Testing Exploits and Malware in an isolated environment

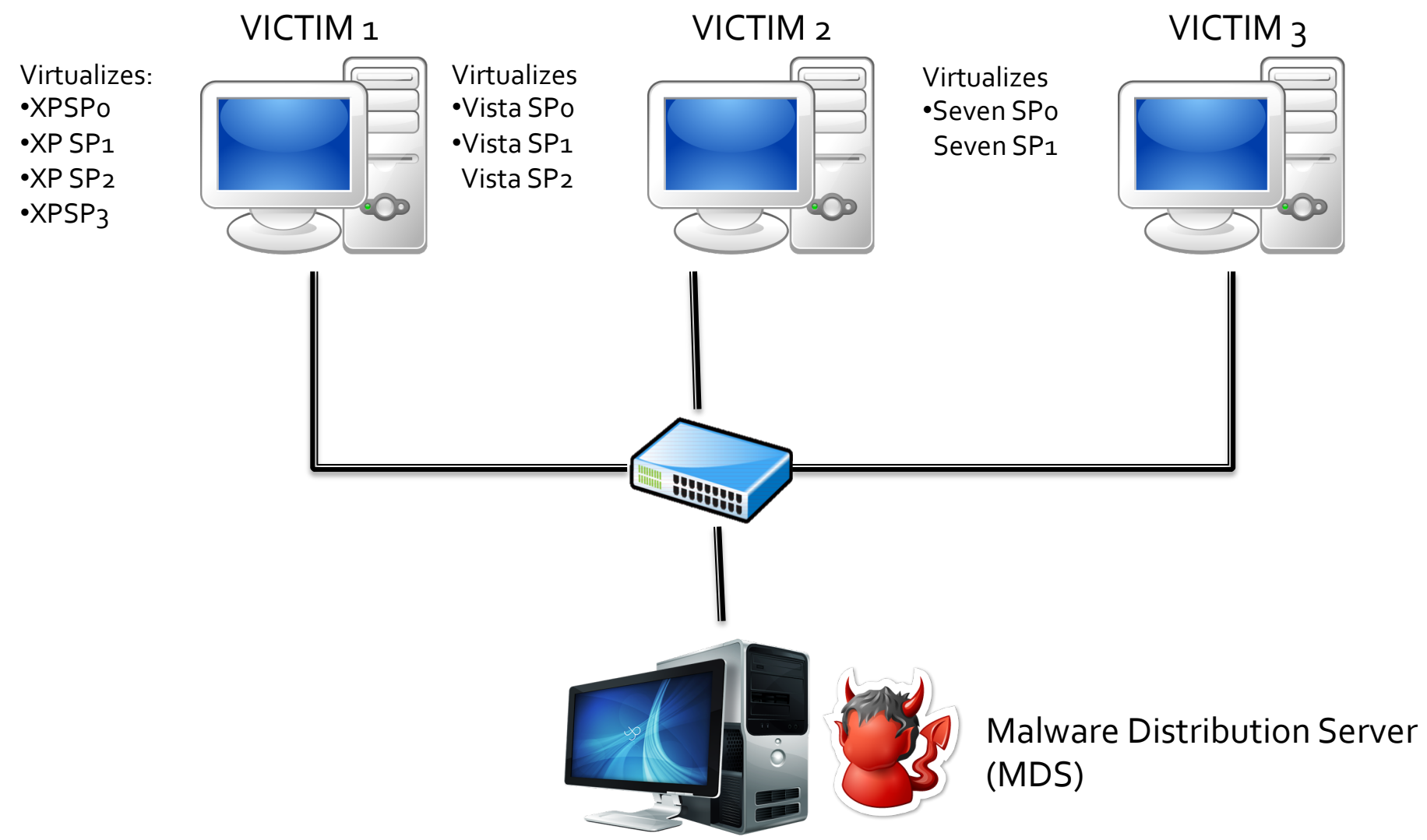
The MalwareLab

Luca Allodi – luca.allodi@unitn.it
Fabio Massacci – fabio.massacci@unitn.it
Vadim Kotov (now @ Bromium Inc., Cupertino CA) – luca.allodi@unitn.it

The MalwareLab

- Laboratory to measure malware as a "software artifact"
  - Does the malware/exploit work?
  - Under which circumstances?
  - How does it perform under different assumptions?
- Disconnected from the network
- At the moment located in Povo2, Floor 1
- Soon to be moved and renovated

