Parasite OS

Disclamer

All content shown here expresses my personal opinions and not of my current employer, Microsoft. Parasite OS is a personal pet project and is not related to any of my work.

Redmond, October, 2017.

Parasite OS

Prologue

Are you safe?

- Firewall
- Anti-virus
- Intrusion Detection
- Patches
- **Processes**

Quiz: are you scared of being hacked?

A. Yes

B. Better not know about it

C. Have processes and technology in place to mitigate the risks

D. Will get hacked anyway, please don’t hurt me

It might not be for you, but…

- Companies will get hacked;
- People will get hacked;
- Whatever the value of information has…
- … whoever is interested in it, with a motive, will get it;
- And everybody will be in denial.

Quiz: defense vs offense, who wins?

A. Defense

B. Offense

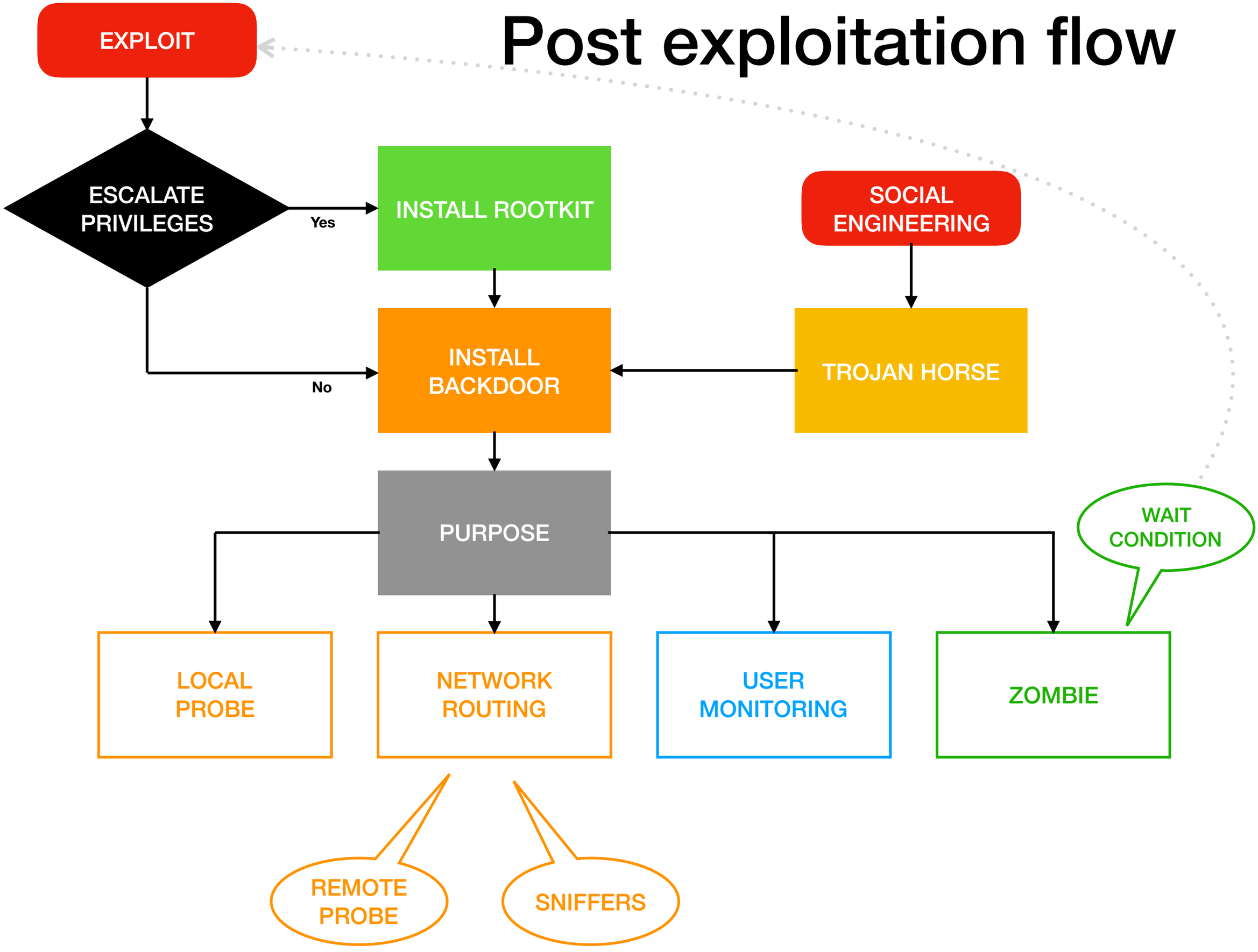
C. The industry

D. Nobody

Parasite OS

Gustavo Scotti

Post exploitation flow



Activity is monitored

