Iván Arce | ivan.arce@corest.com

Automating Penetration Tests

CORE
SECURITY TECHNOLOGIES

Outline

- Problems in the current Penetration Test practice
- Automating Penetration Tests
- The technical challenges
- Overcoming the the technical challenges

Problems in the current penetration test practice

The Penetration Test

- Rationale:
  - "Improving the security of your site by breaking into it"

    Dan Farmer & Wietse Venema, 1993
    http://www.fish.com/security/admin-guide-to-cracking.html

- A plausible definition:
  - "A localized and time-constrained attempt to breach the information security architecture using the attacker's techniques"

The Penetration Test

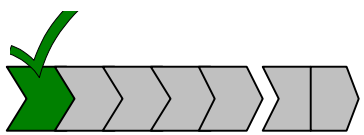
How is it usually done?

- Information Gathering
- Information Analysis and Planning
- Vulnerability Detection
- Penetration
- Attack/Privilege Escalation
- Analysis and reporting
- Clean-up

Problems in the current Penetration Test practice

Information Gathering Phase:

OK



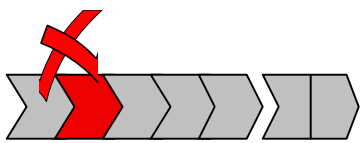
- **Public organization information**
- **M&A, SEC fillings, patent grants,etc.**
- **Job openings**
- **Employee information**
- **Web browsing**
- **Web crawling**
- **Mailing list and newsgroups posts**
- **Nmap, traceroute, firewall, ping sweeps, etc**
- **NIC registrations**
- **DNS records**
- **SNMP scanning**
- **OS fingerprinting**
- **Banner grabbing**
- **War dialers**
- **Social engineering**
- **Dumpster diving**
- **Etcetera**

Problems in the current Penetration Test practice

Information Analysis and Planning Phase:

Not OK

- **Difficult and time consuming task of consolidating all the information gathered and extract high level conclusions that will help to define an attack strategy**
- **Hard to keep an up to date general overview of the components and their interaction**
- **No specific tools aimed at addressing this phase**
- **Experienced and knowledgeable resources required for this stage, overall time constraint could limit the extent of their work**
- **No formal processes or tools to help estimate time and efforts**

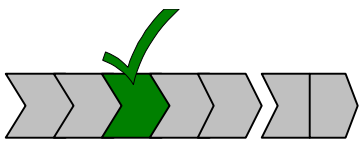


Problems in the current Penetration Test practice

Vulnerability Detection Phase:

OK

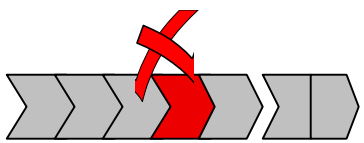
- **Large variety of tools available:**
  - Commercial Vulnerability scanners
  - Free & Open source scanners
  - Application level testing tools
  - OS specific testing tools
- **Large amount of information available:**
  - Publicly known vulnerability information
  - Vulnerability database
  - Various sources of security advisories (vendors, CERTs, information security companies, etc.)
  - SecurityFocus.com
  - Bugtraq, NT bugtraq, pentest mailing list
  - Newsgroups, papers, CVE
- **BUT In-house research is not avoidable**



Problems in the current Penetration Test practice

Penetration Phase:

Not OK



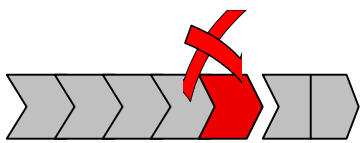
- **Although there are some tools available, they generally require customization and testing**
- **Publicly available exploits are generally unreliable and require customization and testing (quick hacks, proof of concept code)**
- **In-house developed exploits are generally aimed at specific tasks or pen test engagements (mostly due to time constraints)**
- **Knowing that a vulnerability exist does not always imply that it can be exploited easily, thus it is not possible to successfully penetrate even though it is theoretically possible (weakens the overall result of the engagement)**
- **Knowledge and specialization required for exploit and tool development**
- **Considerable lab infrastructure required for successful research, development and testing (platforms, OS flavors, OS versions, applications, networking equipment, etc.)**

