A Penetration Testing Learning Kit

-Ariel Waissbein-

TROOPERS 08

Munich, Germany

April 24, 2008

Objectives

- Present a methodology for teaching pentesting.
  - We take a holistic approach.
  - Discuss what's needed to teach pentesting.
- Introduce a pentesting simulation tool.
  - Show how it covers many teaching necessities.
- Invite you to teach with this tool and discuss future plans.

Outline

1. A first lesson.
   i. Pentesting history & motivation.
   ii. A first attack, dissected.
2. Introducing the Learning Kit.
   i. Functionalities of the simulation-attack suite.
   ii. A glimpse on the simulation-attack suite.
3. Completing the lesson into a course.
4. Other applications for the kit.
5. Discussion.

Our approach is to abstract highly technical functions, e.g., treat them as APIs.

- Is there a way to learn pentesting gradually, without first being an expert in:
  - OS security.
  - Network security.
  - Cryptography.
  - Access controls & authentication.
  - Et cetera (Application security, wireless, voip, embedded devices,…).

- We'll get there with our teaching methodology.
  - designing attacks aided by a theoretical model,
  - executing them with a pen-testing framework that abstracts functionalities, and
  - replacing real networks with simulations.

Disclaimer: Next we will make a pentest mockup. But I won't demo… I'll only show some figures. All of you interested in testing the kit, shoot me an email and I'll send you credentials to download a preliminary version.

A FIRST PENTESTING CLASS

Becoming a pen tester

This course introduces you into pentesting by letting you practice pentests.

Learn about the attacker's motivations and objectives

Pentest goals, scoping and planing

Learn about defense and how to thwart it

Design and use exploits and payloads

Know the other attacker's tools

Conduct complete pentests

CISSPs know all this: *Access control; Telecommunications and network security; Information security and risk management; Application security; Cryptography; Security architecture and design;* Operations security; Business continuity and disaster recovery planning; Legal, regulations, compliance and investigations; Physical (environmental) security.

LEARN SOME HISTORY FIRST

We've grown from the MULTICS years

- Saltzer and Schroeder audited the MULTICS code for security bugs in 1975.
- Within the military:
  - *Computer Security: the Achilles' heel of the electronic air force*, Schell ,1979.
  - The orange book, 1985.

- Worms – the RTM worm hits in 1988.
- Hacking = adding a line in /etc/passwd.
- Enter firewalls (1989? or around that time).
- Go hack yourself!
  - Improving the security of your site by breaking into it (1993) - Dan Farmer, Wietse Venema.

I. Arce's USENIX SECURITY `06 keynote talk/slides.

HACKERS CAN TURN YOUR HOME COMPUTER INTO A BOMB

. . . & blow your family to smithereens!

By RANDY JEFFRIES / *Weekly World News*

WASHINGTON — Right now, computer hackers have the ability to turn your home computer into a bomb and blow you to Kingdom Come — and they can do it anonymously from thousands of miles away!

Experts say the recent "break-ins" that paralyzed the Amazon.com, Buy.com and eBAY websites are tame compared to what will happen in the near future.

Computer expert Arnold Yabenson, president of the Washington-based consumer group National CyberCrime Prevention Foundation (NCPF), says that as far as computer crime is concerned, we've only seen the tip of the iceberg.

"The criminals who knocked out those three major online businesses are the least of our worries," Yabenson told *Weekly World News*.

"There are brilliant but unscrupulous hackers out there who have developed technologies that the average person can't even dream of. Even people who are familiar with how computers work have trouble getting their minds around the terrible things that can be done.

"It is already possible for an assassin to send someone an e-mail with an innocent-looking attachment connected to it. When the receiver downloads the attachment, the electrical current and molecular structure of the central processing unit is altered, causing it to blast apart like a large hand grenade.

"As shocking as this is, it shouldn't surprise anyone. It's just the next step in an ever-escalating progression of horrors conceived and instituted by hackers."

Yabenson points out that these dangerous sociopaths have already:

- Vandalized FBI and U. S. Army websites.
- Broken into Chinese military networks.
- Come within two digits of cracking an 87-digit Russian security code that would have sent deadly missiles hurtling toward five of America's major cities.

"As dangerous as this technology is right now, it's going to get much scarier," Yabenson said.

"Soon it will be sold to terrorists cults and fanatical religious-fringe groups.

"Instead of blowing up a single plane, these groups will be able to patch into the central computer of a large airline and blow up hundreds of planes at once.

"And worse, this e-mail bomb program will eventually find its way into the hands of anyone who wants it.

"That means anyone who has a quarrel with you, holds a grudge against you or just plain doesn't like your looks, can kill you and never be found out."

KABOOM! It might not look like it, but an innocent home computer like this one can be turned into a deadly weapon.

Sickos can wreak death and destruction from thousands of miles away!

Arnold Yabenson.

Attackers are getting more pro

- Mitnick: IP spoofing & TCP hijacking (1994).
- DoS with TCP SYN flood attacks (1996).
- Buffer overflows get widely understood
  - "Smashing the Stack for Fun and Profit," AlephOne, Phrack 49, 1996.
  - "Syscall Proxying," M. Caceres. BlackHat 2002.
  - "The shellcode generation," I. Arce. IEEE S&P, Sept/Oct `04.
- Some nasty incidents:
  - "An analysis of the Slapper Worm," I. Arce & E. Levy, IEEE S&P `03.
  - Web-application worms: Santy, 2004.

See: J. Pincus, B. Baker , "Beyond Stack Smashing: Recent Advances in Exploiting Buffer Overruns," *IEEE S&P, Jul/Aug `04*

A definition

A pentest is

a time-constrained effort

to evaluate the security of a target

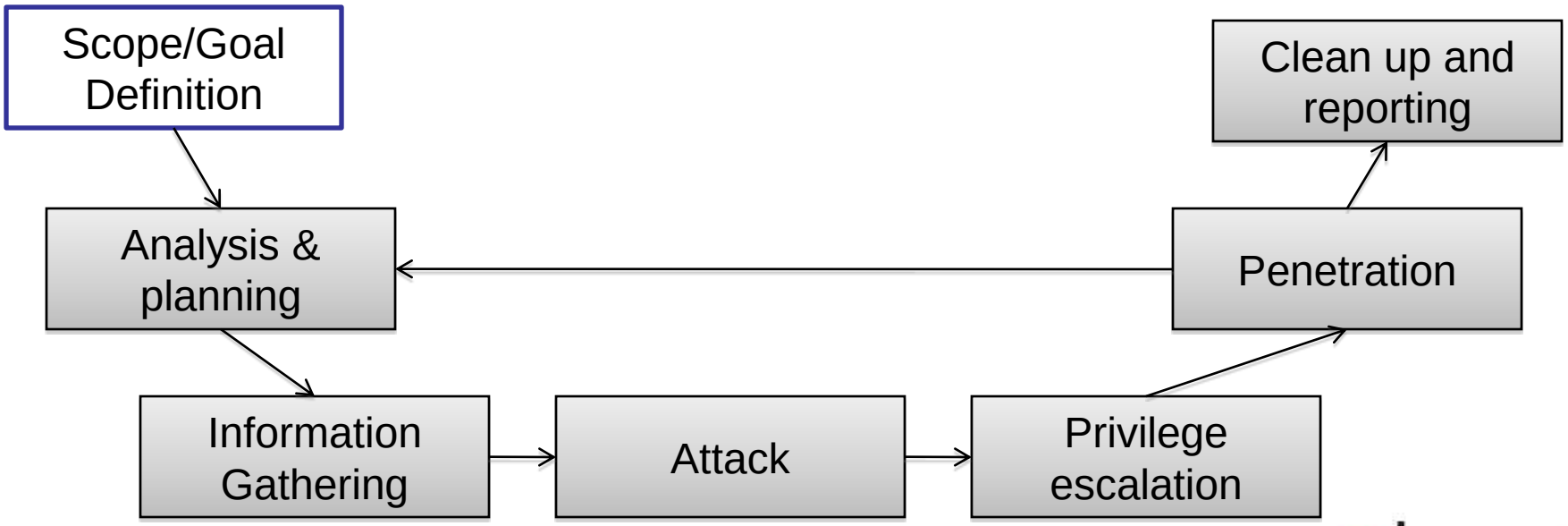
using the enemy's approach

Much more than listing vulnerabilities

- Scoping
  - Define an objective.
  - Fix an adversarial model.
- Executing:
  - Plan, execute the plan doing only what's necessary & gain access to important assets.
  - Actions mimic attackers' behavior, but must be robust.
  - Document all that's done.
- Reporting
  - Presenting detailed technical info & risk analysis.

