Alfredo Pesoli
LaCon 2008

OS X Rootkits: The next level

OS X Rootkits - iCal

- Once upon a time
- XNU Hacking
  - KSpace Hooking
    - Mach vs. BSD
  - Process Infection
    - Thank you very Mach
- High-Level Hooking
  - Bundle Injection in Cocoa Apps

OS X Rootkits - Once upon a time

- WeaponX (KSpace rootkit)
  - First syscall rerouting implementation of a kernel rootkit
- Inqtana
  - Spreading -> CVE-2005-1333 Apple Mac OS X Bluetooth Directory Traversal
  - Launchd used as the loading point
- Leap.A
  - First _virus_ in the wild()
  - Uses Input Manager

OS X Rootkits - Once upon a time

- Process Infection
  - task_for_pid() is a function used for obtaining a communication port for a given process (IPC)
    - used for obtaining a task_port_t object
  - The port object then is used for IPC by the Mach Subsystem:
    - vmwrite, vmalloc, vmfree …
  - No checks over uid/gid->Infection()

OS X Rootkits - Leopard, what now?

- sysent not exported anymore by the kernel (from 10.4.x Tiger)
  - But still present for obvious reasons in the running kernel (ssdt-like struct)
  - not-write-protected (not really obvious…)

- Tunable kernel parameter implemented as a check for the task_for_pid() call

```
#define KERN_TFP_POLICY_DENY            0  /* Priv */
#define KERN_TFP_POLICY                 1  /* Not used */
#define KERN_TFP_POLICY_DEFAULT         2 /* Related */
```

OS X Rootkits - BSD Basic Knowledge

bsd/sys/sysent.h

```
struct sysent {
    int16_t         sy_narg;
    int8_t          sy_resv;
    int8_t          sy_flags;
    sy_call_t       *sy_call;
    sy_munge_t      *sy_arg_munge32;
    sy_munge_t      *sy_arg_munge64;
    int32_t         sy_return_type;
    uint16_t        sy_arg_bytes;
};
```

- **sysent** is an SSDT-like struct which contains all the *bsd syscall*

OS X Rootkits - BSD Basic Knowledge

bsd/sys/sysent.h

```
struct sysent {
    int16_t         sy_narg;
    int8_t          sy_resv;
    int8_t          sy_flags;
    sy_call_t       *sy_call;
    sy_munge_t      *sy_arg_munge32;
    sy_munge_t      *sy_arg_munge64;
    int32_t         sy_return_type;
    uint16_t        sy_arg_bytes;
};
```

- sysent is an SSDT-like struct which contains all the bsd syscall
- ***sy_call** is the variable that contains the function pointer for the given call

OS X Rootkits - BSD Basic Knowledge

osfmk/kern/syscall_sw.h

```
typedef struct {
    int             mach_trap_arg_count;
    int             (*mach_trap_function)(void);
#if defined(__i386__)
    boolean_t       mach_trap_stack;
#else
    mach_munge_t
  *mach_trap_arg_munge32;
    mach_munge_t
  *mach_trap_arg_munge64;
#endif
#if !MACH_ASSERT
    int             mach_trap_unused;
#else
    const char      *mach_trap_name;
#endif
} mach_trap_t;

extern mach_trap_t   mach_trap_table[];
```

- For the *mach syscalls* instead there's the **mach_trap_table**

OS X Rootkits - BSD Basic Knowledge

osfmk/kern/syscall_sw.h

```
typedef struct {
    int                 mach_trap_arg_count;
    int                 (*mach_trap_function)(void);
#if defined(__i386__)
    boolean_t   mach_trap_stack;
#else
    mach_munge_t    *mach_trap_arg_munge32;
    mach_munge_t    *mach_trap_arg_munge64;
#endif
#if  !MACH_ASSERT
    int                 mach_trap_unused;
#else
    const char          *mach_trap_name;
#endif
} mach_trap_t;

extern mach_trap_t   mach_trap_table[];
```

- For the *mach syscalls* instead there's the **mach_trap_table**
- ***mach_trap_function** contains the function pointer for the given call

OS X Rootkits - BSD sysent hooking

~/xnu-1228.3.13/bsd/kern/init_sysent.c

```
__private_extern__ struct sysent sysent[] = {
        {0, 0, 0, (sy_call_t *)nosys, NULL, NULL, _SYSCALL_RET_INT_T, 0}
        {AC(exit_args), 0, 0, (sy_call_t *)exit, munge_w, munge_d, _SYSCALL_RET_NONE, 4}
        {0, 0, 0, (sy_call_t  *)fork, NULL, NULL, _SYSCALL_RET_INT_T, 0},
```

- The first entry is the nosys syscall, the second one is exit, the third is fork
  - *nm /mach_kernel | egrep "_nosys|_exit|_fork"*

```
00389b48 T _nosys
0037027b T _exit
00371dd5 T _fork
```