- Most (if not all) malicious code runs over the host Operating System;
- Operating System can be traced and malicious code detected;
- Example:
  - CreateProcess from java.exe is a bad sign;
  - /bin/sh from *named* is also not good.

Assumptions

- Code is executed by an user (exploitation is irrelevant);
- General users are flawed (easily coerced to run code);
- Mitigation/Heuristic/A.I. tries to catch post-exploitation (limit over infinite time is perfect detection);
- Professionally crafted backdoors / trojan horses can be installed.

What is Parasite?

- It is a pseudo Operating System running on top of the host Operating System;
- Provides System Calls for process, network, file, and memory management;
- It is small kernel code footprint (~100KiB);
- Universal binaries for any OS

Why?

- Persistency is hard - most of the time processes hidden by unreliable rootkits;

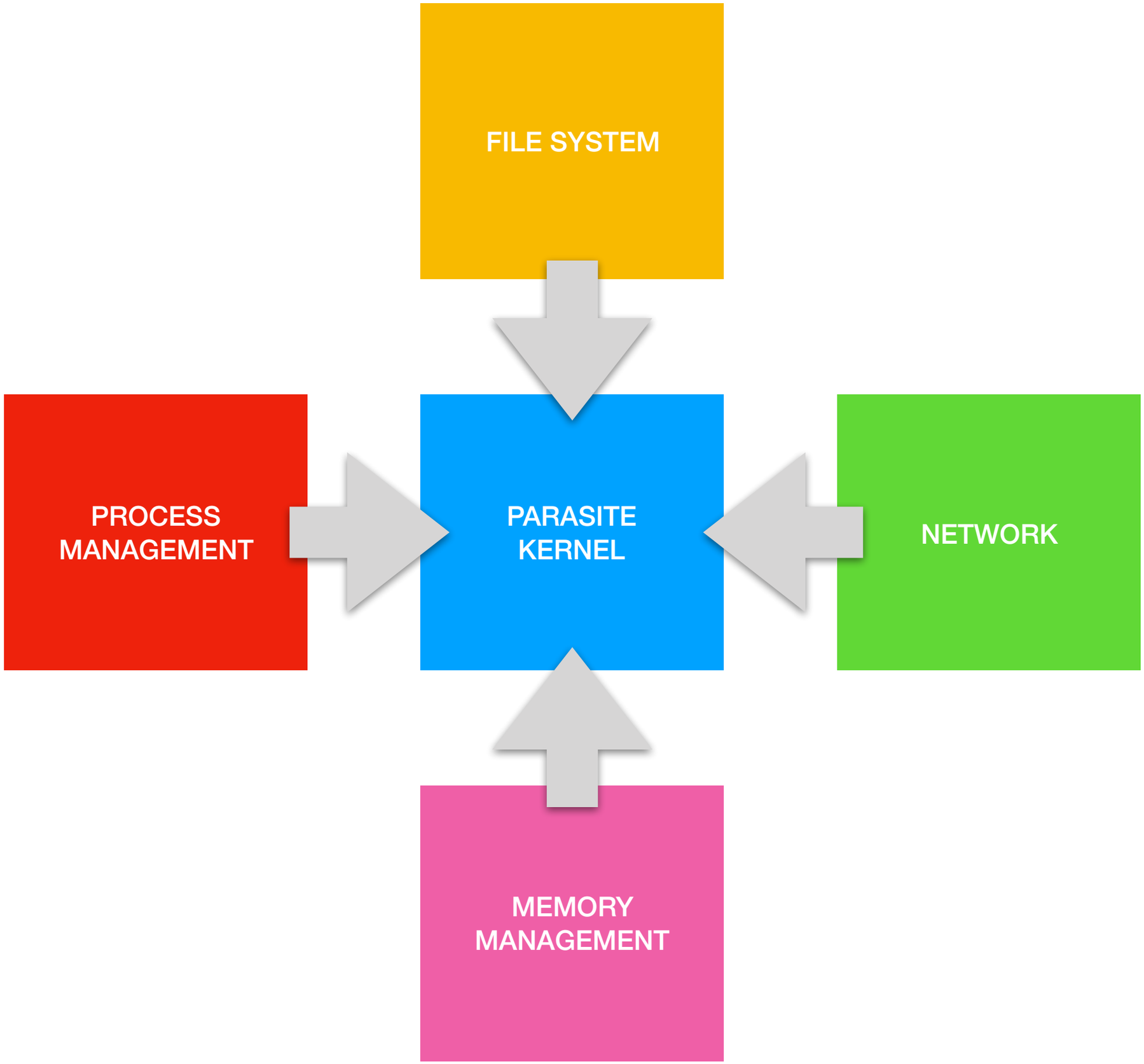
- Patches to kernel code is under surveillance (i.e. - PatchGuard);

