Virtualization In Software Development And QA

Marcelo Picorelli

Quality Assurance Analyst

VMWORLD 2006

What This Talk is About

- It is about why we need to use VMware for our testing process
- It is about the problems we found and the solutions we tried
- It is not about how you have to do it; it's just about our own experience

Introduction – Application description

- CORE IMPACT
  - The first automated comprehensive penetration testing product
  - Safely exploits vulnerabilities in your network infrastructure
    - Targets several popular Operating Systems
      - Windows
      - Linux
      - Solaris
      - AIX
      - OS X
      - BSD
    - *and* their applications
  - Exploits allow an attacker to take control of the target system
  - Systems under our control are used as stepping stones for further pentesting

Introduction-*cont.*

- Exploit code provides an interface that was not intended to exist in the target
  - Prone to functionality and reliabilty problems
  - May render the target system unusable
- Thorough testing is needed to assess and minimize the risk

Introduction – Testing – three years ago

- Manual testing
  - Each exploit is manually tested to guarantee its reliability
- Growing number of exploits
  - Daily tests cannot be performed
    - Not enough hardware resources
      - Small lab
    - Understaffed, 3 people
- Automatic testing is needed

Automatic testing

- Testing types:
  - Smoke test
  - Regression test
  - Thorough test
- Product is tested daily

Automatic testing – test types

- Smoke test
  - Quickly finds coarse errors
    - Misplaced files
    - Wrong CVS commits
    - Typos in code, *sefl* instead of *self*
- Regression test
  - Finds more subtle errors
    - Verifies that what worked yesterday works today
- Thorough test
  - Tests every single exploit with every default parameter combination

Automatic testing – in the beginning

- Had to be done in python language
  - Every module in IMPACT is python based, except for the GUI
- Had to be done within 24hs
  - Risk of error propagation increases with time
- Had to be done with low budget hardware
  - In *ancient* times we couldn't afford new hardware
- Development of PoC tools
    - First milestone: 10 automated tests

In the beginning - Hardware

- Small company, small budget
  - Only spare hardware available
    - AMD 800Mhz 512Mb 80GB HDD
    - Linux based
  - Was my old and previous desktop PC, “*empowered*” by salvaging spare parts

In the beginning - PoC

- PoC must run *inside* CORE IMPACT
  - It is like any other module
  - No VMware API available in python
    - Wrapped VMCom API
      - Python win32api extensions
- First PoC runs fairly well
- New milestone
  - 18 thorough tests

What is a thorough test?

- A thorough test in our context is
  - One exploit executed against:
    - All the available platforms
    - All the available applications
  - Using all default parameters
    - Exploits are parameterizable
    - Usually using at least three possible connection methods:
      - Connect To
      - Connect From
      - Reuse connection

What is a thorough test? *cont.*

- Minimal test
  - One platform (Windows 2000)
  - One application
  - One parameter (connect to)
  - Rarely used

- Typical test
  - Several platforms (Windows 2000, XP)
  - One application
  - Three connection parameter values (to, from, reuse)
  - Summing up 6 runs
  - Usually this is the *real* “minimal” test

First stage

- 18 Tests goal
  - Roughly 60 runs

- Needed support
  - VMImages information
    - Platform
    - Installed applications
    - Image location (for start up)
  - XML used as repository
    - Standard
    - Quick and flexible for explorative development

