Network Security 4: Isolation

i.g.batten@bham.ac.uk

Last Lecture

- Reducing services to minimum
- Removing services which are not being used
- Restricting access to services which are needed to support other services

Least Privilege

- Old-style, the easiest way to run system services is as root / Administrator
- Any compromised service therefore has access to the whole machine
- First step: run services as unprivileged users, with appropriate magic to do anything that needs privilege in a secure way.

Why do you need root?

- To open privileged ports
  - Some operating systems allow you to do this without privilege, or with fine-grained privilege
    - Alternatively, open the port, then permanently rescind your privilege
- To write user files
  - There are other, better ways to do this
- So whole “cyrus” mail system runs as user cyrus

The next step: isolation

- Processes running on a machine can “see” each other, and exploit weaknesses up to attaching a debugger.
- So an attacker who compromises one service can attempt to gain control of other services.
- Eventually they may be able escalate to root via a chain of progressively more “powerful” vulnerabilities, and the processes they take over may also be useful in their own right

In passing…

- In 2018, we don't care quite as much about protecting operating system images, so the "unprivileged" service is probably more important that the operating system instance.
- Real data is more important than keeping the machine running, as we can rebuild from media or restart from snapshots.
- The security model of the operating is precisely the wrong way around: it worries about the precious binary of /bin/cat far more than about your final year project.

So ideally…

- You only have one service per machine (where “machine” means “operating system instance”, as we will see).
  - Then, an attacker who controls one service just controls that service, and there’s nothing else of value on the machine: doesn’t prevent the attack, but mitigates its consequences and avoids escalation.
- But machines are expensive, in terms of power, space, administrator effort, etc.
- Operating system licenses and support contracts aren’t necessarily free, either.

Virtualisation and other OS isolation facilities

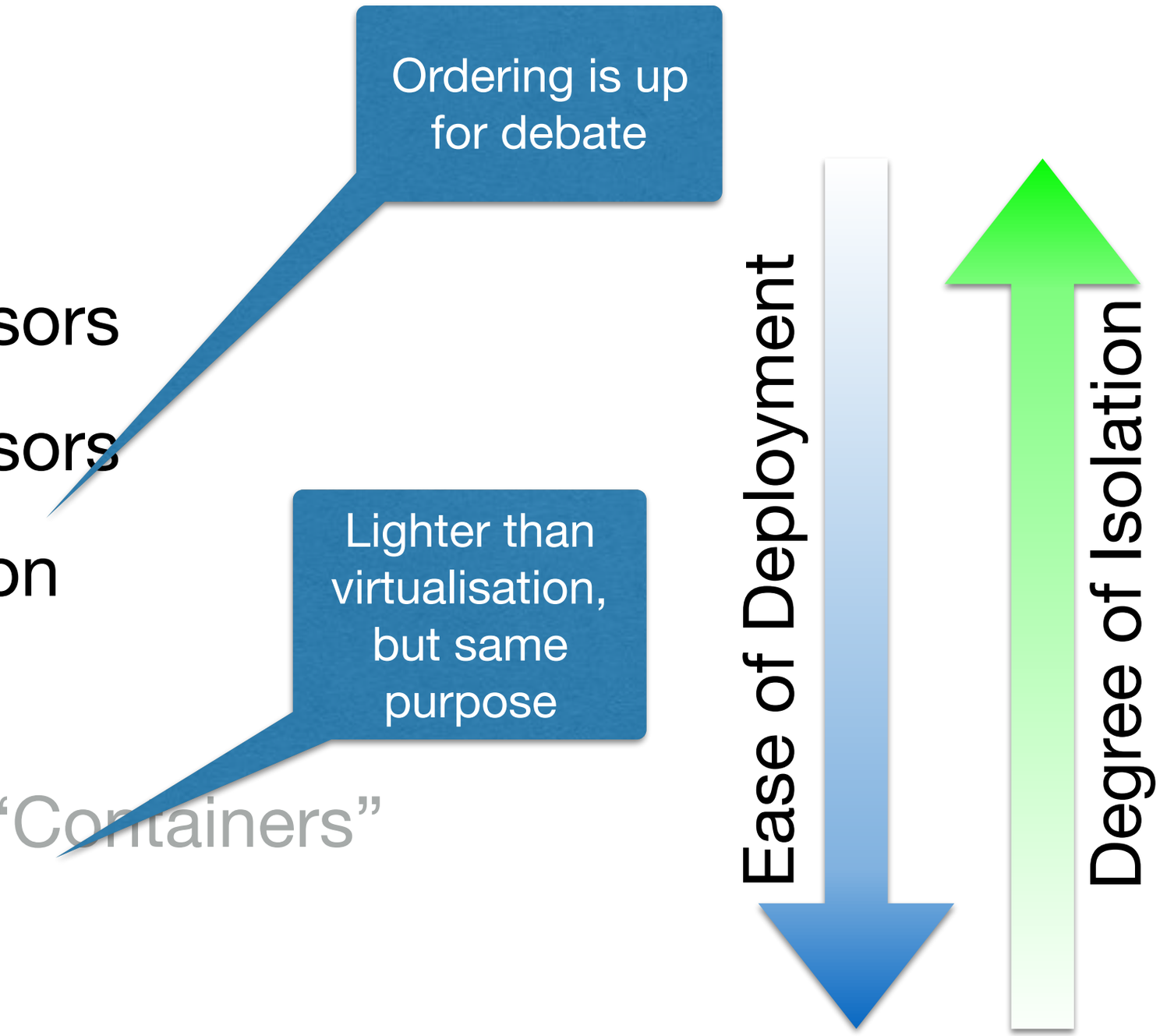
- Virtualisation covers a wide range of techniques for running **guest** operating systems, or things that look like operating sys er (**host**) operating systems.