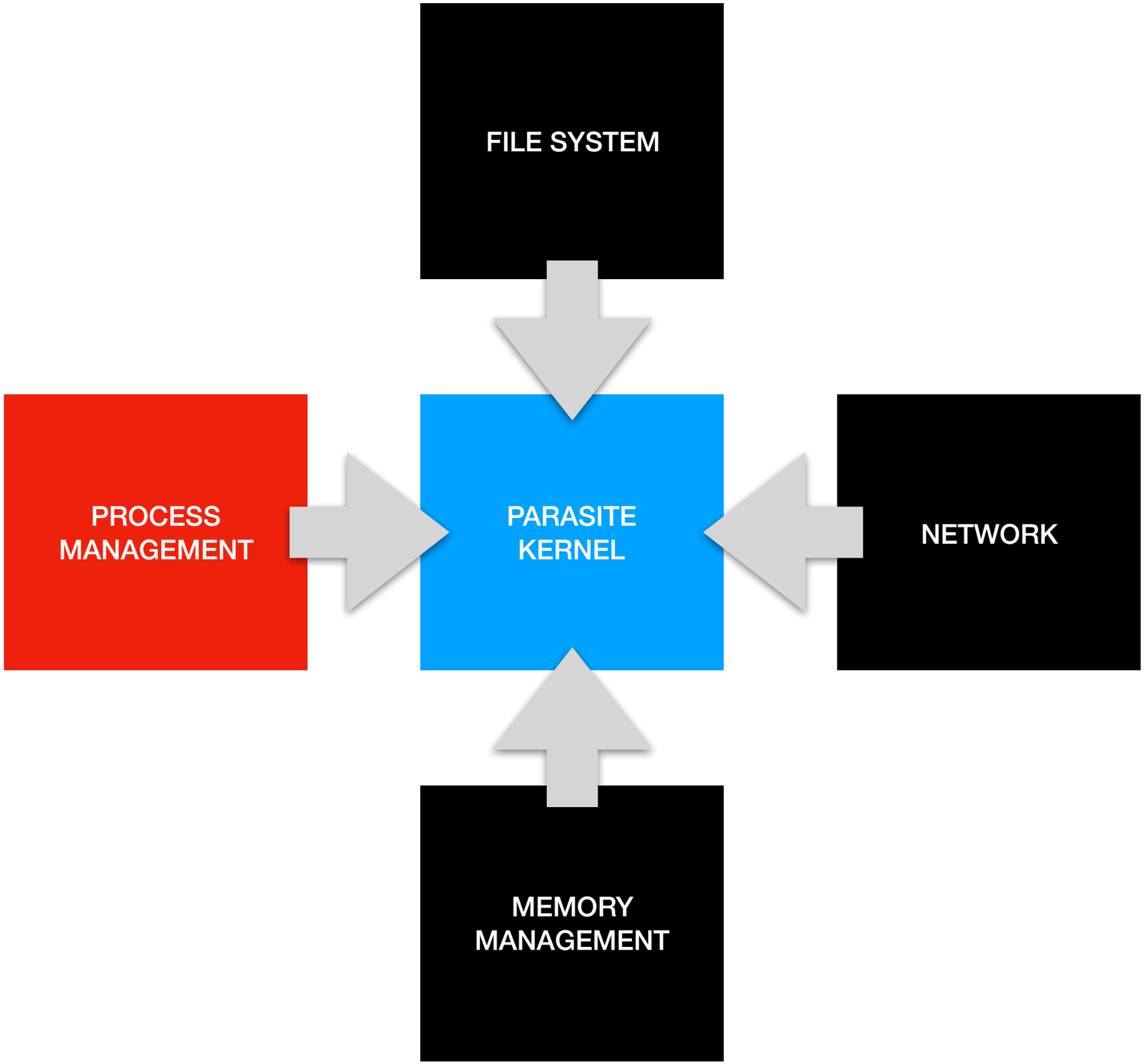
- People are easily tricked into running trojan horses - and with static analysis may eventually block ill intended code;