Problems in the current Penetration Test practice

Attack/ Privilege Escalation Phase:

Not OK

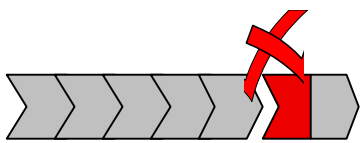
- **Some tools and exploits available, usually require customization and testing (local host exploits, backdoors, sniffers, sniffing/spoofing libraries, etc.)**
- **Monotonous and time consuming task: setting up the new "acquired" vantage point (installing software and tools, compiling for the new platforms, taking into account configuration specific details, etc.)**
- **Pivoting might be a key part for success in a pen test yet it is the less formalized process**
- **Considerable lab infrastructure required for research, development, customization and testing**
- **Lack of a security architecture for the penetration test itself.**



Problems in the current Penetration Test practice

Analysis and Reporting Phase:

Not OK



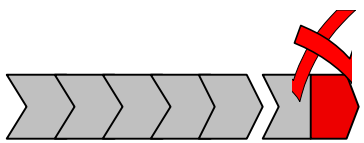
- **Maintaining a record of all actions, commands, inputs and outputs of all tasks performed during the pentest is left as methodology to be enforced by the team members, that does not guarantee accountability and compliance.**
- **Gathering and consolidating all the log information from all phases, including all the program and tools used, is time consuming, boring and prone to error**
- **Organizing the information in a format suitable for analysis and extraction of high level conclusions and recommendations is not trivial**
- **Analysis and definitions for general conclusions and recommendations require experienced and knowledgeable resources**
- **The actual writing of final reports is usually considered the boring leftovers of the penetration test, security expertise and experience is required to ensure quality but such resources could be better assigned to more promising endeavors**
- **No specialized tools dedicated to cover the issues raised above**

Problems in the current Penetration Test practice

Clean Up Phase:

Not OK

- **A detailed and exact list of all actions performed must be kept, yet there are just rudimentary tools for this**
- **Clean up of compromised hosts must be done securely and without affecting normal operations (if possible)**
- **The clean up process should be verifiable and non-repudiable, the current practice does not address this problem.**
- **Often clean up is left as a backup restore job for the pentest customer, affecting normal operations and IT resources.**



Summary

- ✓ Information Gathering
- ✗ Information Analysis and Planning
- ✓ Vulnerability Detection
- ✗ Penetration
- ✗ Attack/Privilege Escalation
- ✗ Analysis and reporting
- ✗ Clean-up

Automating Penetration Tests

Automating Penetration Tests

- Why?
- What is it good for?
- What are the technical challenges?
- How could they be addressed?

Rationale

- Penetration tests are becoming a common practice that involve a mix of hacker handiwork, monotonous tasks and non formal knowledge. Automating penetration tests will bring professionalism to the practice.

APT:

What is it good for?

- To make available valuable resources for the more important phases: high level overview and analysis, strategic attack planning, results analysis and recommendations.
- To encompass all the penetration test phases under a single framework
- To define and standardize the methodology

APT:

What is it good for?

- To enforce following of the methodology and ensure quality
- To improve the security of the practice
- To simplify and speed up monotonous and time consuming tasks

The Technical Challenges

The Technical Challenges (1/3)

- Modeling penetration testing, considering all phases in an intuitive and usable fashion
- Building a tool that reflects the model capable of adopting arbitrary methodologies defined and redefined by the user
- Development and maintenance of a wide range of exploits for different platforms, operating systems and applications and multiple combinations of versions

The Technical Challenges (2/3)

- Assurance that the developed code is functional under different network and host configurations (reliability)
- Addressing the attack/privilege escalation phase in a seamless way.
- Handling interactions between different exploits
- Building a framework that lets the team develop and customize new or existing exploits quickly

The Technical Challenges (3/3)

- Not having to re-invent the wheel each time a new vulnerability is discovered
- Keeping such a beast manageable in terms of size and complexity
- Providing different degrees of ‘stealthy-ness’ (to comply with pen-test requirements)
- Having autonomous capabilities (worm-like?)
- Having mechanism for acquiring and reusing knowledge and experience from successive penetration tests

