Syscall Proxying || Pivoting Systems

Filipe Balestra

filipe@balestra.com.br

Rodrigo Rubira Branco

rodrigo@kernelhacking.com
rodrigo@risesecurity.org

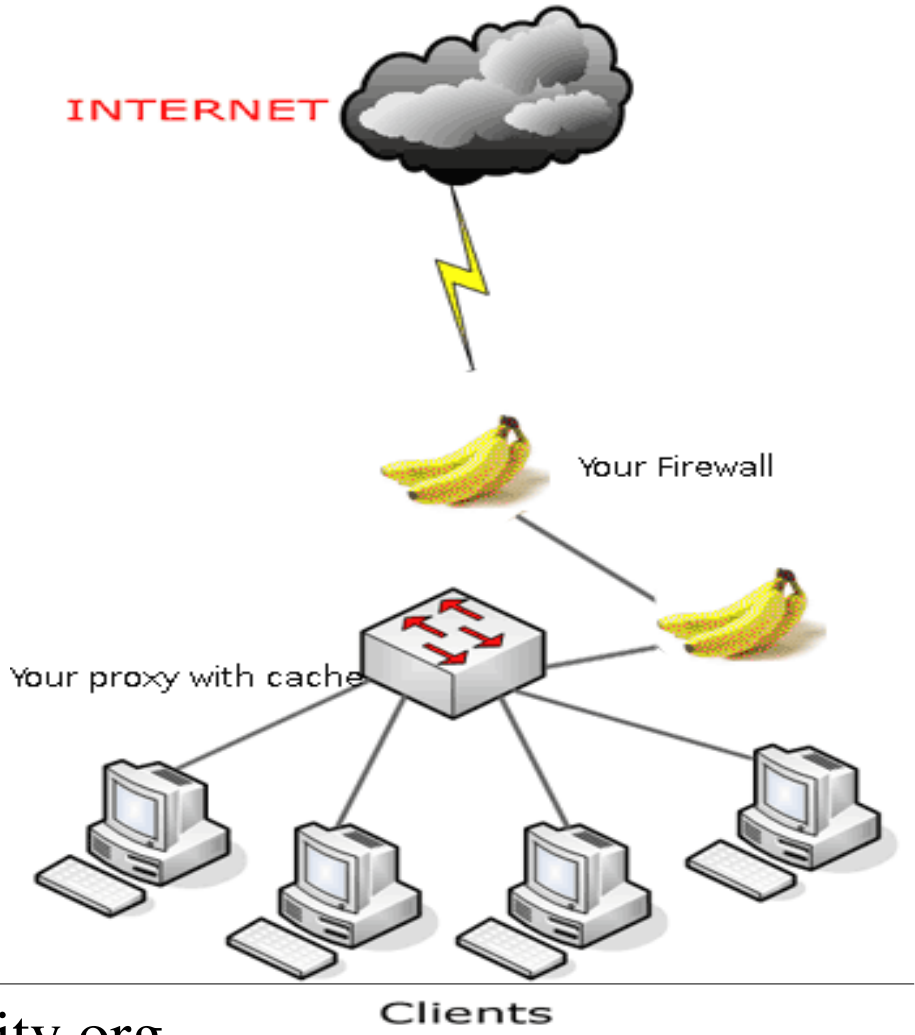
Agenda:

RESEARCH GROUP

- What is Syscall Proxying?
- Why Syscall Proxy is better than the open-source Squid?
- Why not open the Syscall Proxy source-code? Security Reasons!
- Understand the proxy architecture

Front view

Rear view



Agenda (the real one!):

- Background knowledge
- What is Syscall Proxying?
- Why I need it or When it's interesting?
- Difficulties
- Some samples
- The Future
- Acknowledges
- References