- JIT execution makes +rwx pages an interesting statistical model challenge to detect malicious code execution.

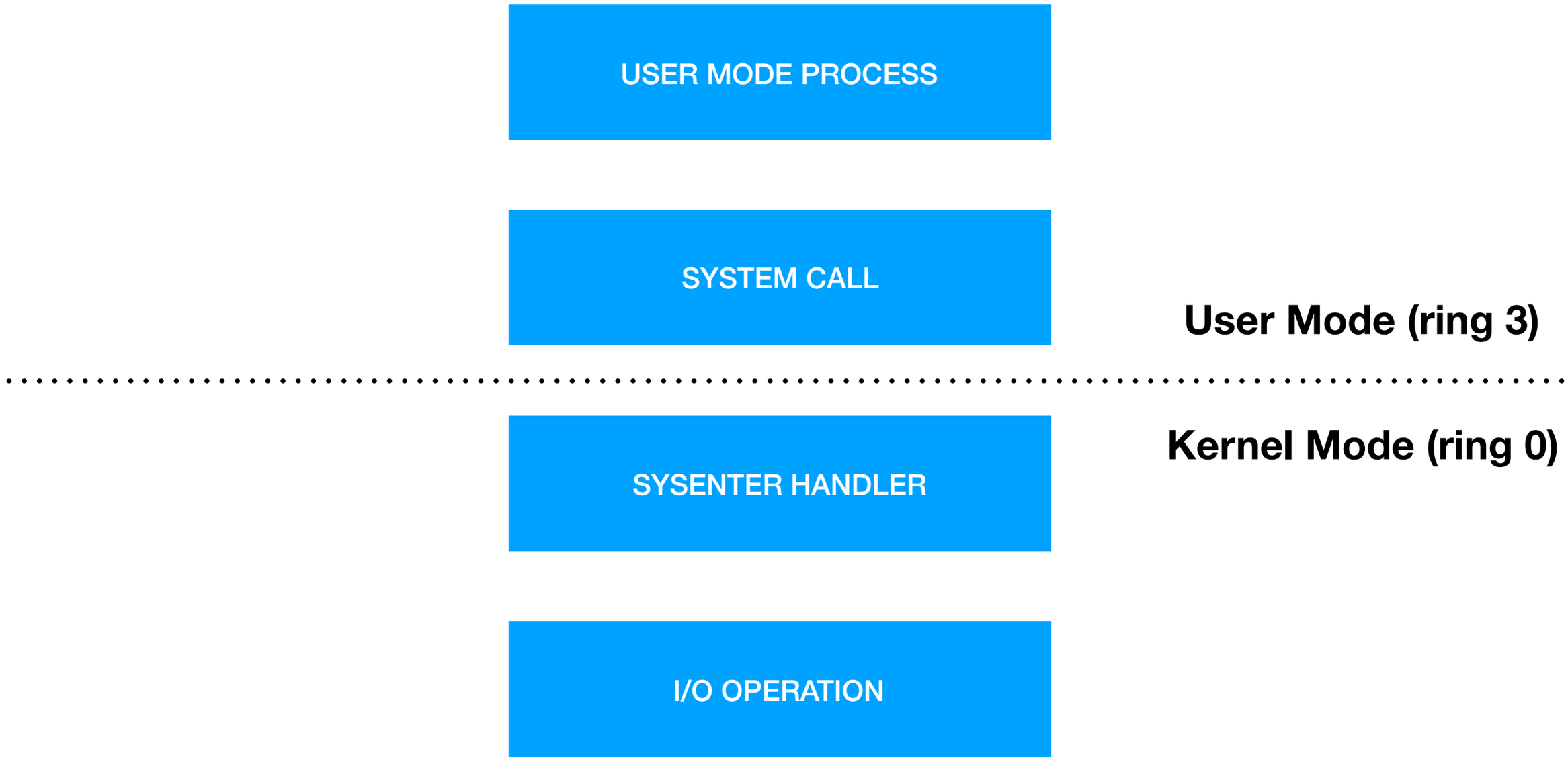
How

- Two pseudo-kernels: user-mode, kernel-mode;
- Process/thread scheduler; basic preemption;
- Semaphore/Event based synchronization;
- Becomes a rootkit by design.


FILE SYSTEM
PROCESS MANAGEMENT
PARASITE KERNEL
NETWORK
MEMORY MANAGEMENT


FILE SYSTEM
PROCESS MANAGEMENT
PARASITE KERNEL
NETWORK
MEMORY MANAGEMENT

Typical System Call



Pseudo system call overview

PROCESS

PSEUDO SYSCALL

WRAPPER

HOST SYSCALL

Parasite OS kernel

Differences: ring0 vs ring3

- Parasite contexts wrapped around kernel-mode threads
- Memory management using kernel-mode heap with r+x permission
- Network implements raw phy packets and an independent TCP/IP stack
- File system is memory-based and persisted
- Harder to detect, as it looks like normal OS-kernel activities

- Parasite contexts wrapped around user-mode threads
- Memory management using user-mode heap with rw+x permission
- Network wraps BSD sockets from host OS
- File system is memory-based only
- Easier to detect, better than most trojans around

Limitations

- Not POSIX compatible; sorry, no ./configure for compiling most of your tools

- No process isolation: all pseudo-processes runs and accesses other pseudo-process' memory.

- Network code is somewhat convoluted as BSD socket is the preferred choice for ring3 code and RAW TCP/IP used for ring0;

- Inefficient process scheduler - no quanta or deadlock prevention - pre-emption is rudimentary;

- Exception handling tries to stabilize processes, but not as robust as an actual OS

Synchronization

- Context switched over blocked system calls;
- Preemption is very hard (problem: how to stop thread execution);
- There’s no interruptions in the pseudo kernel;

Filesystem persistency

- Steganography is a good option;
- Exploit TLV (type/length/value) on something:
  - Windows Registry;
  - Page file.