The Technical Challenges

And more…

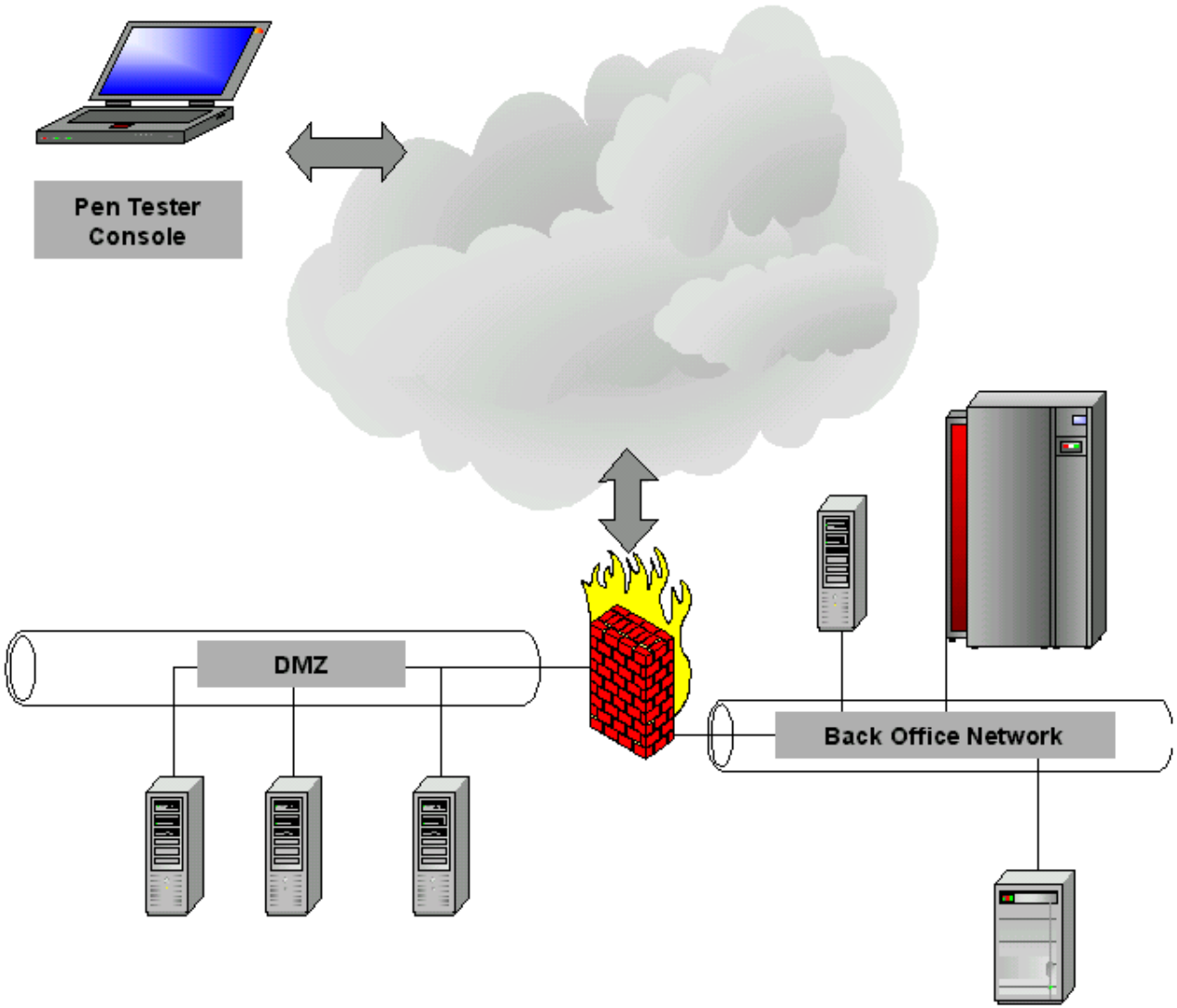
- Buffer overflows
  - **Exec/no-exec stack**
  - **Multiple platforms/Multiple Operating systems**
  - **Encoding, compression, encryption, etc.**
- Sniffing/Spoofing
- IP Stack based attacks