Background Knowledge

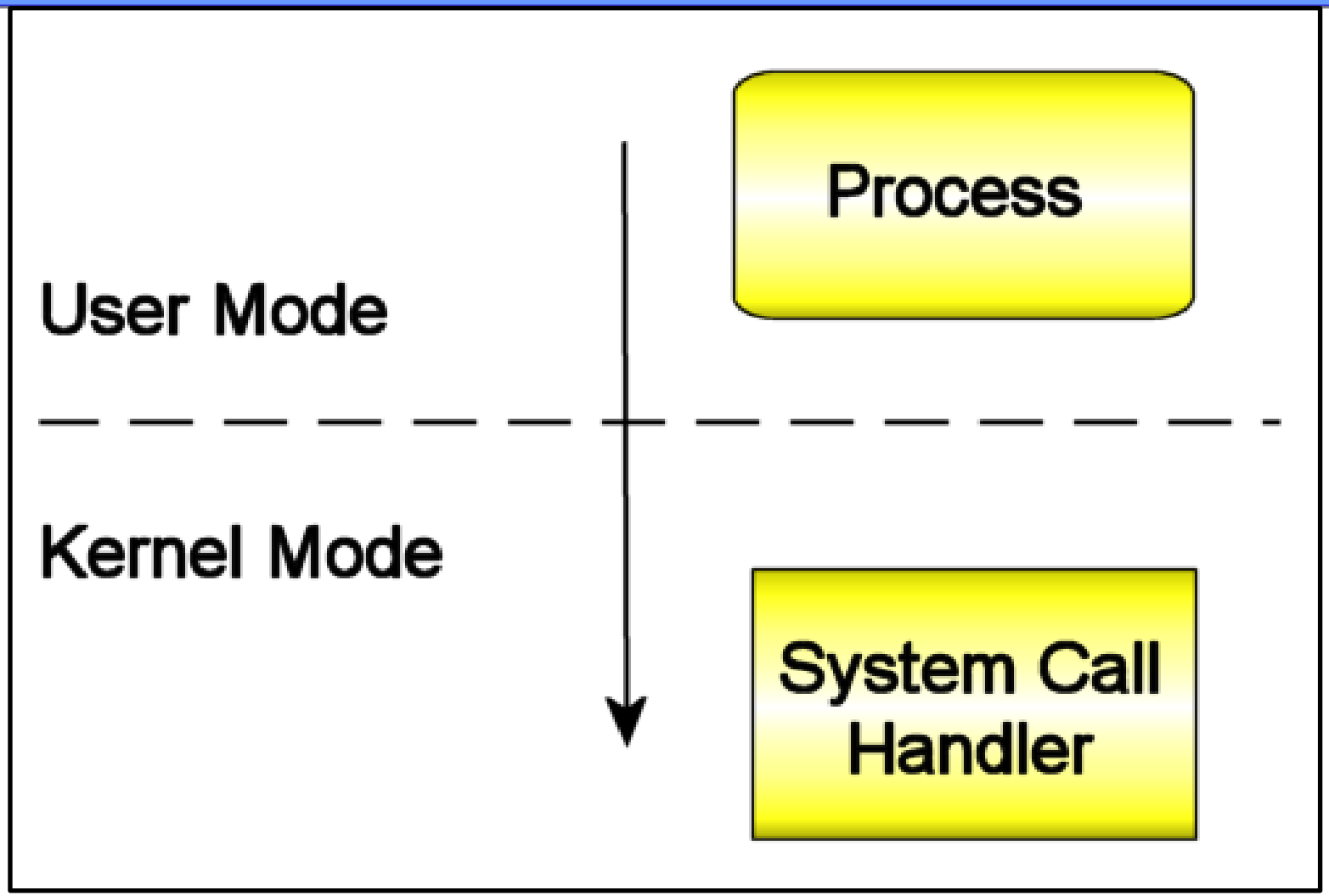
- When a process need any resource it must perform a system call to ask the operation system for the needed resource.

