Breaking cloud isolation

HITB, Amsterdam, 30/05/14
Short BIO

- bug hunter (Facebook, Google, Nokia, etc)
- security researcher
- CEO and lead security expert of Wallarm
Clouds

- Between business functions and hardware
- Between application code and environment
Shared hostings

- Grandfather of clouds ;)
- Many technologies that were made here became basis of clouds
The basics: cloud aliases

- Same application with different data - SaaS
- Same hardware with different platform - PaaS, IaaS
- It’s easy to determine technology point of next *aaS marketing
The basics: resource sharing

- Filesystems
- Network services
- Execution context at OS
The basics: resource sharing

● Filesystems
  ○ files contents
  ○ files names <- don’t forget about that: sess_{abcdefg}
  ○ file descriptors <- so IMPORTANT

● Network services

● Execution context at OS
File sharing

- Different application instances on the same filesystem

- Sensitive files
  - cross-instances content (application code,
  - temporary, reports and other race conditions
File sharing

- Different application instances on the same filesystem
- Sensitive files
  - authentication such as sessions
  - uploaded files
  - temporary, reports and other race conditions
File sharing

- Different application instances on the same filesystem

- How to protect:
  - different chroot and user for each?
  - only 65535 uids at OS =)
  - control chuid() for forks
File sharing

- Different application instances on the same filesystem
- Required LFI/Path traversal bug first at SaaS
- Typically for SaaS, shared hostings fixed that at late 90th ;)
File descriptors

- Important when you open FD before fork or after - privileges for chuid() programs
- API for all interpreters (Ruby, Python, PHP, ...)
- Typical cases:
  - descriptor for database connection (already authed)
  - descriptors for log files and journals
Difficult case from a wild (our practice SaaS security audit)

- Code prototype:
  - fopen()
  - do something, such as fwrite(), flush(), …
  - fclose()
Difficult case from a wild (our practice SaaS security audit)

- Hacker’s look at execution flow:
  - fopen()
  - fwrite() something interesting
  - application crash crash (by memory or exec.time)!
  - fclose() - never called
  - garbage collector magic
  - use foreign FD for our purposes

Important thing!
Theme for another full report

Same worker (PID)

victim’s HTTP request processing

attacker’s HTTP request processing
The basics: resource sharing

- Filesystems
- Network services
  - databases tables (MySQL, Oracle, Postgres, ...)
  - noSQL values (memcached, Tarantool, Redis, Couch, MongoDB, ...)
  - custom services (monitoring, billing, management)
- Execution context at OS
Network resource sharing

- Authentification
  - Privileged ports protection (<1024)
  - Host-based <- SSRF power here
  - Plain/text (login+passwords) <- MITM here
  - Challenge/response (SASL and others)
Network resource attack ways

- Spoofing
  - Classic UDP - rare from Internet, common from intranet (from cloud node) - net.ipv4.<all>.rp
  - TCP Fast Open secret leak at clouds (IP reusing)
- Unprivilege (<1024) local port reusing
- SSRF classics - bypassing host-based auth
Classic UDP spoofing nowadays

- Packet routes between interfaces!
  - By default at Debian/RHel ;)
  - Use `sysctl net.ipv4.<all>.rp` to disable
- UDP services at loopback interface are really common
- TFTP - netboot images, gain control at new nodes at (P|I)aaS (SNMP also, but community str there)
- Memcached (by default 11211 TCP and UDP both)
TCP fast open spoofing at clouds

1. Timestamp A
   - TFO SYN with cookie request
   - TFO cookie for IP 10.3.13.37

2. Timestamp B>A
   - TFO SYN+data+old cookie for IP 10.3.13.37 spoofing Host C
Local port reusing

- Required RCE first of course
- 3rd party privileged application on non-privileged ports
- Crash them then open this port. I think you can do that! Fuzz it guys, FuZ5!!!
- Get some private data from others
Local port reusing

- Cases from a wild
  - monitoring
  - management systems
  - privileged daemon for anything
  - different integration daemons
  - different databases - SQL/noSQL
Classic SSRF

- From the Internet to Intranet
- Sometimes better than many A01 injections
- Internal API and others - are you forget about auth there?
- Intranet resources: monitoring/wiki/etc - vlan!
FastCGI SSRF features

- Local port for fastcgi is bad
- Use unix sockets for that
- In other cases applications can communicate locally by each others
- For PHP-FPM admin_value provide RCE

https://github.com/ONsec-Lab/scripts/blob/master/fastcgipacket.rb
The basics: resource sharing

● Filesystems

● Network services

● Execution context at OS
  ○ classic race condition at daemon init scripts
  ○ depletion entropy of urandom ???
What the problem?

- Look at CVE-2013-1048 first - that really cool
- $ install utility has great error - race condition between create file and set privileges
- Good way:
  - fd = open(...)
  - fchmod(fd,...)
/dev/random concept

- Just only CONCEPT
- Attacker’s worker read all /dev/random
- Victim’s worker read /dev/urandom consists of hashes from /dev/random readed before by attacker
- Attacker now know victim’s randoms
- There are many limitation of cource...
The end

Contacts:

@wallarm, @d0znpp

http://github.com/wallarm