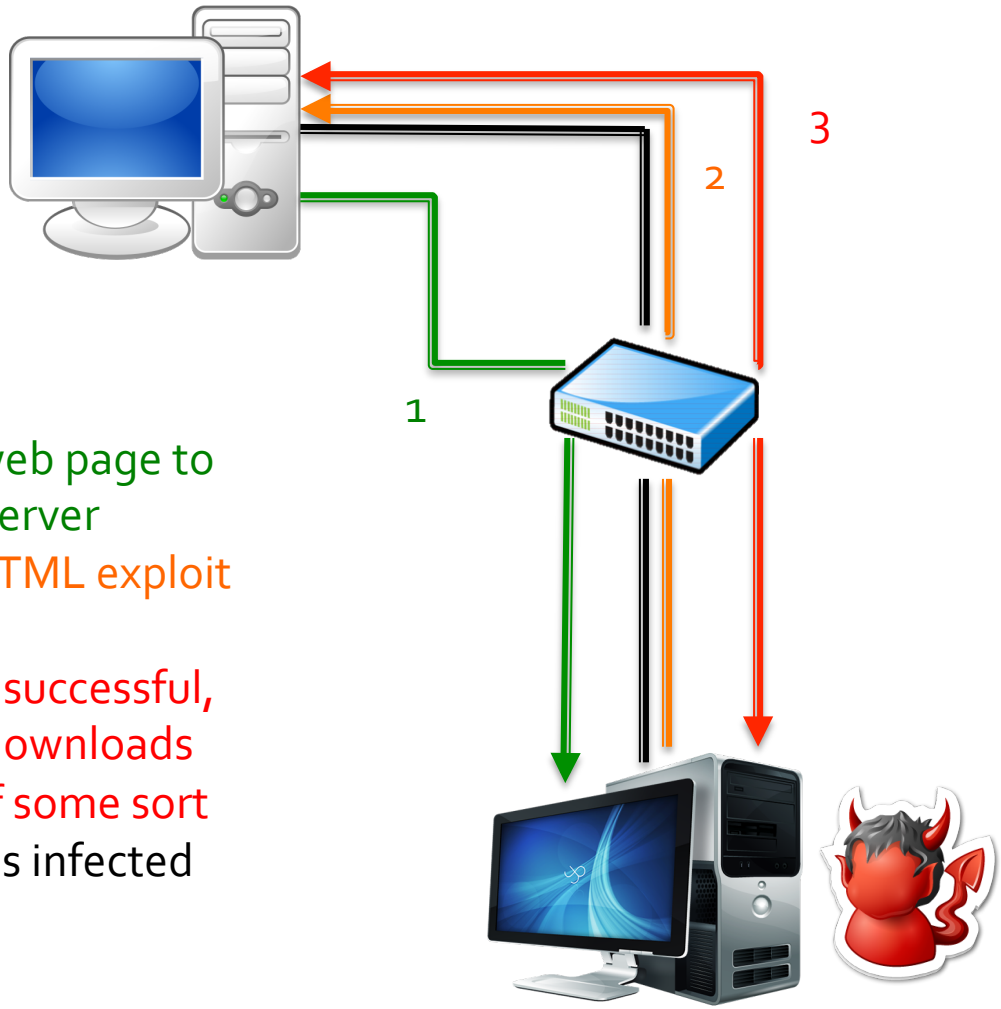
MalwareLab structure



MalwareLab functionalities

- Python infrastructure
- Automatically operate on Virtual Machines
  - Create, delete, restore VM Snapshots
- Automatically install and verify software configurations on the VMs
  - Configuration file contains list of software
  - Script pushes the software on VM, lunches silent install
  - Possibility to verify the install with a batch file
  - Firefox, Opera, Java, Quicktime, Flash, Adobe Reader
- Automated mechanism to verify exploit successfulness.
- Fully modularized - Easy to add functionalities / software/malware

Run example: testing Exploit Kits (1)



1. Requests web page to malicious server
2. Receives HTML exploit page
3. If exploit is successful, shellcode downloads malware of some sort
4. Computer is infected

Run example: testing Exploit Kits (2)