Scoping

- The contract & the client
  - Is in the finance space,
  - Wants to have a perimeter audit,
  - Will pay for two weeks time & 1 pen tester,
  - Is specially concerned about data loss in an internal server and some employee's workstations.

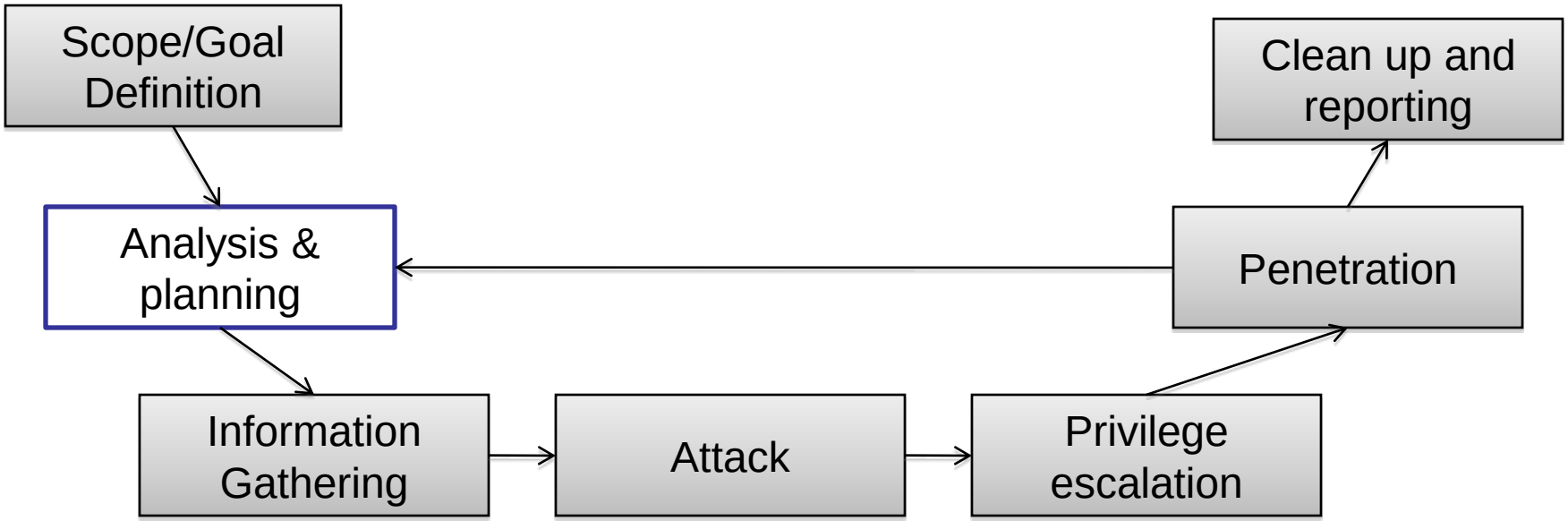


Gera Richarte "Modern Intrusion Practices," BlackHat `03

Planing

- The plan is to tackle internet-facing servers first.
  - Next, attempt to pivot inside.
  - Use new exploits!

- Let's see how to do this in the next slide…

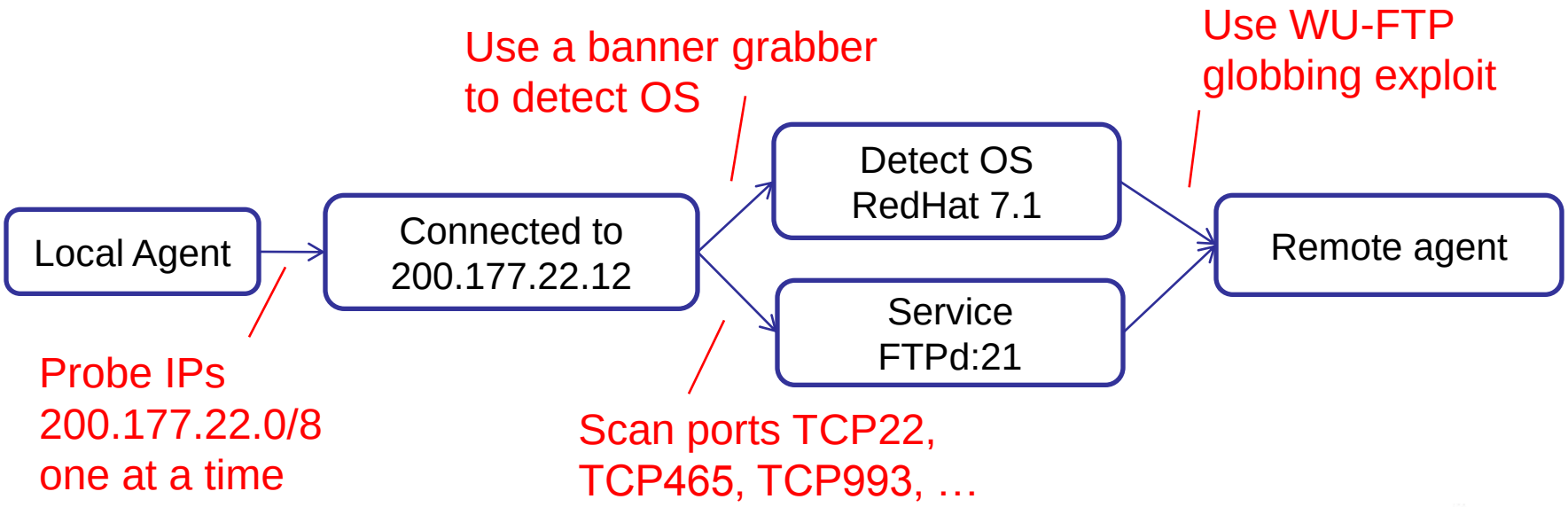


Futoransky, Notarfrancesco, Richarte, Sarraute, "Building Computer Network Attacks," 2003