First stage - XML Files

```
<os>
  <family name="windows">
    <version name="2000">
      <image arch="i386" build="unknown" edition="advanced server" sp="0" id="image0000">
        <location>
          /images/Windows 2000 Advanced Server - SP0/Windows 2000 Advanced Server - SP0.vmx
        </location>
        <boot_time>120</boot_time>
      </image>
      <image arch="i386" build="2195" edition="advanced server" sp="0" id="image0001">
        <applications>
          <application name="Helix Server" version="9.0.2.766" />
          <application name="Microsoft IIS" version="5.0" />
          <application name="Windows Media Services" />
          <application name="IE" version="5.00.2920.0000" />
          <application name="ipswitch" version="8.13" />
        </applications>
        <location>
          /images/Windows 2000 Advanced Server - SP0 - helix - iis - ie/Windows 2000 Advanced Server - SP0 - helix
        </location>
        <boot_time>140</boot_time>
      </image>
      <image arch="i386" build="2195" edition="advanced server" sp="0" id="image0002">
        <applications>
          <application name="RealServer" version="8.0.1.367" />
          <application name="IE" version="5.00.2920.0000" />
          <application name="Kerio Personal Firewall" version="2.1.4" />
          <application name="WINS" />
        </applications>
        <location>
          /images/Windows 2000 Advanced Server - SP0 - rs - IE - kerio - securecrt/Windows 2000 Advanced Server -
        </location>
        <boot_time>120</boot_time>
      </image>
      <image arch="i386" build="2195" edition="advanced server" sp="0" id="image0003">
        <applications>
          <application name="RealServer" version="8.0.0.149" />
          <application name="IE" version="6.0.2600.0000" />
        </applications>
        <location>
```

First stage - XML Files

```xml
<?xml version="1.0"?>
<module name="IE createTextRange() Exploit">
  <category name="exploit">
    <subcategory name="client-side"/>
  </category>
    <parameters>
        <parameter type="string" key="TARGET"></parameter>
        <subparameters key="EMail Sending">
            <parameter type="string" key="FROM EMAIL">someone@example.com</parameter>
            <parameter type="string" key="FROM FULL NAME">Mail Delivery System</parameter>
            <parameter type="string" key="MESSAGE SUBJECT">Undelivered Mail Returned to Sender</parameter>
            <parameter type="string" key="SMTP SERVER"></parameter>
            <parameter type="uint16" key="SMTP PORT">25</parameter>
            <parameter type="string" key="MAIL SENDER AGENT">/localagent</parameter>
        </subparameters>
        <subparameters key="Web Server">
            <parameter type="string" key="WEB SERVER AGENT">/localagent</parameter>
            <parameter type="uint16" key="WEB SERVER PORT">80</parameter>
            <parameter type="string" key="URL PREFIX">CTRPAGE</parameter>
            <parameter type="string" key="URL BASE"></parameter>
        </subparameters>
        <parameter type="uint16" key="AGENT_PORT">0</parameter>
        <parameter type="string" key="CONNECTION_METHOD">Connect from target</parameter>
    </parameters>
</module>
```

First Stage – Problems - IPs

- Exploits need target IP
  - Images cannot have fixed IP addresses
    - Old and internal constraint (now gone)
  - VMTools used to retrieve IPs for almost all images, except:
    - Windows NT4 pre SP6
    - OpenBSD
  - Solved using a RARP like script
    - Vmimages allow custom MAC address
    - Internal repository keeps info about those images

First Stage – Problems – mutual exclusion

- Automatic tests shut down images after use
- GSX shared between automatic tools and testers
  - Unhappy testers
- Simple locking mechanism
  - Do not start up image if in use by human tester
  - Refcounting of images (when used by multiple automatic tests)
  - Last one turns image off