- Question: *How resilent are cybercrime ekits to software updates?*

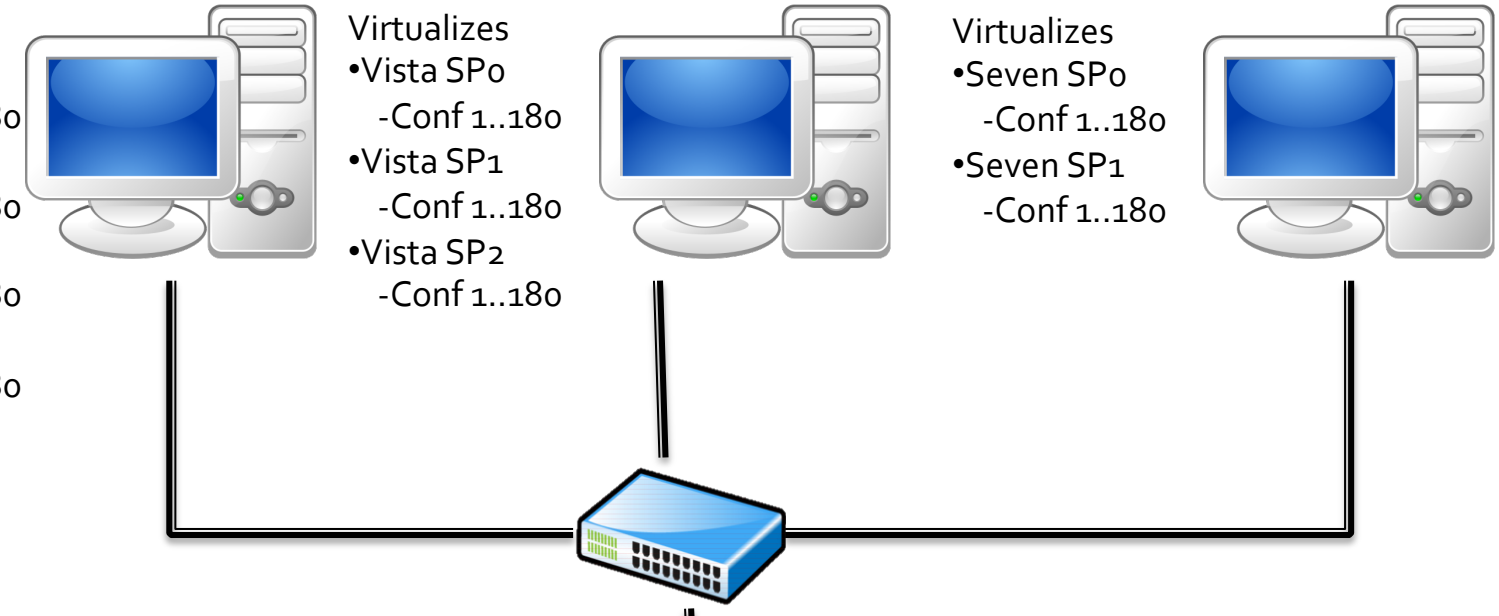
- Exploit kits span from (2007-2011)
  - How we chose the exploit kits
    - Release date
    - Popularity (as reported in industry reports)
    - CrimePack, Eleonore, Bleeding Life, Shaman, …
- Software: most popular one
  - Windows XP, Vista, Seven
    - All service packs are treated like independent operating systems
  - Browsers: Firefox, Internet explorer
  - Plugins: Flash, Acrobat Reader, Java
- 247 software versions
  - spanning from 2005 to 2013
- We randomly generate 180 sw combinations (times 9 Operating Systems) to be the configurations we test

Experiment setup (1)

Virtualizes:
- XPSP0
  - Conf 1..180
- XP SP1
  - Conf 1..180
- XP SP2
  - Conf 1..180
- XPSP3
  - Conf 1..180

Virtualizes
- Vista SP0
  - Conf 1..180
- Vista SP1
  - Conf 1..180
- Vista SP2
  - Conf 1..180