Plan carefully to avoid spurious actions

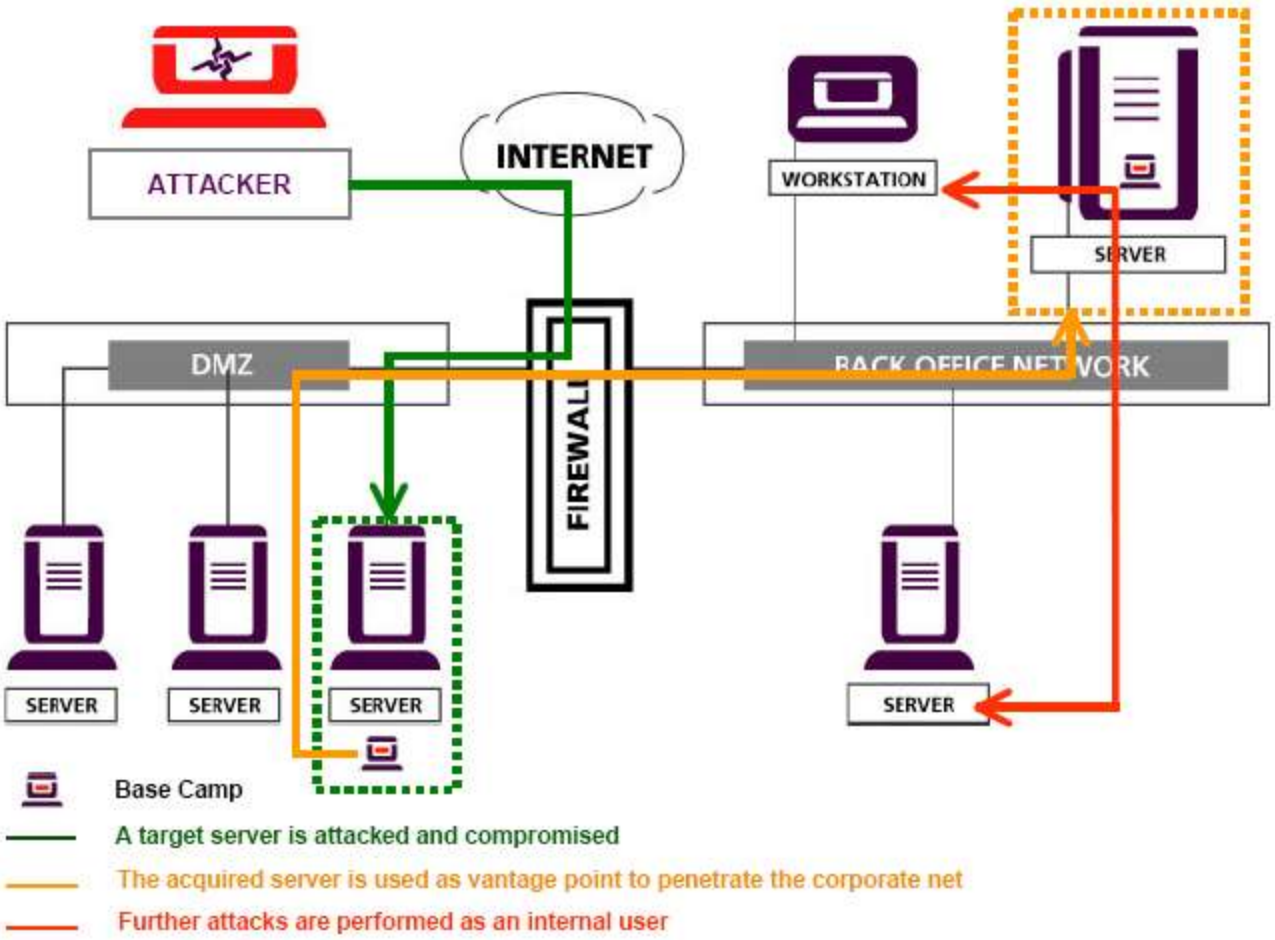
- Most of what happens in a pentest can be described by **actions** that are used for obtaining **assets**.
  - We will only execute an action if we can profit from the result.
  - The result of an action depends on the scenario characteristics, the already acquired assets (and probably a probabilistic factor).

- A plan is a path in a graph describing a sequence of actions that must be consecutively executed to get the asset “remote agent” in one of the target machines.



Futoransky, Notarfrancesco, Richarte, Sarraute “Building Computer Network Attacks,” Corelabs Technical Report. 2003.

Something like this should work!

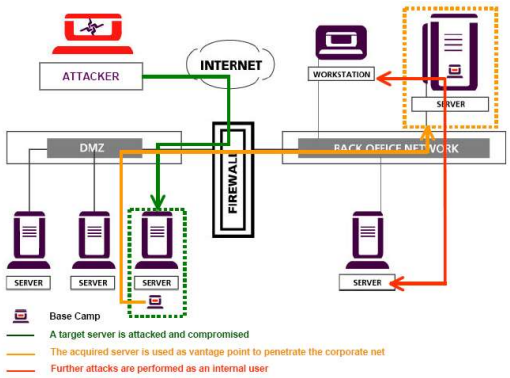


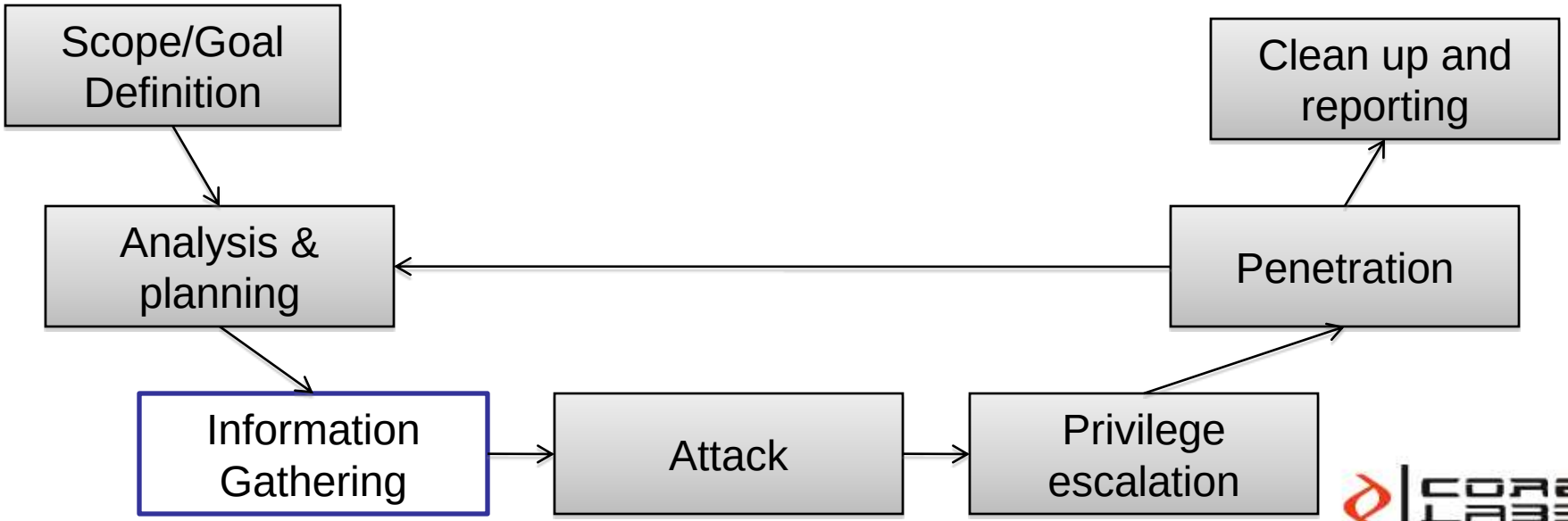
Notes:

Although this attack is not likely to succeed today, it makes for a nice example.

IG

- After some Intelligence gathering (passive) we find web servers, name servers and mail servers.
- We use information gethering (passive & active) tools to detect OSs and open UDP & TCP ports,
  - looking for one computer at a time.
  - only for known vulnerable services that we can exploit.
- The 2nd one is a RedHat 7.1 with TCP:21 open.