- Syscall interface are generally offered by the libc (the programmer doesn't need to care about system calls)

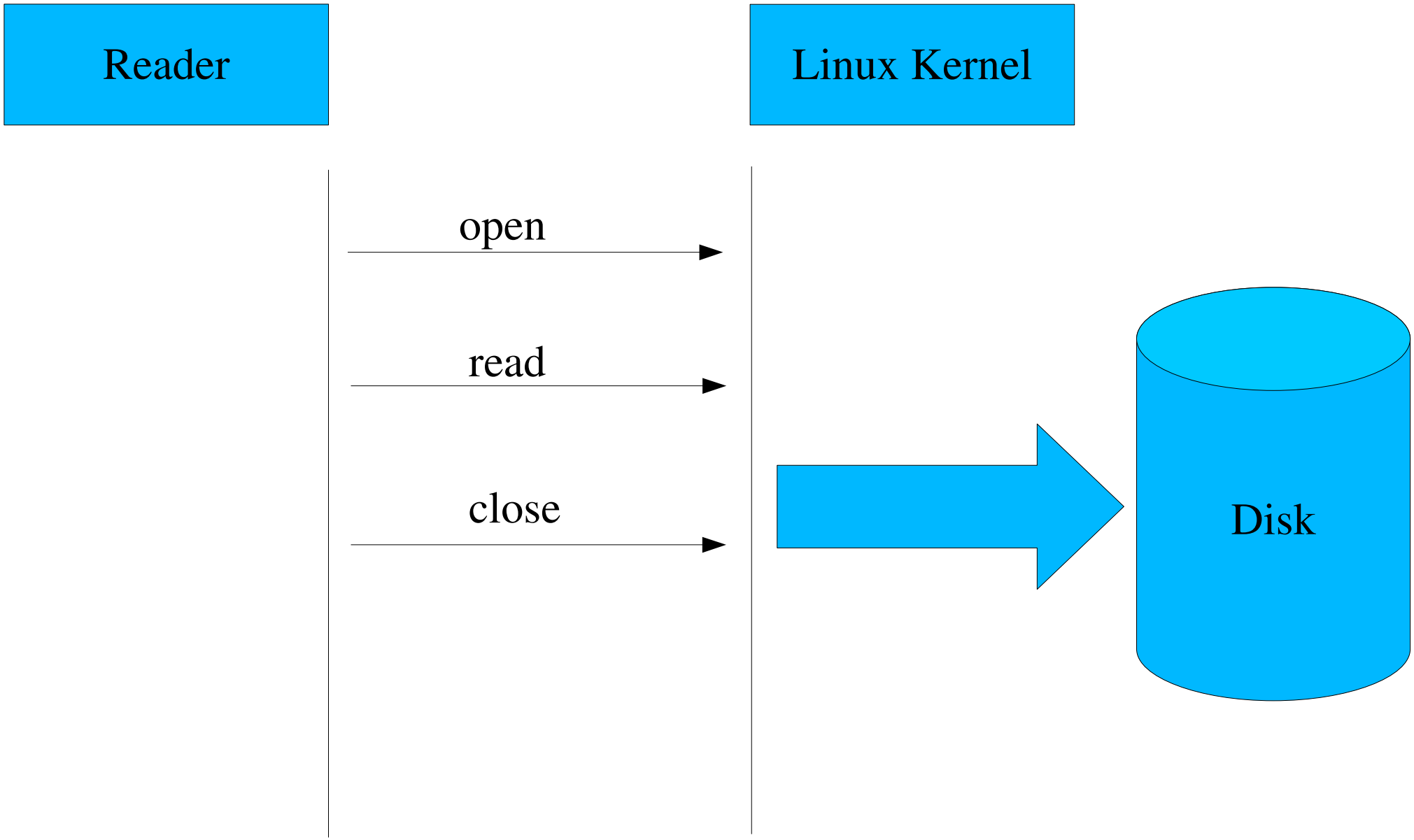
- We will discuss syscall proxying under Linux environment, so, some aspects:
    * Homogeneous way for calling syscalls (by number)
    * Arguments passed via register (or a pointer to the stack)
    * Little number of system calls

RESEARCH GROUP

Background Knowledge



Syscall Interface – Read from a file



Syscall Interface – Read from a file

```
# strace cat /etc/passwd


execve("/bin/cat", ["cat", "/etc/passwd"], [/* 17 vars */]) = 0
...
open("/etc/passwd", O_RDONLY|O_LARGEFILE) = 3
...
...
read(3, "root:x:0:0:root:/root:/bin/bash\n"..., 4096) = 1669
...
close(3)                                = 0
```

- As we can see using the strace program, even a simple command uses many syscalls to accomplish the task

Syscall Interface – Passing arguments using registers

- EAX holds the system call number

- EBX, ECX, EDX, ESI and EDI are the arguments (some system calls, like socket call can use the stack to pass arguments)

- Call int $0x80 (software interrupt)

- Value is returned in EAX

RESEARCH GROUP

What is Syscall Proxying?