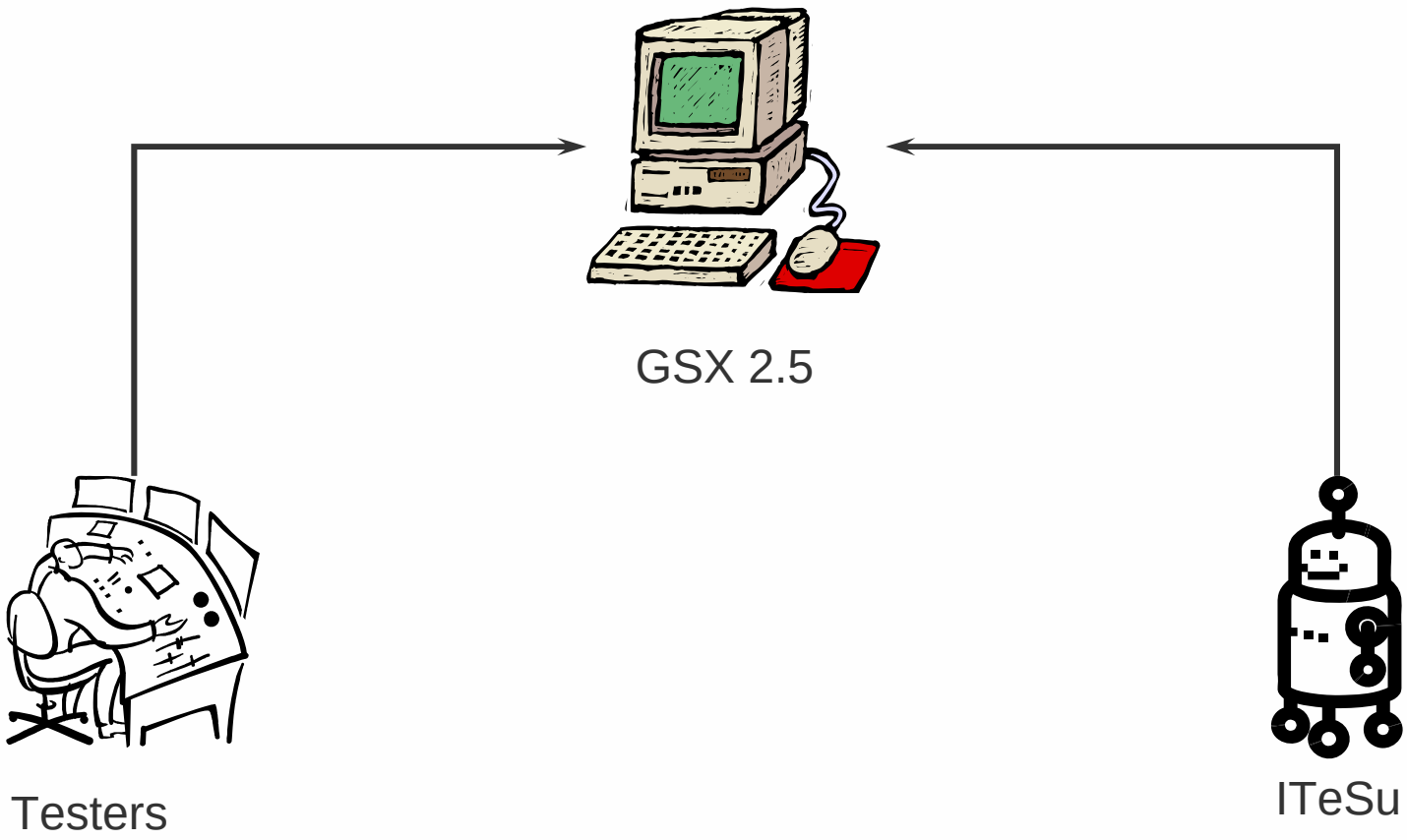
First Stage – Problems - Perfomance

- One-shot exploits
  - Some exploits are considered one-shot
  - No matter if they succeeded or not, machine/service becomes unreliable or unavailable
    - Requires additionals restarts between runs
- Virtual machines power on/off accounts for 85% of testing time
- Memory shortage
  - 512Mb allow for 3 or 4 simultaneus images
    - Staff is growing and resources became critical

First Stage - achieved

- Milestone is achieved
- Automatic test process is considered very promising
- Planning starts for second stage
- At this time there are 60 exploits
  - All must be included and executed daily
- Automatic testing tool named ITeSu