Virtualizes
- Seven SP0
  - Conf 1..180
- Seven SP1
  - Conf 1..180



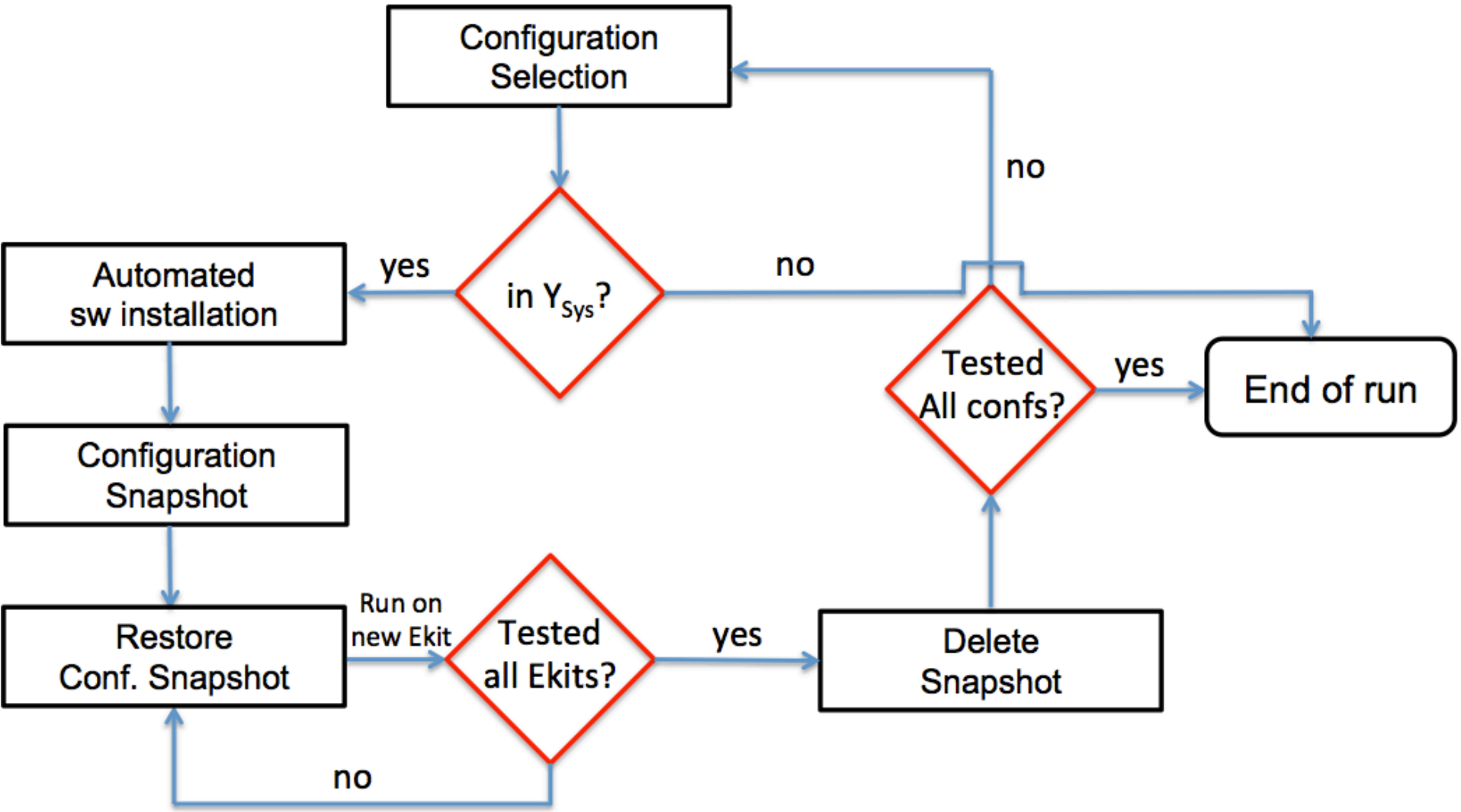
- Exploit kit 1
- Exploit kit 2
- ..
- Exploit kit 10

Malware Distribution Server (MDS)