- The idea is to split the default syscall functionality in two steps:

* A client stub
Receives the requests for resources from the programs
Prepair the requests to be sent to the server (called marshalling)
Send requests to the server
Marshall back the answers

* A syscall proxy server
Handle request from the client
Convert the request in the native form (in our case, just linux)
Calls the asked system call
Sends back the response

RESEARCH GROUP

I don't wanna ask again! What a Hell is Syscall Proxying?

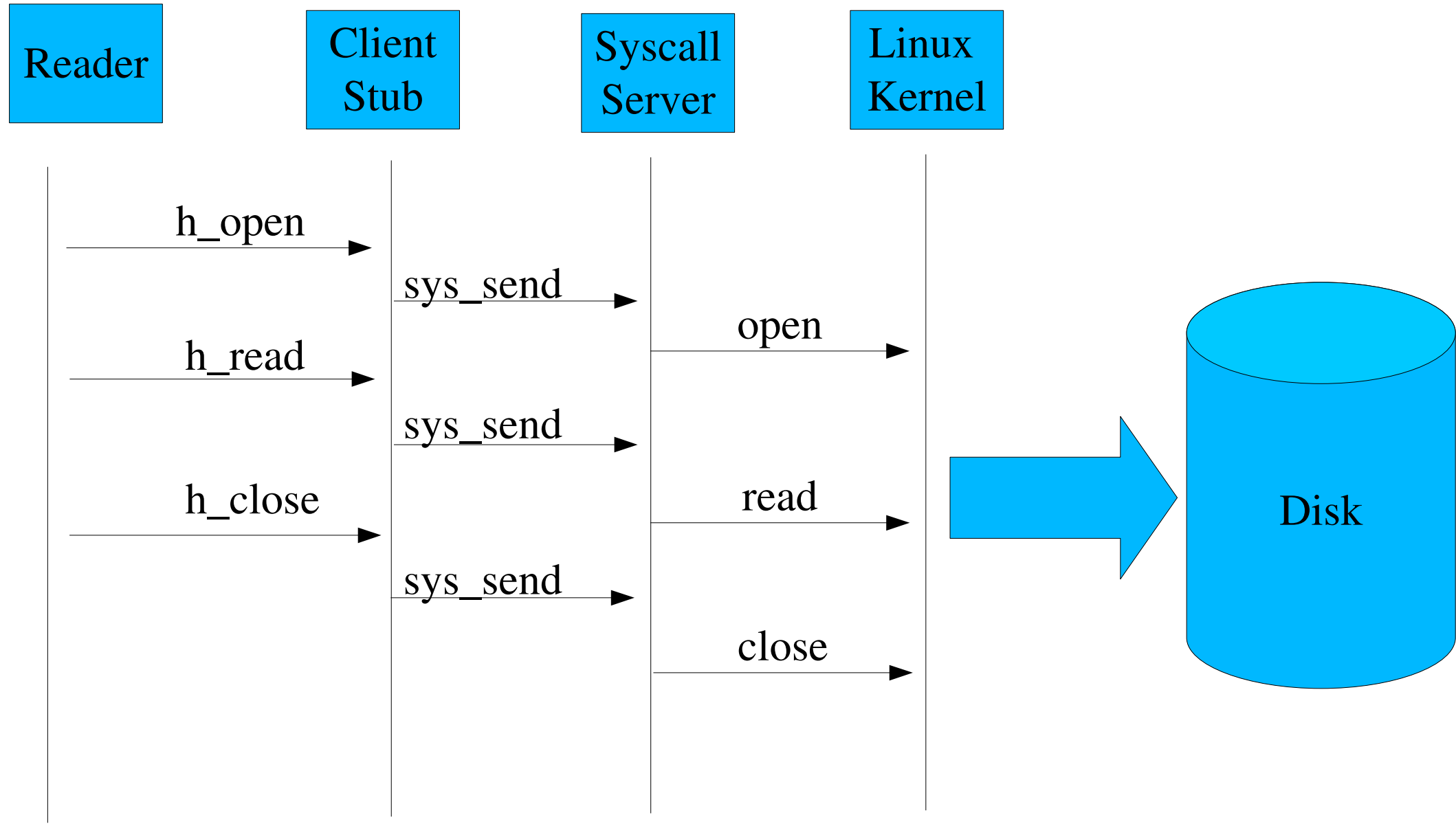
- A way to use many tools without install anything in an owned machine (in an attack or in a penetration test)