First Stage - Infrastructure

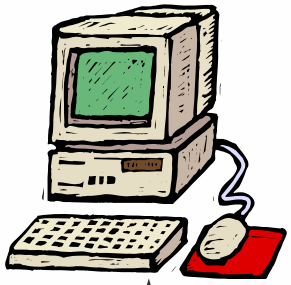


Second stage

- New hardware is acquired
  - 2 AMD Sempron 2800+ 2Gb RAM
  - Linux based
- One GSX dedicated to manual tests
- One GSX dedicated to automatic tests
  - Mutual exclusion problems “*solved*” this way (no more critical resources)
- New problems
  - Image desynchronisation
  - HDD Space shortage

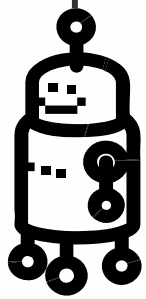
Second stage - Infrastructure

GSX 2.5



Testers

GSX 2.5



ITeSu

Image desynchronisation

- Both GSX started as exact copies
- As time goes by, subtle differences are introduced
- Going further gross differences are introduced
- Impossible to switch between one GSX and the other, too many differences
- Problem: Too many people in a small place

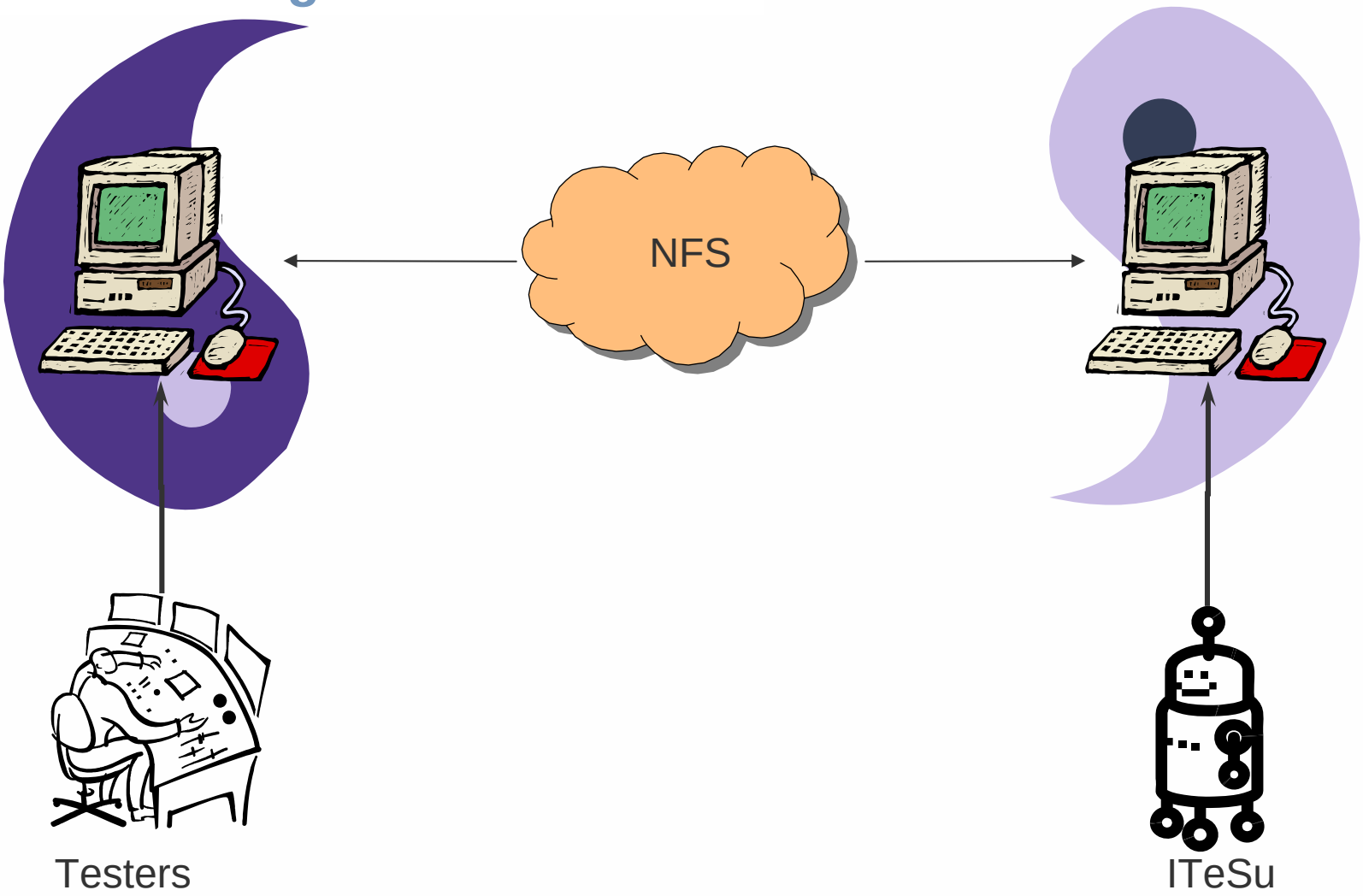
HDD Space shortage

- Given
  - The success of automatic tests
  - QA team growth
  - Exploits growth
  - Clients growth
- Images are added at a very quick rate
- Soon HDD space is almost gone
  - Need for temporal space growths
  - Testing becomes problematic

Desynch. and HDD shortage solution

- Images are cleaned up
  - Unnecesary applications removed
  - Similar (non conflicting) images merged
- File system is split
  - Half of the images are stored in one GSX, the rest in the other one
- Images are shared between systems via NFS
  - Now there is only one copy of each image, (impossible to desynchronise)
  - Each HDD at 50%+ of its capacity
- Drawback
  - If one server goes down, 50% images are unavailable