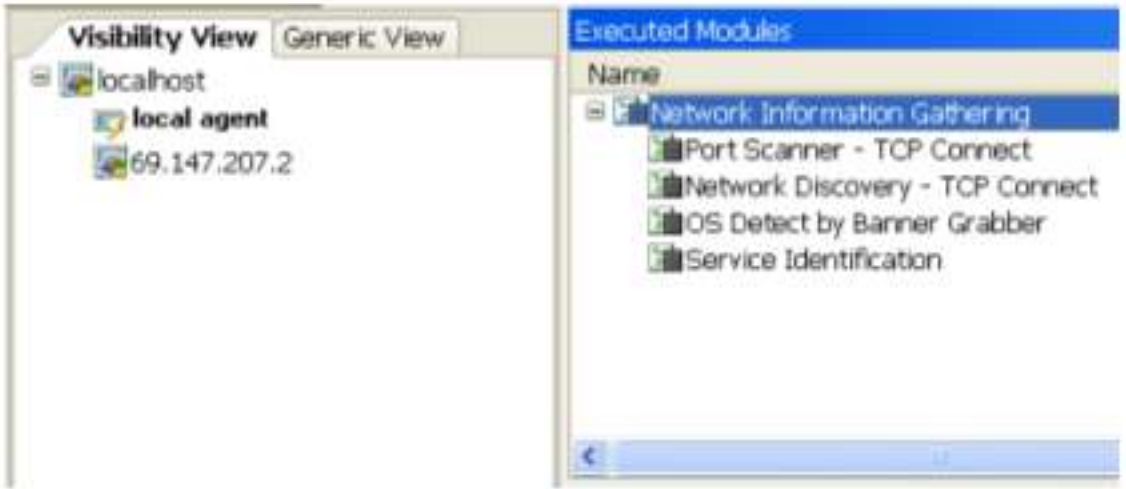




IG in details

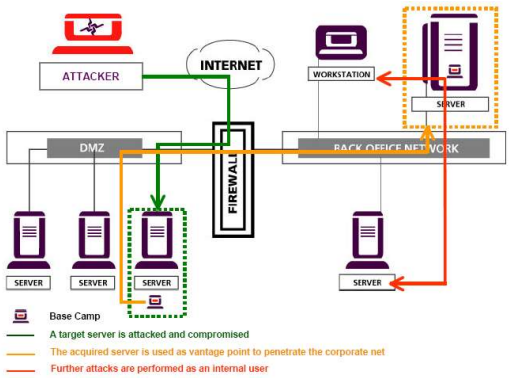
- The user chooses one of a few tools for each step. All data that's found is combined and viewed by the user.

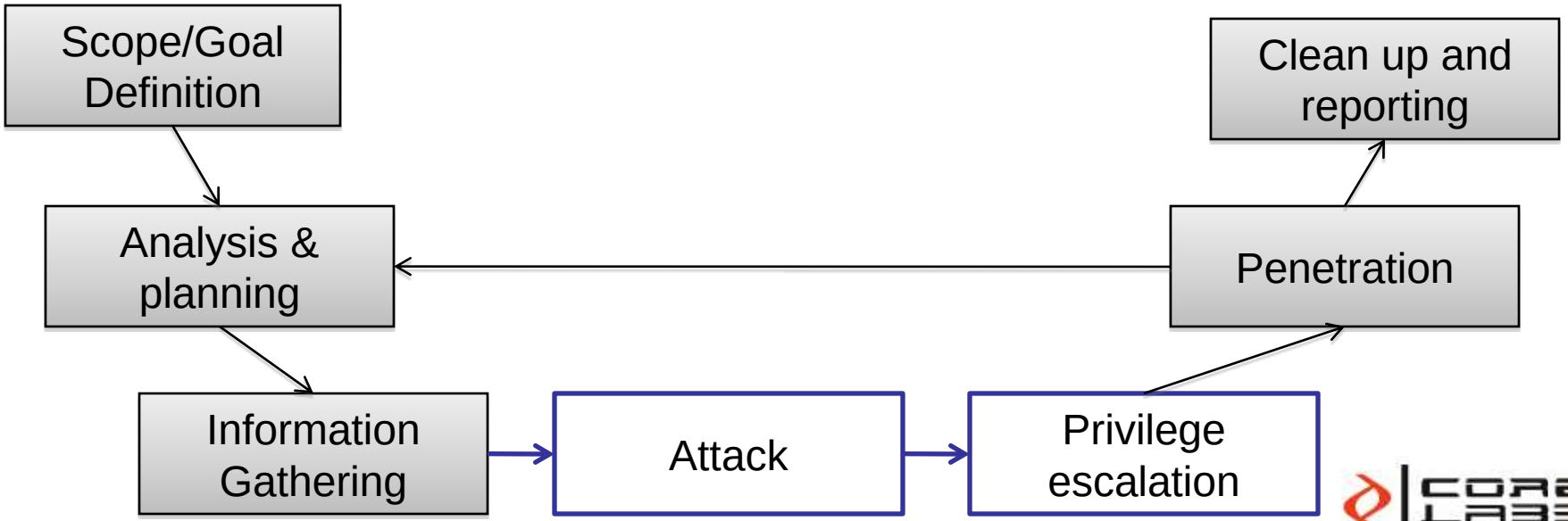
- Some Information gathering tools included are:
  - Network discovery: ARP, TCP SYN packets, ICMP echo request, TCP connect and passive discovery.
  - Port scan.
  - OS fingerprinting: Nmap OS stack fingerprinting, a home-brew OS fingerpringting by neural networks tool.
  - Or simply use one of GFI LANguard, Qualys, nmap, Nessus, PatchLink STAT, and eEye Retina and import the results!



Attack & privilege escalation

- We can detect vulnerabilities now!
  - Say, because we know that RedHat 7.1 had a vulnerable wu-ftp by default and port 21 is open.
- For each potential vulnerability discovered in IG, we send an exploit.
  - We send the exploit and wait for the exploit to work.
  - In some cases, we might need to do some local IG and execute a second exploit to gain root privileges.





An exploit is code that adds features

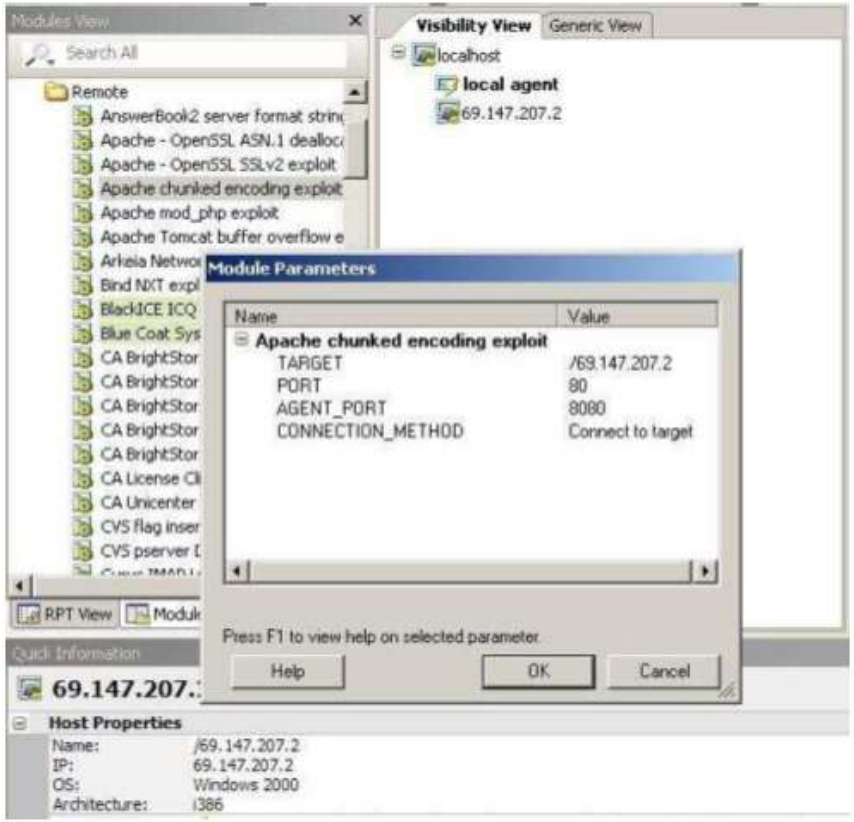
- Sometimes applications are developed or designed with so-called security bugs.