- Just play with the memory

- Use your own tools (for your own native system) and don't care about what is the operation system owned (the syscall proxy server for sure need to be coded for many operation systems)

- Pivoting is because you can use the syscall proxy server to attack many other servers in the local network of the syscall proxy

Syscall Interface – Read from a file



RESEARCH GROUP

Why I need it or When it's interesting?

- The syscall proxy server can be injected in a remote process and offer an interface to the underlying operation system (including remotely 'local' privilege escalation)

- You can use any tools you like to your own system, without change then (if you use LD_PRELOAD)

- The local process you are running under the stub does not know it is running remotely

- Easily infection of remote process and the kernel itself (using the IDT patch you can easily execute arbitrary commands in the kernel – showed by me in the latest Hackers to Hackers Conference)

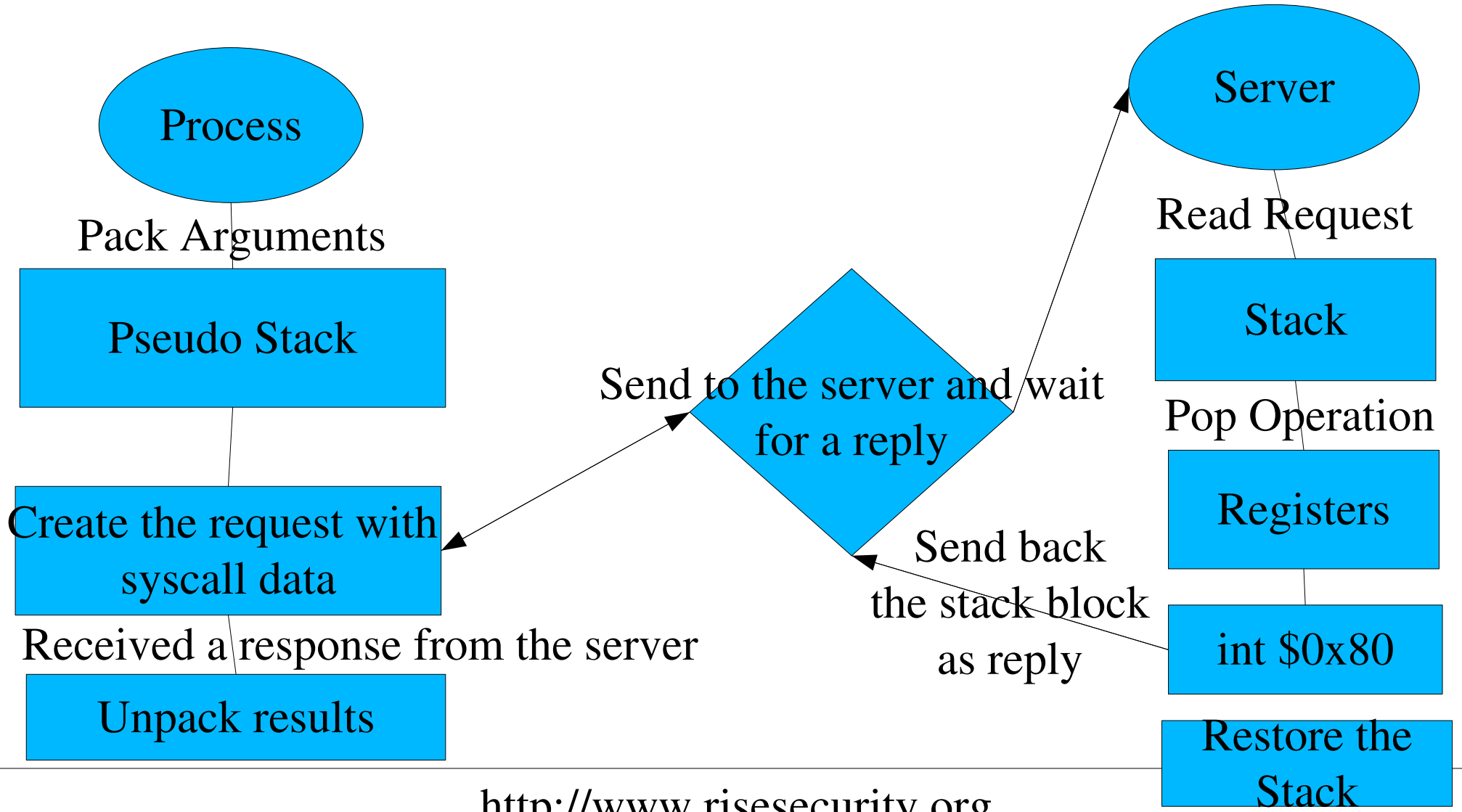
Difficulties

- RPC does the same idea that we have showed, but the problem is in the server-side we have lots of complexity

- We need a small piece of code in the server-side, because it will be injected in the vulnerable program

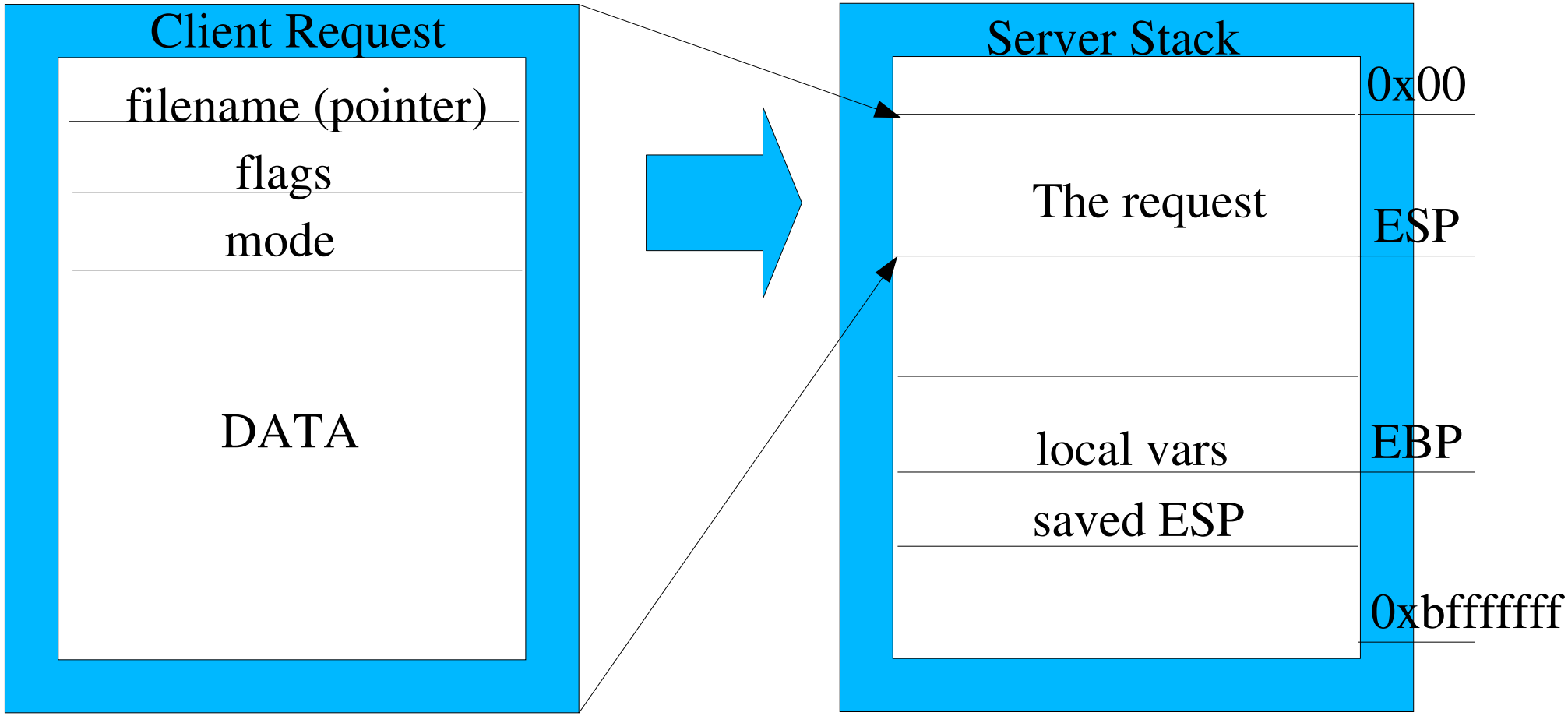
- To accomplish that, we have in the client-side the 'image' of the stack-state of the server-side