Second stage - Infrastructure



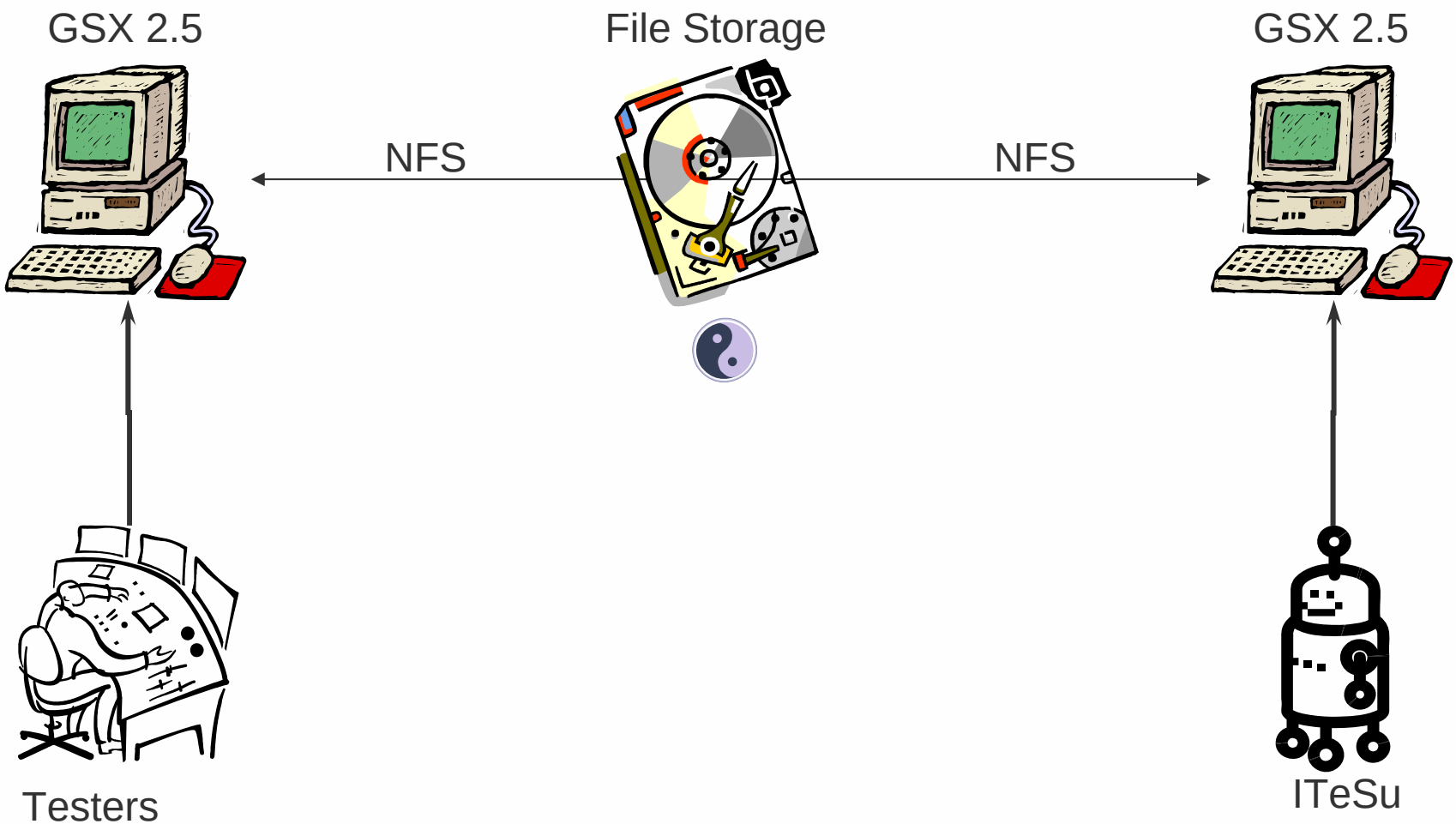
Desynch. and HDD shortage solution (didn’t last)

- Eventually image quantity and content grew as the product evolved
- Temporal space exhausted again
- Hardware problems made GSX sporadically unavailable, 50% images gone

Desynch. and HDD shortage solution II - The revenge

- New hardware is acquired
  - Used as file repository
- HDD with hardware RAID
  - Plenty of space, *“640k should be enough for anybody”*
- No space problems since then

Second stage - Infrastructure



Second stage – approaching limits

- Plenty of HDD space achieved
  - This is a problem
- More than 100 vmimages installed
- Roughly 60 modules included in daily tests
  - Accounting for 200+ runs
- Minimum running time 18 hours
  - Start up / shut down times raising
    - More services in each image, boot times increase
- Still lots of exploits to be included
  - Estimated time when completed is far beyond 24hs
- IMPACT is more complex every time
  - More tests and test types are needed

Third stage

- GSX 3.2 migration
- Use of snapshots
- Support for (so far) unsupported platforms
  - Windows 2003
  - Mandrake 10
- Improved performance