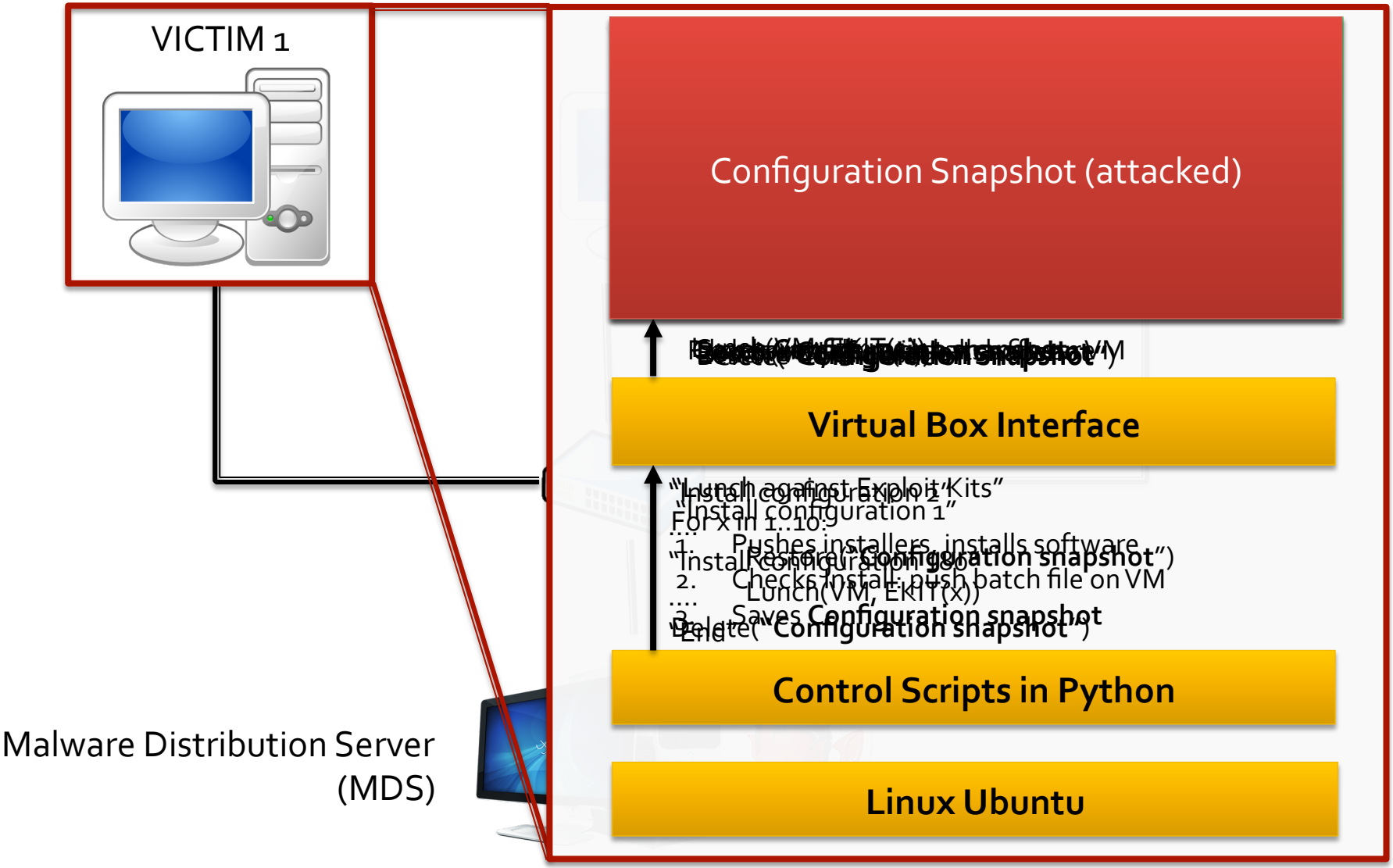
Configuration example

- One configuration for: Windows XP Service Pack 2
  - Firefox 1.5.0.5
  - Flash 9.0.28.0
  - Acrobat Reader 8.0.0.0
  - Quicktime 7.0.4.0
  - Java 1.5.0.7
- One configuration for: Windows Seven Service Pack 1
  - Firefox 8.0.1.0
  - Flash 10.3.183.10
  - Acrobat Reader 10.1.1.0
  - Quicktime: No version
  - Java 6.27