Difficulties



Difficulties

- In the beginning of a connection, the server returns its ESP



Some Samples

- Remote Shell using syscall proxying

- Remote Process Injection (using ptrace – be explained in another h2hc 3 presentation)

- Remote Scanning (pivoting system)

- Remote Kernel Infection (not using IDT patch – explained in my presentation at h2hc 2)

The Future

- Finish the stub implementation of more system calls

- Have a better version of the syscall proxy server for *BSD systems

- Optimize the size of the syscall proxy server

- Optimize the speed of execution of the syscall proxy scheme (for remotely sniffing sections)

RESEARCH GROUP

Acknowledges

- RISE Security

- H2HC Organization? I'm included!

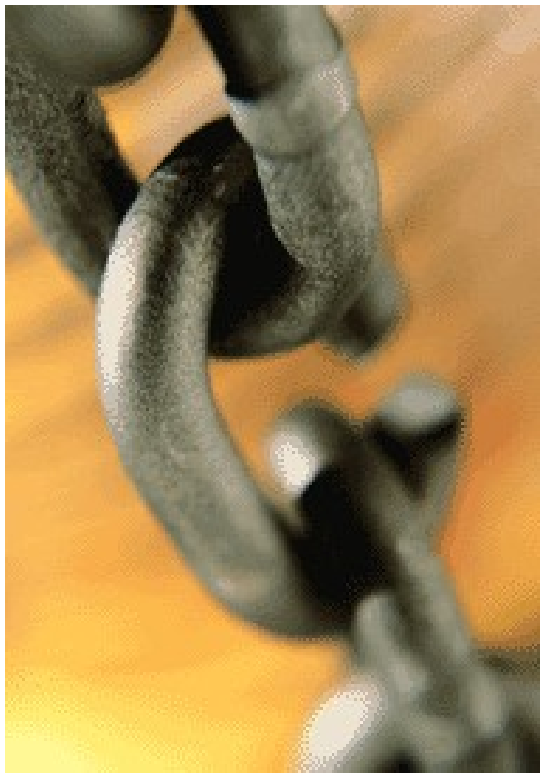
- Casek from UberWall for some news and ideas

- Tiago Assumpção – First person to talk with me about it (~2003)

- Ramon de Carvalho Valle from RISE - Assembly Master

- Your patience!

- Without friends, we are nothing, let's drink!



References

- Syscall Proxying – Simulating Remote Execution by Maximiliano Caceres http://www.coresecurity.com/blackhat2002.htm -> This presentation are embased in lot of ideas and samples showed here

- Syscall Proxying fun and applications by Casek http://events.ccc.de/congress/2005/fahrplan/events/553.en.html -> The sample injection and scan codes are ideas shared by Casek in his presentation

- RISE Security - http://www.risesecurity.org

- Personal Website - http://www.kernelhacking.com/rodrigo

END! Really is?

DOUBTS ?

Filipe Balestra

filipe@balestra.com.br

Rodrigo Rubira Branco

rodrigo@kernelhacking.com
rodrigo@risesecurity.org