Overcoming the technical challenges

The model

- Simplify and abstract all the components of the system and their relations
- Provide a foundation on which to build
- Provide a common language to talk about the different components

The model



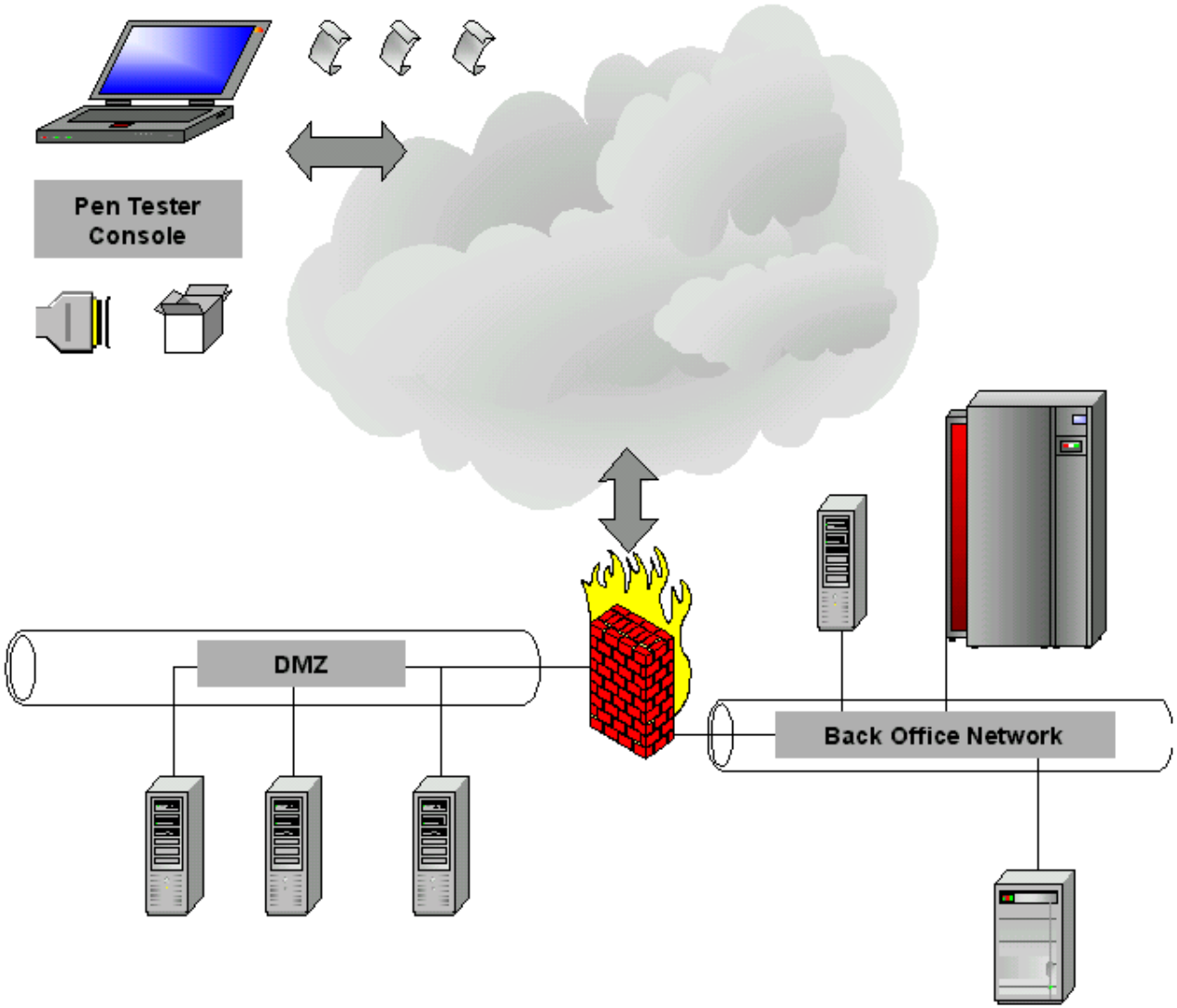
Agents and Modules

- Agents
  - "The pivoting point" or "the vantage point"
    - Run modules
    - Installable on any compromised host
    - Local stealth techniques for hiding (ala rootkit)
    - Some autonomy (worm-like) and limited life-span
    - Secure (shouldn't render the client infrastructure more insecure than before the pentest)
    - Remotely control other agents
    - Clean up functionality (uninstall)