- This means that attackers/users are provided with a feature that wasn't there by design. For example,
  - A buffer overflow will typically give the attacker the ability to execute arbitrary code in the compromised system.
  - A SQL-injection vulnerability will give the attacker the ability to execute queries in the underlying web application.
    - Sometimes this will allow the pentester to compromise the server in a second step.

Gera Richarte "Assembly for Exploit Writing," course.

Exploits are around the corner!

- As of today, there are several means to get exploits:
  - Develop them on your own,
  - Get a pen-testing product (Metasploit, Impact, Canvas, …),
  - Buy them/download them on the internet.

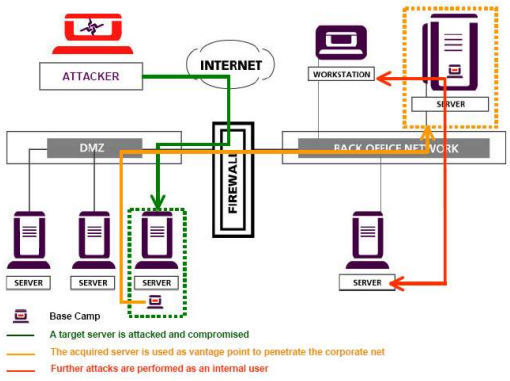


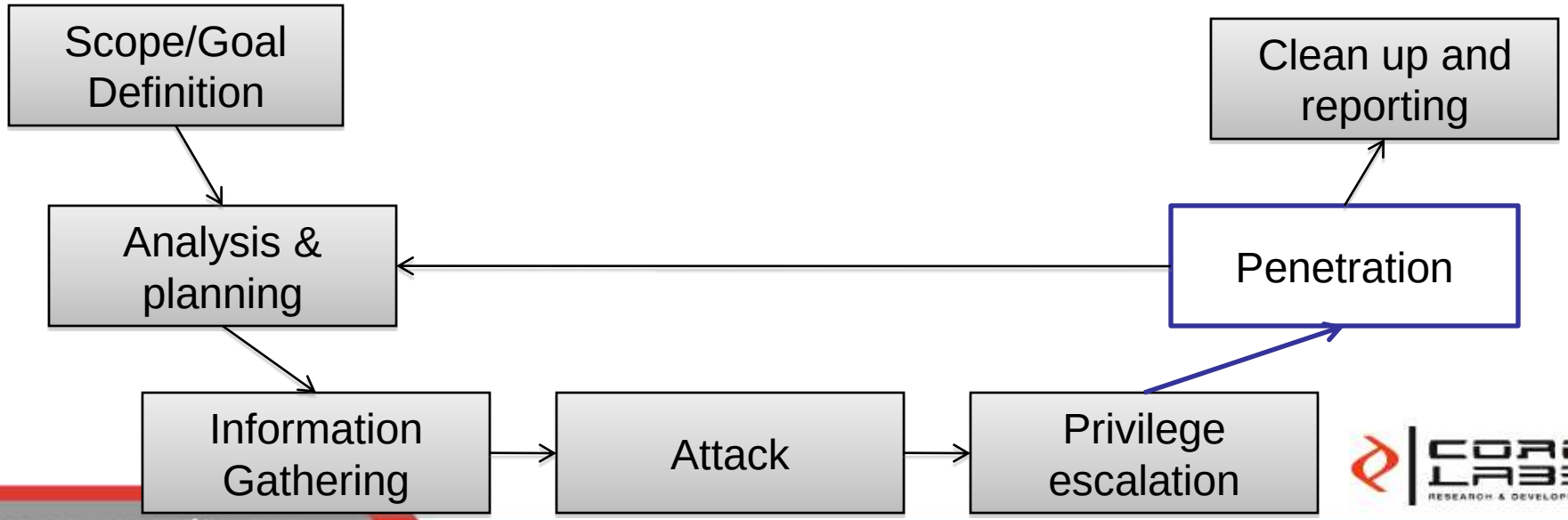
Gera Richarte “Assembly for Exploit Writing,” course.

Penetration

- The exploit is followed by a payload or “egg” that installs an agent in the compromised system.

- We use the “(syscall proxying) agent” that will provide:
  - a stealth connection between agent and pen tester (I.e., much like a rootkit),
  - “shellcode functionality,” and
  - the ability to chain new agents.





Advances in payload engineering

- Pen testers must leverage exploits code to profit from vulnerabilities:
  - To this end they use the “new features.”
  - For example, buffer overflows allow the pentester to inject code into a new/running process. The code, is called an egg.
  - Likewise a SQL-injection attack will allow the pentester to execute ~arbitrary SQL commands.

- Mastering these “egg injections” is what we call payload engineering
  - rx (read and exec).
  - Loader payloads: *MOSDEF (Canvas), InlineEgg (Gera)*.
  - *Meterpreter*, *VNC server, dll injection tool (Metasploit).*
  - polymorphic & factorized payload (K2).
  - Syscall proxying (Core).

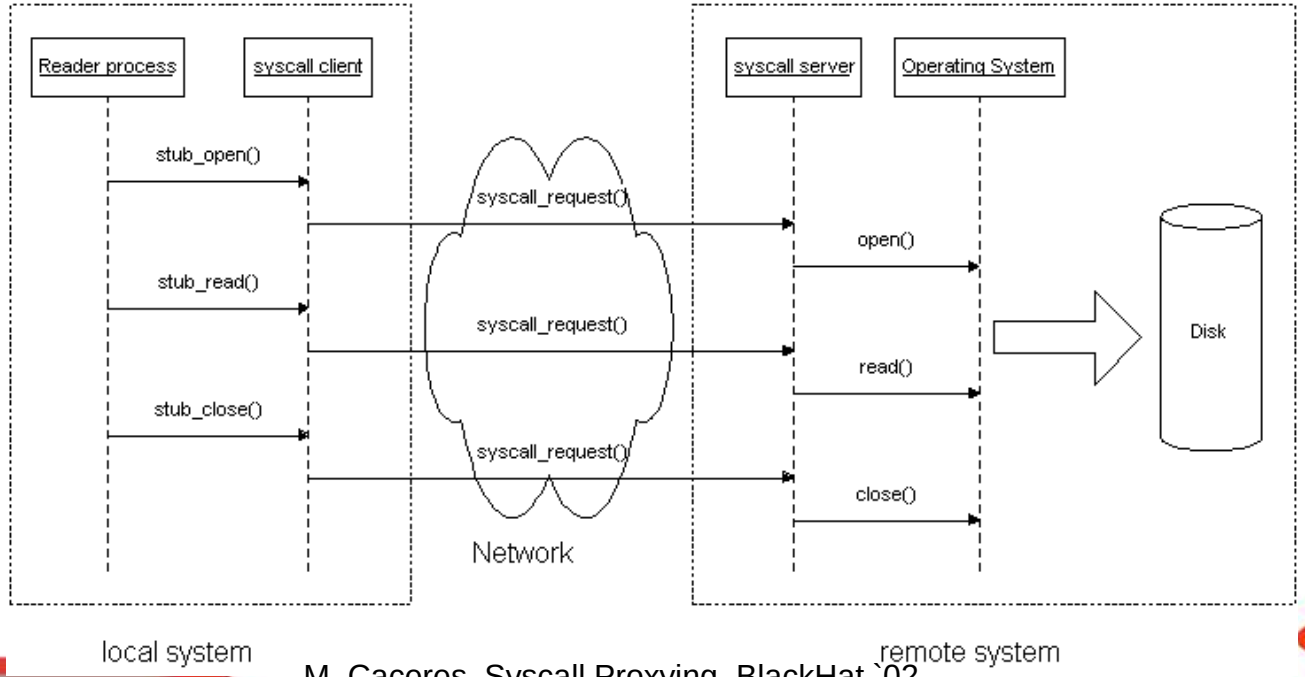
See: *A. Waissbein,* “Strong Payload Obfuscation and Encryption,“ PacSec 2006

Syscall proxying

- System calls or syscall are instructions issued by applications to the OS.
  - Typically through libraries, such as libc.