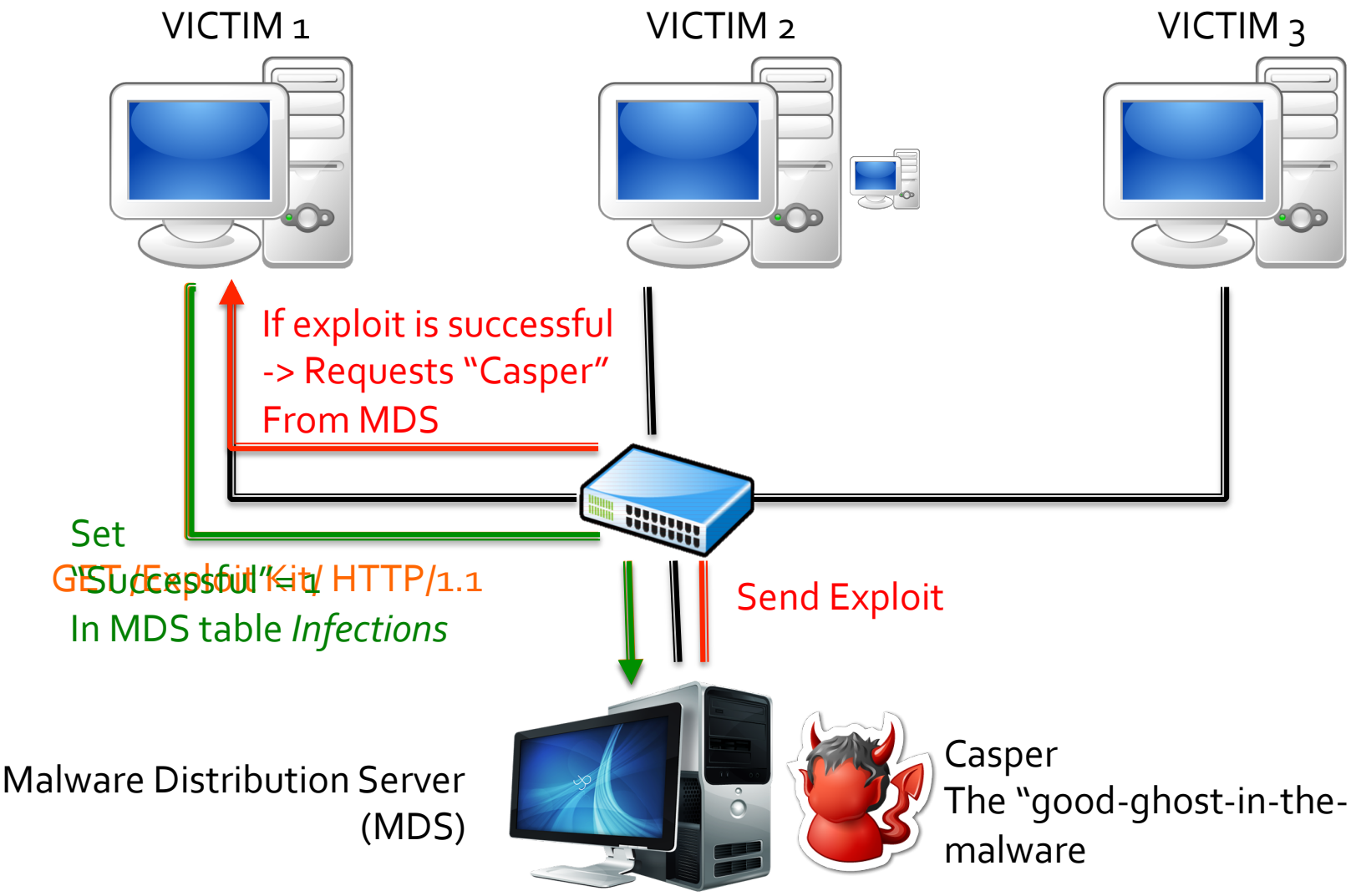
Experiment setup (2)