Agents and Modules

- Modules
  - "Any executable task"
    - Information gathering
    - Information analysis
    - Actual attacks, exploit code
    - Report generation
    - Scripting
    - Meta-modules
    - Simple and easy to extend
    - Have every tool together, under the same framework

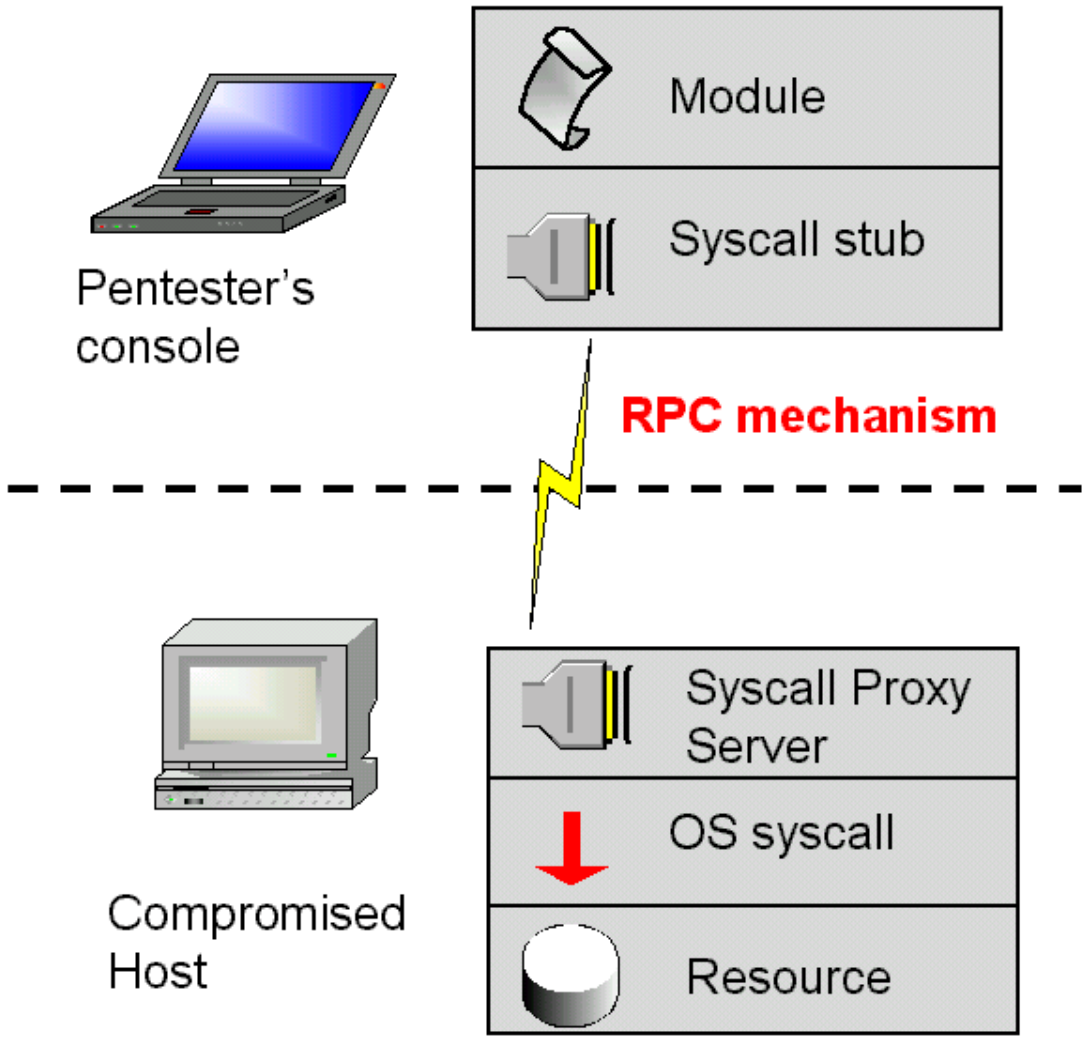
Agents and Modules



Syscall Proxying

- Provides a uniform layer for the interaction with the underlying system
- All modules ultimately access any resource through this layer
- Changing this layer with a proxy effectively simulates the remote execution of the module

Syscall Proxying



Syscall Proxying

- Example case (break chroot, Linux style):

```
main(int argv, char **argc)
{
        mkdir("bla", 0755);
        chroot("bla");
        chdir("../../../../../..");
        chroot(".");
}
```

Now, run it locally but proxy the system calls!