- A syscall proxy mechanism enables a syscall client (running in one computer) to send syscalls to a syscall server (running in a 2nd computer) that executes them and returns answers.

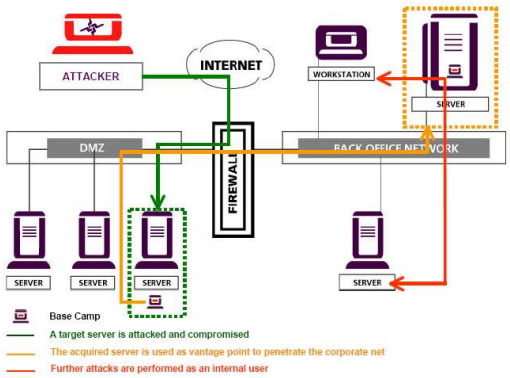
- It works as if the penetration tester were executing code in the remote computer, but he really is in the local computer.

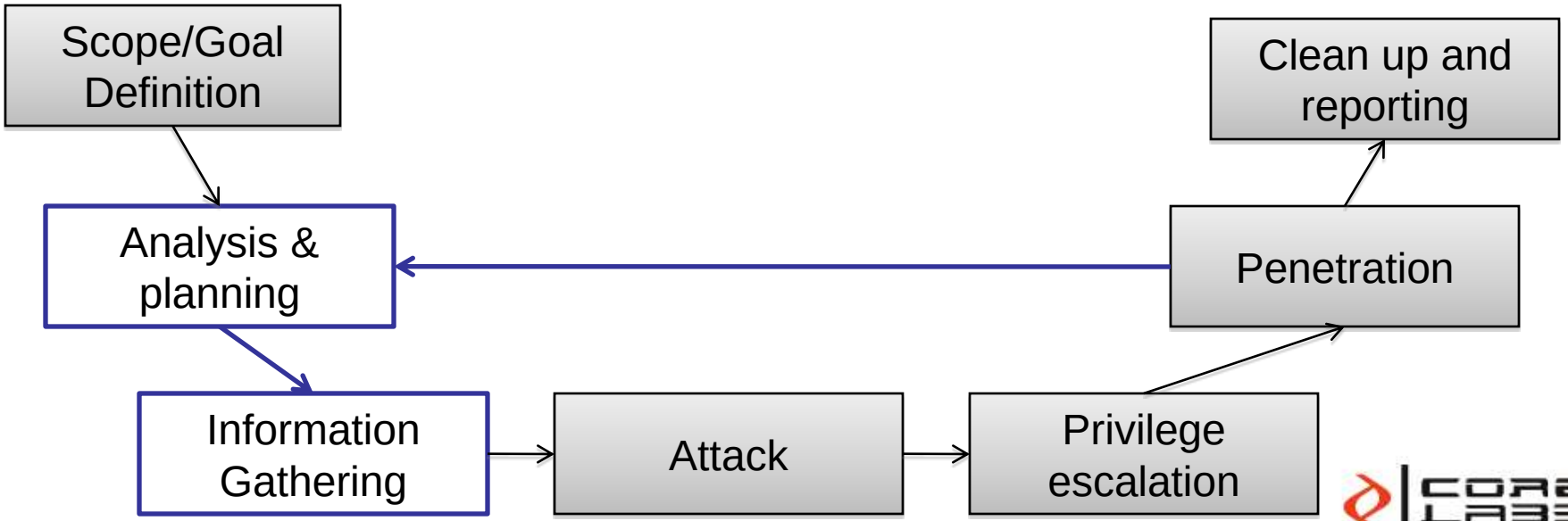


M. Caceres, Syscall Proxying. BlackHat `02.

Next, we pivot and restart the cycle

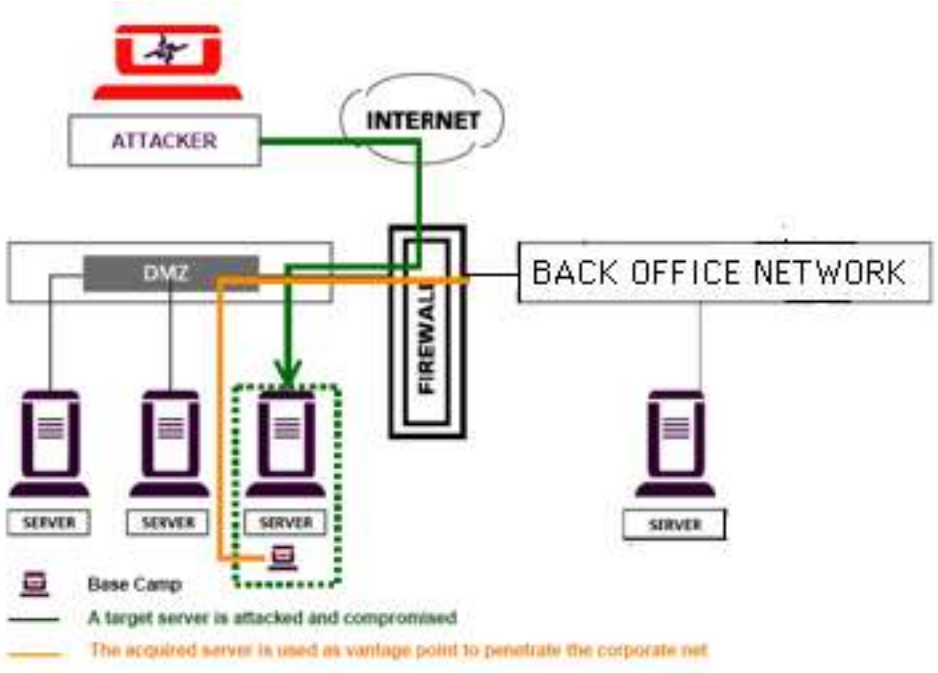
- The pen-tester has a new base camp!
- Continuing with the plan, he searches for internal servers.