OS X Rootkits - BSD sysent hooking

- otool –d /mach_kernel | grep “48 9b 38”

```
00504780  ab 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00504790  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005047a0  00 00 00 00 48 9b 38 00 00 00 00 00 00 00 00 00
005047b0  01 00 00 00 00 00 00 00 01 00 00 00 7b 02 37 00
005047c0  80 d0 3d 00 00 00 00 00 00 00 00 00 04 00 00 00
005047d0  00 00 00 00 d5 1d 37 00 00 00 00 00 00 00 00 00
```

OS X Rootkits - BSD sysent hooking

- Now we need an exported symbol in order to obtain a fixed VA
  - Hopefully not far-far-away and reliable (with a fixed offset far from the sysent struct)

- nm /mach_kernel | grep 504780

  00504780 _nsysent <- Number of syscalls

- grep –ir ~/kern/1228.3.13/bsd/ “nsysent”

  sys/sysent.h:extern int nsysent;

- W00t!

OS X Rootkits - BSD sysent hooking

- How to find the sysent struct

```
    struct sysent *table;

    table_size = sizeof(struct sysent) * nsysent;
    table = (struct sysent *) ( ((char *) &nsysent) + sizeof(nsysent) );

#if __i386__
    /*
     * 28 bytes padding for i386
     */
    table = (struct sysent *) ( ((uint8_t *) table) + 28);
#endif
```

OS X Rootkits - BSD sysent hooking

- In case nsysent would not be exported anymore
  - Bruteforcing
- It's very simple to find a static pattern to match on the running kernel
  - E.g. sequences of syscall args
- As long as there will be one single export it's ok

OS X Rootkits - Low-level Injection Map

- Thread Injection
  - task_for_pid() (OpenProcess)
  - vm_allocate() (VirtualAlloc)
  - vm_write() (WriteProcessMemory)
  - thread_create_running (CreateRemoteThread)

OS X Rootkits - Process Infection

- What happen now is that we have some problems to deal with while infecting in-memory processes
  - Problem #1: Complete control over the target application
  - Problem #2: A single reboot can delete the infection.
  - Problem #3: Silent Mode please
  - Anything else ?

OS X Rootkits - Process Infection

- Function Overriding / Detour
  - Hooking performed by interposing the malicious code between the function call and the original implementation
    - CALL -> Malicious_Funct() -> Original_Funct()
  - Good old Inline hooking
    - Replace the first bytes of the original function with a relative JMP
  - Reliability ? Escape Branch Island
    - Stability and execution flow correctly restored
    - We will copy inside the Branch Island the original bytes of the function that we patched in order to restore them back later

OS X Rootkits - Hooking Map

- Function Overriding
  - _dyld_lookup_and_bind() (GetProcAddress)
  - _dyld_lookup_and_bind_with_hint(lib_name) (GetProcAddress)
  - vm_protect(page) (VirtualIProtect)
  - vm_allocate() (VirtualIAlloc)
  - MakeDataExecutable/msync (VirtualIProtect)
  - Patching Istructions (WriteProcessMemory)

OS X Rootkits - High-Level Hooking

- Input Manager
  - **“An input manager (NSInputManager object) serves as a proxy for a particular input server and passes messages to the active input server”**
  - Officially they’re plugins used by Apple for extending the Input Languages Methods inside all the Cocoa Applications (aka localization)

OS X Rootkits - High-Level Hooking

- Input Manager
  - Injecting Arbitrary Code in everything [Hacking Cocoa]
  - /Library/InputManagers
  - **Every single** application will load our code
  - The bundle itself can decide about which application he wants to attach to

An NSBundle object represents a location in the file system that groups code and resources that can be used in a program

NSBundle bundle = [NSBundle bundleWithPath:[_plugin path]];*

OS X Rootkits - High-level “stuff”

- **plist** -- property list format
- *defaults write /Library/Preferences/ com.apple.loginwindow HiddenUsersList –array-add “user”*
- *defaults write /Library/Preferences/ com.apple.SystemLoginItems AutoLaunchedApplicationDictionary -array-add '<dict><key>Hide</key><true/><key>Path</ key><string>app_path</string></dict>’*

OS X Rootkits - Process Infection

- Tell app “Finder” to get name of first window/file in first window
- Tell app “mail” to get name of every account
- Tell app “ARDAgent” to do shell script “kextload pwned.kext”
  - Now patched

OS X Rootkits – References

- Fixing ptrace(pt_deny_attach,…) on Mac OS X 10.5 Leopard (Landon Fuller)
  - http://landonf.bikemonkey.org/code/macosx/Leopard_PT_DENY_ATTACH.20080122.html
- Dinamically overriding Mac OS X (rentzsch)
  - http://rentzsch.com/papers/overridingMacOSX
- Abusing Mach on Mac OS X (Nemo)
  - http://www.uninformed.org/?v=4&a=3&t=txt
- weaponX (Nemo)
- Mac OS X wars – a XNU Hope
  - http://phrack.org/issues.html?issue=64&id=11#article
- Smart InputManager Bundle Loader
  - http://www.culater.net/software/SIMBL/SIMBL.php

QUESTIONS ?

Alfredo Pesoli
<revenge@0xcafebabe.it>
www.0xcafebabe.it

THANK YOU!