Overcoming the Technical Challenges

Syscall Proxying

Pentester's console

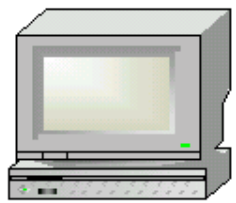
mkdir("bla", 0755);

chroot("bla");

chdir("../../../../../..");

chroot(".");

RPC mechanism



Compromised Host

mkdir("bla", 0755);

chroot("bla");

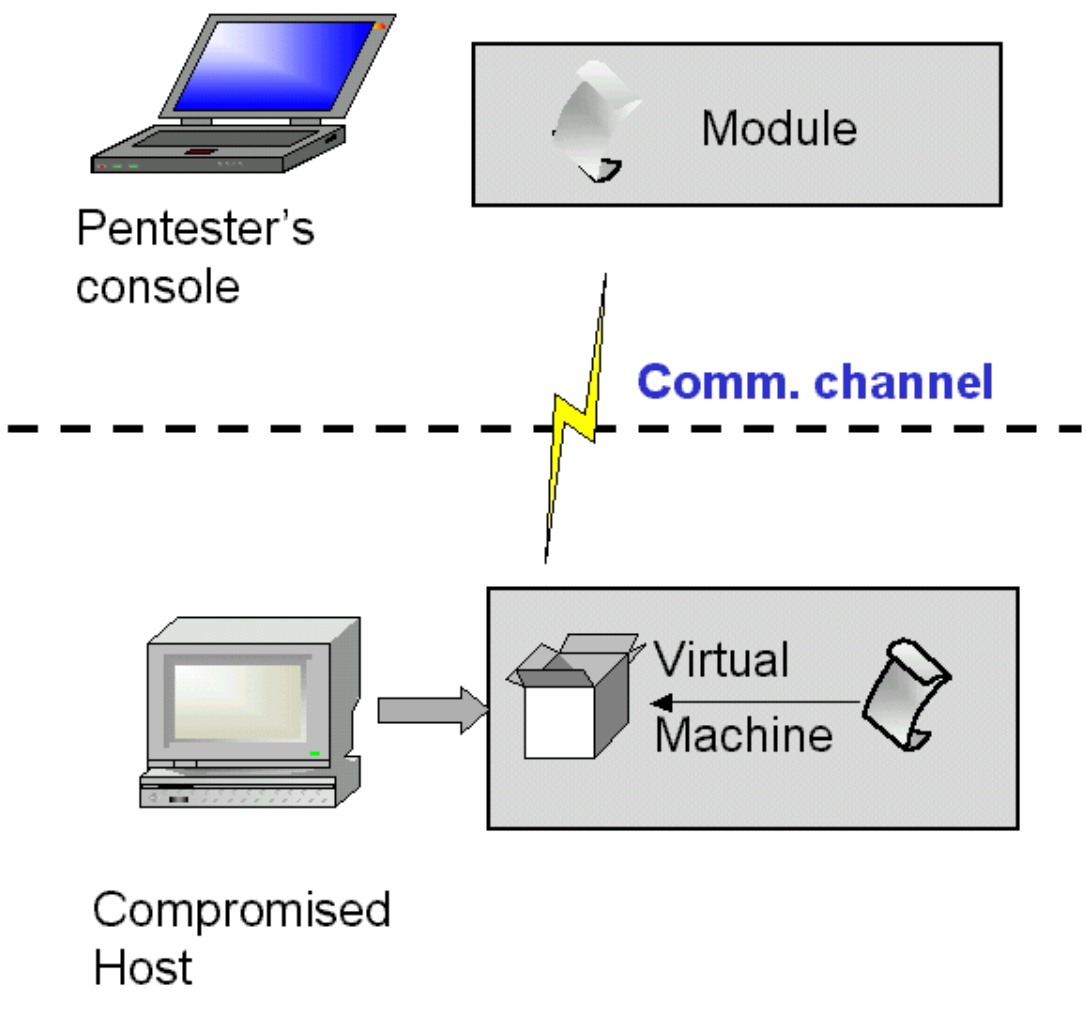
chdir("../../../../../..");

chroot(".");