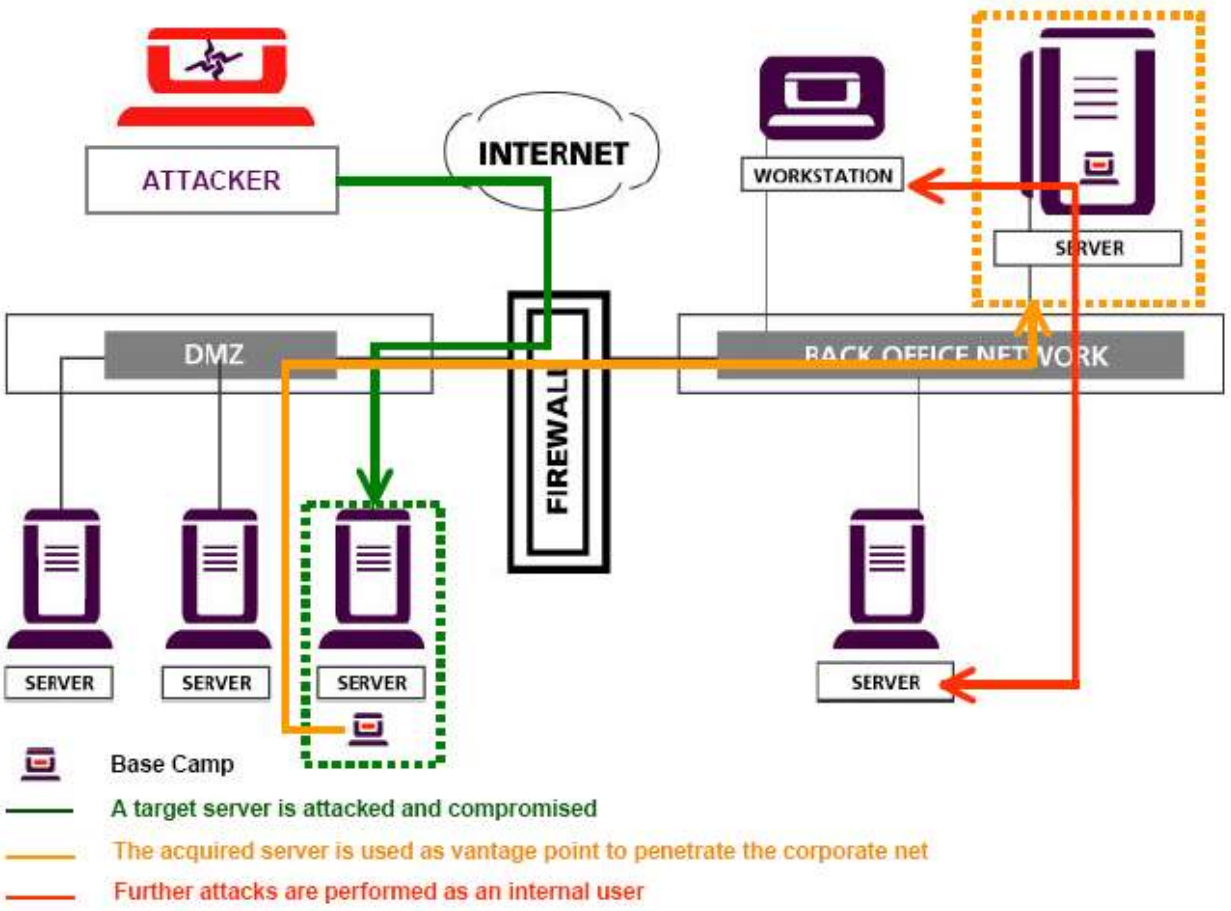
The pentester's view is enlared

- Continuing with the plan, he searches for internal servers.
- Say, by sniffing traffic he discovers a mail server on the other side of a firewall.



Et cetera

- You can imagine how this should continue…
  - The lesson, again, is think before you act.



About the report

- The report will include
  - A list of findings prioritized according to the risk (as perceived by the pen-tester and taking into account the scoping).
  - A log of all what the pen-tester did (including compromises, accidents & crashes,…).

- To prove that he broke in the different computers the pen-tester can:
  - Capture screenshots for workstations.
  - Describe the explored network.

STUDENTS DISMISSED

Closing remarks (for students)

What have students learned here?

1. Performing standard IG.
2. Detecting vulnerabilities and using exploits.
3. Pen-testing tactics:
   - Pivoting.
   - Think before taking an action.
4. Network security
   - The three-legged DMZ configuration.

Closing remarks (for teachers)

What have students learned here?

1. By reading the module logs teacher can check whether students used the right tools with the correct parameters.

2. Test the students' ability to plan, e.g., did they perform unnecessary actions.
   - Let's say we change firewall rules and network topology to another (standard) scenario: can they guess the network topology and figure out how to make the pentest?

3. Did they understand the essence of pivoting?

4. By understanding the students' performance, the teacher can identify their weaknesses as pentesters and plan new exercises to work on these.

So why is our learning kit necessary?

- But all this cannot be done in a small virtualization lab/ farm! : *Hardly!*
- With our tool all students can do this at once!
  - we require only one computer per student.
  - in case the network / computers hangs, the simulation environment can be easily reset.
- And there's the problem of configuring new scenarios.
  - larger or more with more complex topologies.
- Last: students must be evaluated.
  - success, performance, stealth, quality of reports.

QUESTIONS?

SIMULATING COMPUTER ATTACKS - REALISTICALLY

The tool is aimed at simulating network attacks

- A preliminary version is available, it runs on Win XP and Vista.
  - Tradeoffs can be made to simulate 2k to 13K.

- It contains two main components:
  - A trimmed & modified version of *Core Impact* Pen-testing Framework.

    which thinks it is connected to a real network when actually it connects to:
  - A network attack simulator that we call *Core Insight*.

It can simulate a wide variety of scenarios

- Simulated scenarios are composed of computers & network devices.
  - Each system can run processes and connect to other devices.
  - We simulate Windows workstations and servers, many Unix systems, routers and firewalls.
  - Each is configured independently according to the triple:
    - OS, services, file system.
    - Network connections to hubs, switches, dial-up connections.
    - Vulnerabilities.
- In the current version, scenarios are generated by scripts.

Simulation of sockets and syscalls

- Agents are controlled by the pen-tester, e.g., from Impact.
  - There's always a local agent + agents for compromised computers.
- Sockets are simulated. We support TCP and UDP sockets.
  - We sort of wrapped a real BSD socket to do this.
  - Raw packets and other more granular network operations are not supported.
- Each (simulated) host is represented by a simulated syscall server.
  - So that the "controlling agent" will issue a syscall for the simulated host (through thea standard interface), and receive its answer.
  - Each system has at least a simulated thread (real thread for the host).
  - Most of the syscalls are supported (but some are not).

Simulation tradeoffs and the reality factor

- File systems simulation is supported.
  - But to boost efficiency all systems are configured with a template file system by default.
  - When the template is modified, a separate file system is simulated.

- Some actions are only emulated (not simulated).
  - For example, exploits are emulated: when an agent is commanded to send an exploit, our simulator will look up for the outcome in a DB, e.g., ApacheChunkEncodingExploit yields compromise if the system should be vulnerable.

- All actions have a probabilistic nature,
  - E.g., ApacheChunkEncodingExploit against a RedHat 7 will work with %80 probability, crash the system with %5 and do nothing with %15.
  - We have pre-computed probability distributions according to experimentation with our model.
  - The simulation kit has a DB that stores this information.

pros & cons

PROS

- This system provides a very good tradeoff between the *simulation level required to produce a pentest* and the *hardware and setup requirements*.
  - It requires less hardware than other solutions, e.g., can simulate *more* systems with the same hardware.
  - Setup of scenarios is much easier, as well.
- The “module log” contains a list of all the modules executed and its results.
  - this provides a tool for grading students.
- Support for hypothetic vulnerabilities.

CONS

- There is a divergence between reality and simulation, e.g.,
  - raw packet is not implemented.
- The current version lacks support for client-side, wireless, voip, scada, …

QUESTIONS?

MORE LESSONS

The first lesson should be a glimpse of the methodology, and the learning kit's capabilities.

