```

hook_func

- logowanie parametrów
- zmiana parametrów
- zastąpienie oryginalnej funkcji
- zmiana adresu ret

hook_func_ret

- logowanie wartości zwracanych
- zmiana wartości zwracanych
- inne

Jak działają inline hooki?

```
...  
push param2  
push param1  
call func  
add esp, 8  
...
```

func:

```
jmp hook_func
```

```
sub esp, 24
```

```
...
```

```
leave
```

```
ret
```

hook_func

- logowanie parametrów
- zmiana parametrów
- zastąpienie oryginalnej funkcji
- zmiana adresu ret

hook_func_ret

- logowanie wartości zwracanych
- zmiana wartości zwracanych
- inne

Jak działają inline hooki?

```
...  
push param2  
push param1  
call func  
add esp, 8  
...
```

func:

```
jmp hook_func
```

```
sub esp, 24
```

```
...
```

```
leave
```

```
ret
```

hook_func

- logowanie parametrów
- zmiana parametrów
- zastąpienie oryginalnej funkcji
- zmiana adresu ret

hook_func_ret

- logowanie wartości zwracanych
- zmiana wartości zwracanych
- inne

Jak działają inline hooki?

```
...  
push param2  
push param1  
call func  
add esp, 8  
...
```

func:

```
jmp hook_func
```

```
sub esp, 24
```

```
...  
leave  
ret
```

hook_func

- logowanie parametrów
- zmiana parametrów
- zastąpienie oryginalnej funkcji
- zmiana adresu ret

hook_func_ret

- logowanie wartości zwracanych
- zmiana wartości zwracanych
- inne

Jak działają inline hooki?

```
...  
push param2  
push param1  
call func  
add esp, 8  
...
```

```
func:  
jmp hook_func  
sub esp, 24  
...  
leave  
ret
```

hook_func

- logowanie parametrów
- zmiana parametrów
- zastąpienie oryginalnej funkcji
- zmiana adresu ret

hook_func_ret

- logowanie wartości zwracanych
- zmiana wartości zwracanych
- inne

Jak działają inline hooki?

```
...  
push param2  
push param1  
call func  
add esp, 8  
...
```

func:

```
jmp hook_func
```

```
sub esp, 24
```

```
...  
leave  
ret
```

hook_func

- logowanie parametrów
- zmiana parametrów
- zastąpienie oryginalnej funkcji
- zmiana adresu ret

hook_func_ret

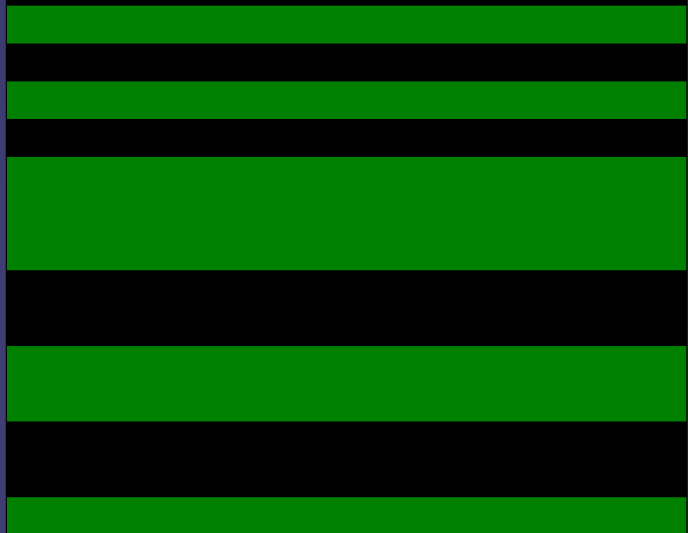
- logowanie wartości zwracanych
- zmiana wartości zwracanych
- inne

DLL Injection

Proces A

Thready: 0, 1, 2
DLL: msvcrt.dll
kernel32.dll
ntdll.dll

Pamięć procesu:



Proces B

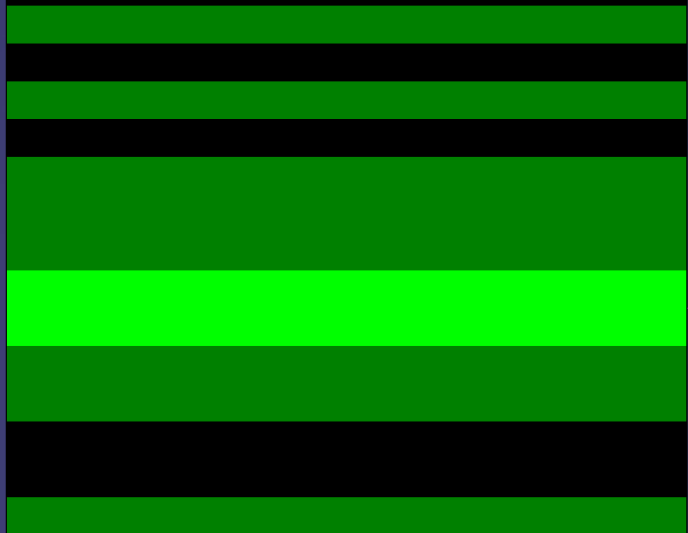
VirtualAllocEx
WriteProcessMemory
CreateRemoteThread
(addr LoadLibrary,
addr DLLName)

DLL Injection

Proces A

Thready: 0, 1, 2
DLL: msvcrt.dll
kernel32.dll
ntdll.dll

Pamięć procesu:



Proces B

VirtualAllocEx
WriteProcessMemory
CreateRemoteThread
(addr LoadLibrary,
addr DLLName)



DLL Injection

Proces A

Thready: 0, 1, 2
DLL: msvcrt.dll
kernel32.dll
ntdll.dll

Pamięć procesu:

mylib.dll\0

Proces B

VirtualAllocEx

WriteProcessMemory

CreateRemoteThread
(addr LoadLibrary,
addr DLLName)



DLL Injection

Proces A

Thready: 0, 1, 2, 3
DLL: msvcrt.dll
kernel32.dll
ntdll.dll

Pamięć procesu:

EIP -> LoadLibrary()

mylib.dll\0

Proces B

VirtualAllocEx
WriteProcessMemory
CreateRemoteThread
(addr LoadLibrary,
addr DLLName)



DLL Injection

Proces A

Thready: 0, 1, 2, 3

DLL: msvcrt.dll
kernel32.dll
ntdll.dll
mylib.dll

Pamięć procesu:

EIP -> LoadLibrary()

mylib.dll\0

MYLIB.DLL .text .data ...

Proces B

VirtualAllocEx
WriteProcessMemory
CreateRemoteThread
(addr LoadLibrary,
addr DLLName)

DLL Injection

Proces A

Thready: 0, 1, 2, 3

DLL: msvcrt.dll

kernel32.dll

ntdll.dll

mylib.dll

Pamięć procesu:

mylib.dll\0

EIP -> MYLIB.DLL.DllMain()

Proces B

VirtualAllocEx

WriteProcessMemory

CreateRemoteThread

(addr LoadLibrary,
addr DLLName)

Inline hook w praktyce (wersja manualna)

Zadanie:

Aplikacja X wypisuje na stdout pewien tekst. Tekst zawiera m.in. tag [WHAT].

Stwórz w kontekście aplikacji X inline hook na funkcje **msvcrt.puts**. Hook handler powinien zamieniać w locie [WHAT] na słowo „World”.

Hint:

użyj techniki DLL injection aby dostać się w kontekst procesu
użyj funkcji GetProcAddress aby ustalić adres funkcji puts

Podsumowanie

Inline hooks

1. metoda wysoce inwazyjna, ale i wysoce pewna
2. niewielka elastyczność (w wersji manual)
3. środowisko musi pozostać nienaruszone
4. parameter shader, return shader, replace func

DLL Injection

1. bardzo prosta metoda wprowadzenia kodu w app
2. pozwala na napisanie większości kodu w HLL
3. bardzo głośna metoda, zależna od LoadLibrary*
4. nie zawsze działa przy CreateProcess z SUSPENDED



Dziękuję za uwagę :)

Strony projektu:

<http://re.coldwind.pl/>
<http://www.uw-team.org/>