Virtual Machine

- Isolates the particular characteristics of the “pivoting host” platform from the module
  - This effectively eliminates all the burden related to the setup of a vantage point
  - Just port the VM
- Provides a comfortable environment for the development of new exploits
  - Productivity is higher on interpreted languages than on compiled ones
- Provides a simple way of scripting (automating) any task, even higher level ones
- Lots of free and powerful VMs are available (Perl, Python, Squeak)

Virtual Machine



APIs and Helpers Libraries

- Any common and general use functionality related to the coding of exploits should evolve into an API
  - Prioritizes code-reuse and sharing
  - Simplifies exploit code, focused on the particular vulnerability and not on common vulnerability-writing tasks
  - Makes the life of the exploit developer easier (just build on top of existing code)
  - API's can evolve independently of written exploits
- Some examples
  - Shellcode building for different platforms
  - Sniffing and packet parsing
  - Spoofing (packet crafting)
  - Application layer protocols
    - HTTP, FTP, DNS, SMTP, SNMP, etc

APIs and Helpers Libraries

- Example case (sample exploit, automated egg generation API):

```
h = get_host(target)
e = get_egg(h.os, h.arch, “syscall-proxy-srv”)
jmp_addr = h.os.sp + offset
e = ListenEgg(e)
e.set_listen_port(31337)
e = ASCIIEgg(e)
msg = join( [ “HELO “, e.get_buffer, jmp_addr], “”)

s = socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(h, 25)
s.send( msg )

a = Agent(h, 31337)
output (a)
```

Component
Communications

- Use crypto protocols to provide privacy & mutual authentication
  - Do not weaken the infrastructure being tested
- Define an abstract “transport” than can be interchangeable and mounted on top of any networking protocol
  - Firewall piercing
  - Covert channels
  - IDS evasion
- Agent chaining (ala source-routing)
  - Provides a way of “jumping” between vantage points, allowing communication across diverse security domains (with different security policies)

Scripting

- Scripting of modules
  - Autonomous action (for more worm-like attacks, or for scenarios where online communication with agents before compromise might not be possible)
  - A more constructive approach to module development:
    - Build higher level attacks/strategies using available modules
  - BUT, please! Do not invent YASASL

Scripting

- Example case (a wild IIS worm):

```
map = get_module("network-mapping")
map.run( { "TARGET" :  "192.168.88.0/24"  } )
net = get_network("192.168.88.0/24")
for h in net:
        fingerp = get_module("os-fingerprint")
        fingerp.run( { "TARGET" : h, "PORT" : "80" } )
        if h.get_port(80) ) == NetService.OPEN
                        and h.os == os.WIN2K:
                                attack = get_module("iis-printer")
                                attack.run( { "TARGET" : h } )
                                if h.has_agent:
                                    output "Agent installed on ", h
```

Logging and Reporting

- Since a single-tool / single-framework is used for all the pen-test related tasks, it's easy to keep logs of every single activity
- Getting the information together and building a report can be done by a module that accesses the objects in the model

Logging and Reporting

- Example case (a report generating module):