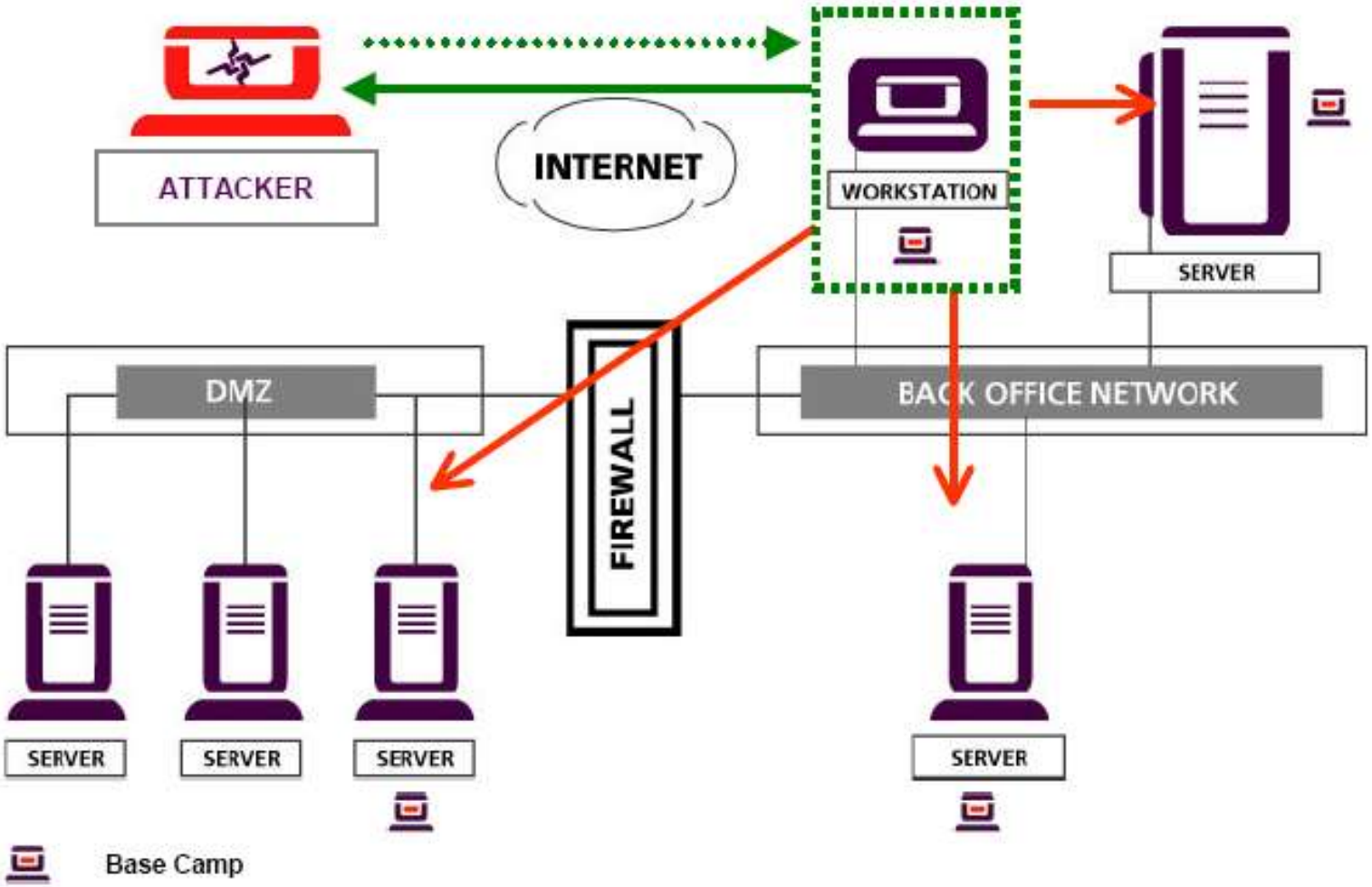
The teaching program

- We want to complete this first lesson into a complete course.

- New lessons go in two directions
  - Scenarios get more complicated (with network segmentation and more firewalls, IDS detection that closes connections under noisy attacker activity, …).
  - No more abstract tools, as students must understand tools' to use or modify them.

- Plus
  - Teaching exploit writing & engineering.
  - Crypto protocols (e.g., I gained control of a box and can sign packets, what can I do?).
  - New attack vectors are explained (client-side, application, wireless, etc).

Going after the workstation first



Other exercises are performed outside the kit

- Exploits
  - Writing your own exploits (binary, application,…)
  - Testing exploits: exploits should be reliable.
    - Students can use a small virtualized farm with a handful of OS configurations to this end.
- Getting to know the tools
  - Sometimes in a real pen-test one needs non-standard tools, or modifying known tools.
- Hybrid support
  - Future versions of our kit will support hybrid simulation were simulated computers are connected to real computers.
  - An immediate application of this is webapps pen-testing with a real webserver connected to a simulated network.

QUESTIONS?

OTHER APPLICATIONS OF THE NETWORK ATTACK SIMULATOR

A research tool 1/2

1. Research automatic attack planning.
   - Using the kit one could device planning algorithms that execute pen-tests against input scenarios and optimize them for… performance, stealth, etc.

2. Iterative risk assessment:
   - Let's say we make a pen-test against a real scenario using Impact and succeed.
   - "Import" all the discovered information to the simulator.
   - Loop until you feel safe:
     - Modify the simulated network.
     - Attack it.

A research tool 2/2

3. Generating logs (e.g., for training log analysis & IDSs).
    1. Say, by executing several attacks against a given scenario.

4. Montecarlo analysis of different scenarios against a given attack (or attack family).
    1. We fix the scenario and vary on the attacks.

IN CLOSING

ANY QUESTIONS ?

MANY THANKS

E-mail:
ariel.waissbein/\at/\coresecurity.com

Bibliography

- *Multics Security Evaluation: Vulnerability Analysis,* Karger, Schell *(Air Force Electronic Systems Division, 1974) http://csrc.nist.gov/publications/history/karg74.pdf*
- *The Protection of Information in Computer Systems*, Saltzer and Schroeder (1975). http://www.cs.virginia.edu/~evans/cs551/saltzer/
- *Computer Security: the Achiles heel of the electronic air force,* Schell (1979). http://www.airpower.maxwell.af.mil/airchronicles/aureview/1979/jan-feb/schell.html
- *Trusted Computer System Evaluation Criteria (aka, The orange book),* National Security Institute - 5200.28-STD (1985). http://nsi.org/Library/Compsec/orangebo.txt
- *The Internet Worm Incident*, G. Spafford (1989). Technical Report CSD-TR-933, 1989. http://homes.cerias.purdue.edu/~spaf/tech-reps/933.pdf
- Improving the security of your site by breaking into it (1993) - Dan Farmer, Wietse Venema. http://www.porcupine.org/satan/admin-guide-to-cracking.html
- Syscall proxying, Max Caceres (2002). Black Hat Briefings 2002, Las Vegas.
- *Modern Intrusion Practices*, Gera Richarte (2003). Black Hat Briefings 2003, Las Vegas.
- Building Computer Network Attacks, A. Futoransky, L. Notarfrancesco, G. Richarte and C. Sarraute (2003). http://www.corest.com/files/attachments/Futoransky_Notarfrancesco_Richarte_Sarraute_NetworkAttacks_2003.pdf

Bibliography (2)

- *Simulating computer network attacks*, F. Miranda, J. Orlicki, C. Sarraute. AST `07 in JAIIO `07, Mar del Plata, Bs. As., Argentina.
- *Zombie 2.0*, D. Tiscornia and F. Russ. HACK.LU 2007.
- *Nmap*, Fyodor. http://nmap.org. 1997.
- *OS fingerprinting by neural networks*, J. Burroni and C. Sarraute, PacSec 2004.
- *Beyond Stack Smashing: Recent Advances in Exploiting Buffer Overruns*, Jonathan Pincus and Brandon Baker. *IEEE Security & Privacy, July/Aug., pp. 20−27. 2004.*