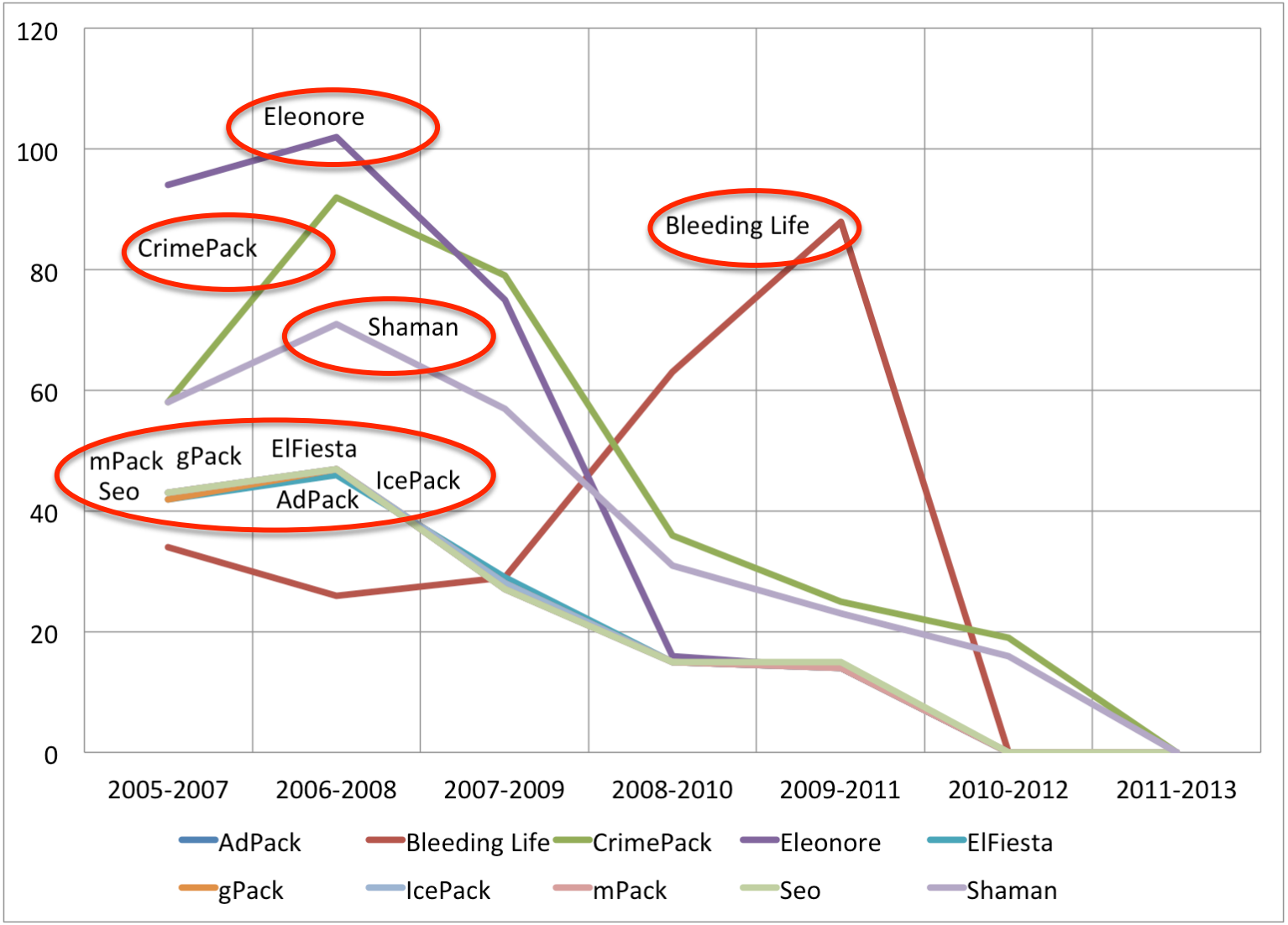
Experiment run (read: Example of MalwareLab functionalities)



Assess Exploit Successfulness



Some results



Useful Reads

- MalwareLab & Ekits:
  - *CSET '13*: MalwareLab: Experimentation with Cybercrime Attack Tools.
  - *ESSoS '13*: Anatomy of Exploit Kits - Preliminary Analysis of Exploit Kits as Software Artefacts.
- Exploitation 101
  - [BOOK] HACKING: The Art of Exploitation – Erickson
  - *Phrack Magazine*: Smashing The Stack For Fun And Profit
- Advanced exploitation
  - *Usenix '11* – Q: Exploit Hardening Made Easy
  - Blackhat 2013 - JUST-IN-TIME CODE REUSE: THE MORE THINGS CHANGE, THE MORE THEY STAY THE SAME
  - *Usenix '14* - ROP is Still Dangerous: Breaking Modern Defenses
  - *Usenix '14* - Size Does Matter: Why Using Gadget - Chain Length to Prevent Code-Reuse Attacks is Hard
  - *IEEE Symposium on Security & Privacy '14*: Framing Signals — A Return to Portable Shellcode
- Tools
  - Damn Vulnerable Linux
  - gcc, gdb
  - MalwareLab

Showtime

- Exploit kit inner workings
- Overview of an exploit
  - Acrobat Reader, CVE-2010-0188
- Demo of attack

Buffer overflow vulnerability

- Buffer overflow: a variable can grow arbitrarily big in memory
  - No control over its size
- If the attacker can control the variable, he can write into memory outside of the variable boundaries
- It is possible to hijack program execution by redirecting it to a shellcode injected by the attacker
- Shellcode can execute actions such as downloading and executing malware

Memory layout