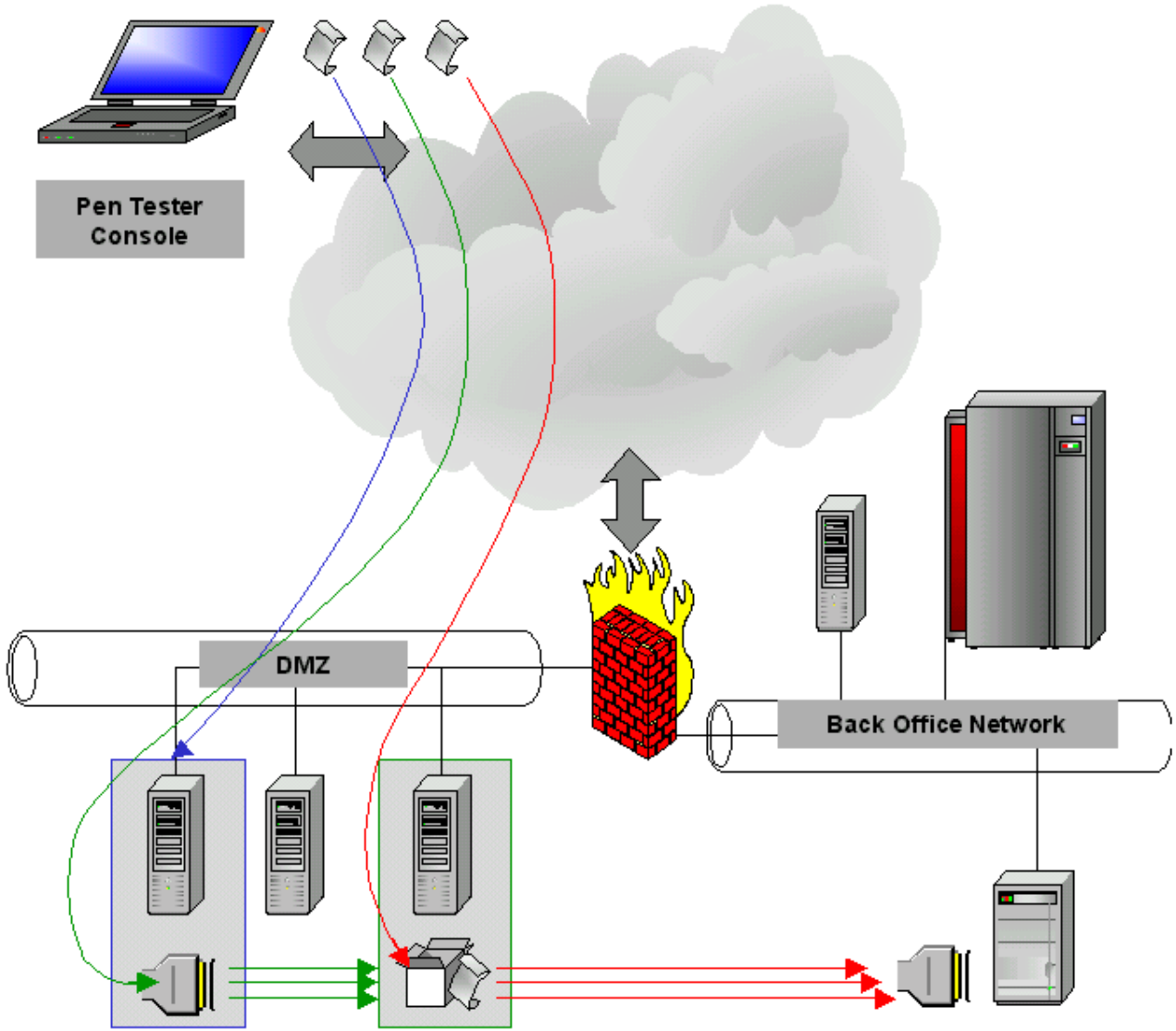
```
net = get_network(“192.168.88.0/24”)
output “Compromised hosts on network“, net
for h in net:
        if h.has_agent:
                output h
                modules = log.get_executed_modules(h)
                for m in modules :
                        if m.successful:
                                output m.description
```

Knowledge Base

- A database of information on common attack strategies and success configurations on common customer scenarios
- Guidelines on how to do a specific pentest depending on target characteristics
  - IT & security infrastructure
  - Technology in use
  - Organization's business model
- Full activity logs
  - Easier to identify common strategies & trends along different projects
  - Easier to do accurate planning

Overcoming the Technical Challenges

The Whole Picture



Thank You!

Iván Arce
ivan.arce@corest.com

Elías Levy
aleph1@securityfocus.com