Network infiltration

- DNS triggering;

- Junk email;

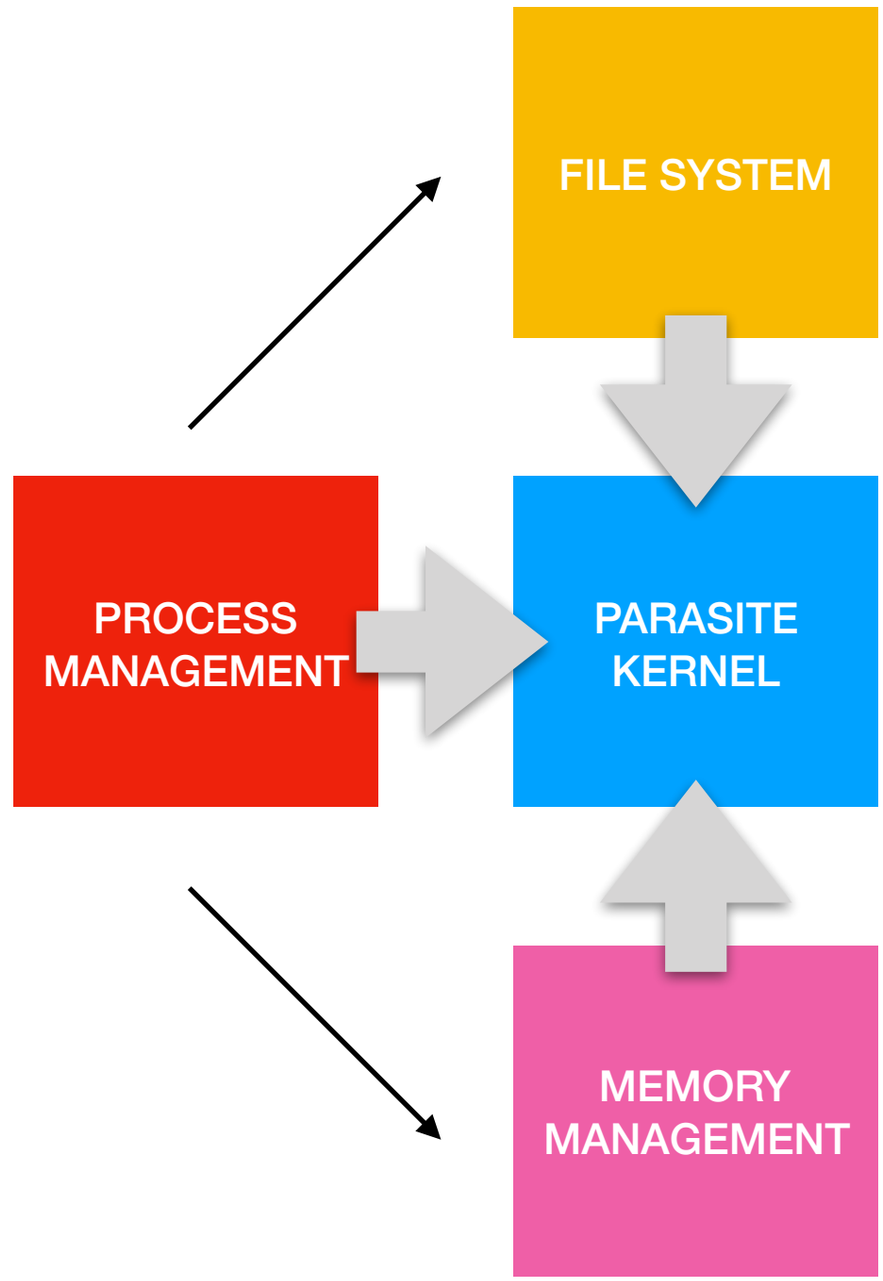
Network exfiltration

- DNS tunnel;

- JavaScript injection;

Process Creation

- Allocates memory;
- Loads binary into memory (.text*,.*data*,.bss)
- Sets section’s permissions
- Creates process structure and adds to process list
- Returns process handle if called using asynchronous flag, otherwise, switch context to just created process.



Future

- Should Parasite OS be free? Open? Consequences.
- Code completion
- Alter’s compiler for rogue calling convention (obfuscation).

Questions?

csh@outlook.com