KERNEL EXPLOITATION
Background for chall8
The good news

• It’s an easy and fun challenge!

• Today, small intro (reminder) about necessary concepts

• You will find a well guided lab online
  – With an incomplete exploit
  – A Vagrant machine configuration (Read the ReadMe!)

• Once you made it work on your (Vagrant) machine
What is a Kernel after all

- A Kernel is code that:
  - Is privileged
  - Executes because interrupts or system calls
  - Isolates processes from each other
  - Handles hardware and provides services

- Rather large and complex code
- Rather protected from userland
- Mandatory for the system, easy to “panic”
Under Linux (until early 2018...):
user address space / kernel address space in the same address space

<table>
<thead>
<tr>
<th>Kernel space</th>
<th>0xFFFFFFFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>User space</td>
<td>0xC0000000</td>
</tr>
<tr>
<td></td>
<td>0x80000000</td>
</tr>
<tr>
<td></td>
<td>0x00000000</td>
</tr>
</tbody>
</table>
Null Pointer Dereference

- **PAX UDREF**: forbid dereference of user space addresses
  - Uses Segmentation, somehow deprecated

- **Mainline Linux**: Current solution is to prevent memory mapping below a threshold address
  - Dereferencing any pointer in user space?
  - How to detect dereference of other invalid pointers?
Under Linux (until early 2018...):
user address space / kernel address space in the same address space

Kernel code has a bug and accesses user space
Under Linux (until early 2018...):
user address space / kernel address space in the same address space

Kernel space

0xFFFFFFFFF

Here code will execute with Kernel privilege but in user pages

• Access to null page was in fact accessing a function pointer
• This function pointer was placed there by the attacker and points to the exploit code which then gets executed

User space

0xC0000000

Exploit code

Fct\^ ptr

0x80000000

0x00000000
Linux Kernel Specifics

Under Linux (until early 2018...):
user address space / kernel address space in the same address space

Kernel space

User space

Exploit code needs to properly return to kernel after doing $malicious_stuff then only the system call can terminate
Null Pointer Dereference

• Intel introduced a HW mechanism to prevent kernel mode to access user mode data:
  – SMEP (Supervisor Mode Execution Protection Enable)
  – SMAP (Supervisor Mode Access Protection Enable)

• See:
  – http://vulnfactory.org/blog/2011/06/05/smep-what-is-it-and-how-to-beat-it-on-linux/
  – https://forums.grsecurity.net/viewtopic.php?f=7&t=3046