GSX3.2 - Snapshots

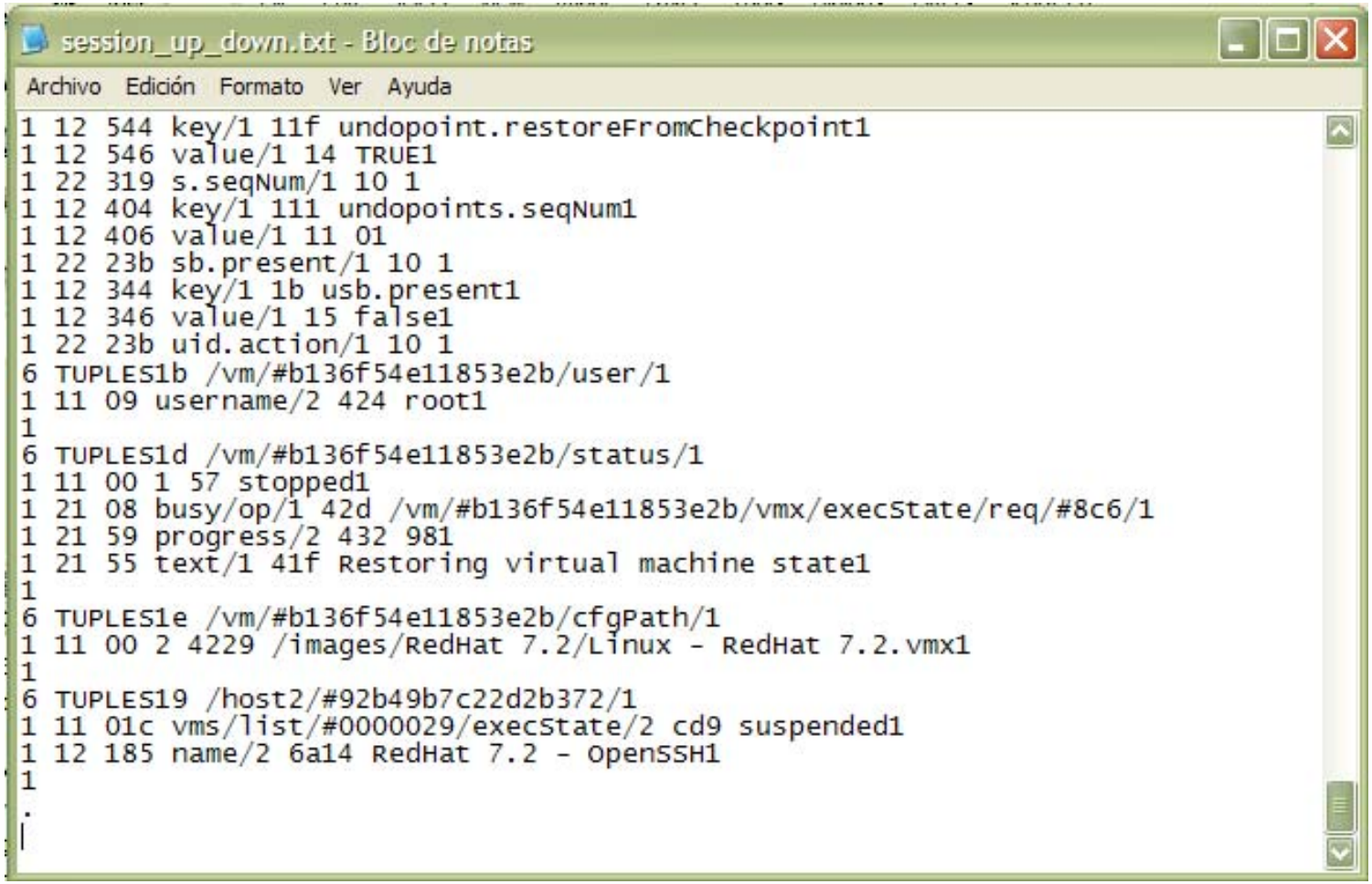
- THE solution to our prays
  - Or not?

- Lack of API snapshot support
  - There is no way to use snapshots
  - Snapshots restart by themselves (revert to snapshot when power off)
  - Snapshots cannot be HARD powered off, lots of broken images

GSX3.2 – Snapshots *cont.*

- VMware console sniffed
  - Somehow console was able to use snapshots
- Custom “snapshot” command
  - Images configured to revert to snapshot on power off
    - Power on -> shut down
    - Power off, wait to power on
    - Power on, suspend
  - It works

GSX 3.2 – snapshots – sniff detail



Third stage – current status

- Testing time dramatically decreased
- Images are now 183
- Included exploits 170
- Total runs 700
- Running time 7 hours

That’s all

Questions?