Got here 1500 Tue 16

- Unprivileged instr are metal, privileged instructions which interact with real hardware are trapped and emulated
- It's worth looking at the spectrum of options.
  - There is some hand-waving in this, as the distinctions may not be quite as clear-cut as I imply

Spectrum of Isolation

- Type 1 Hypervisors
- Type 2 Hypervisors
- Paravirtualisation
- Jails / Zones / “Containers”



Type 1 Hypervisors

- Host runs a hypervisor, which does nothing but manage guest operating systems. You can't run anything in the host, and it's normally protected from the outside world.
  - Hush, hush, whisper it: the Hypervisor is often actually Linux…
- Guest operating systems are unmodified, and think they are running on a defined and supported hardware platform.
- Offers a wide range of facilities for redundancy, load balancing, migration, etc, etc.
- With minor exceptions, each OS instance runs its own networking, talking to virtual bridges or virtual VLAN, often with hardware assistance (VT-c, SR-IOV)
- VMware's big enterprise solution, ideal for big server farms. In best practice, the hypervisor has its own network interfaces on a private management network.

Type 2 Hypervisors

- Full-fat operating system running on the host
- Which then supports a hypervisor which allows guests (again unmodified) to run
- Guests can (for example) display in a host-OS window.
- Guest OSes might have a bridge to the network, but might be NAT-ed and therefore all traffic goes through the (full-fat) host OS networking stack.
- Full-size host OS is attractive target for attackers: an attacker who is root on the host (via attacks on its processes) can almost certainly immediately compromise all guests.
- VMware Fusion, Virtualbox, etc.

“Real” Virtualisation

- Guest and host can be completely disjoint: common scenario is running Windows as a guest in a Linux host, or vice versa. Great for developers, great for PaaS, great for data centre managers (most computers are idle most of the time, so better value per U of rack space)
- Resource allocation and scheduling can be complex and difficult to control, but are not a topic for this course.

Para Virtualisation

- Operating system is slightly modified, rather than running unchanged, so that it uses special network, graphics and other drivers, which exist only as stubs in the guest OS, and often a lot of reliance on the host virtual memory subsystem.
- Arguably more efficient, but requires the OS vendor to co-operate.
- Hypervisor has more visibility of guest, and potentially vice versa (active research in department on running virus/malware scanners in host to look “into” guest OSes: security as a service).
- Usually Linux or other open-source OS as the guest: Xen-guest versions of other OSes are intermittently available (Solaris was for a while, MSFT research did a Xen-XP in the early days) but are not common.

Zones, Jails, Containers

- “Jails” are available in FreeBSD.
- “Zones” (aka “Containers” in marketing speak) are available in Solaris derivatives (ie, commercially and supported).
- Machine runs one kernel, but a subset of processes are isolated with their own init, filesystem, sometimes networking.
- Very lightweight and easy to manage, consumes almost no additional resources. Only needs one OS license.
- Security properties look good, but fair to say analysis is complex as the interface between container and its host is “fat”. Ripe topic for a PhD. Cambridge Mirage project (in OCAML!) is an interesting alternative spin to look at.
- Linux analogues came later but are storming up the inside into the lead: Linux Containers now (because of Docker) do almost everything that Solaris container do, and are cheaper/easier/more common.
    - I believe you can’t have separate IPsec configurations in distinct containers on Linux. About three people in the world care about this, one of them me, and the security properties of it would be tricky to assess.

Hardware

Host OS (if Type 2 Hypervisor)

Hypervisor

Solaris

Linux

FreeBSD

Zone

Zone

Zone

Container

Container

Container

Jail

Jail

Jail

Collectively probably managed with Docker today

chroot ()

- Very old-fashioned Unix tool, the ultimate basis for things like zones and containers.
- After chroot (“/some/directory”), all processes which are started from point on see “/some/directory” as “/“. Processes cannot get outside that part of the filesystem.
- Specifically, “cd ..” stops at the new “/“, similarly “cd /“.
- Complex to make user code work correctly (libraries, devices, /proc, etc).
- Anyone clever enough to use it is clever enough to use something better.

How to choose?

- All forms of virtualisation separate the number of application instances from the volume of tin in the machine room (“tin” —computers for the (un) hip).
- The lightest forms separate the number of application instances from the number of OS instances.
- All offer some sort of mobility from “shut down, copy the underlying files, start up on another machine” to almost real-time failover.

“Real” Virtualisation

- Doesn’t require modified Operating Systems
  - Probably only realistic choice for running Windows on multi-tenant platforms and getting any support, although situation is changing rapidly.
  - Ideal for running legacy OSes on modern hardware (DOS 5.0! Windows XP! BeOS!)
    - Some configurations allow you to hide legacy networking code behind modern stack, but security properties of this again hard to evaluate.
- Usually has extensive support for redundancy, relocation, live upgrade, etc.
- Hardware required to do it well is expensive
  - Requires lots of RAM, in particular
  - Requires high-end SAN storage behind the servers to support mobility.
    - Storing the virtual images on NAS is for the brave

Para Virtualisation

- Good luck getting OS support!
- Ideal for selling private servers to people that want them.
- Guests are reliant on host for some services (for example, clock synchronisation is usually done by host OS)
- Not so easy for live mobility (because the “state” of the guest is in part the “state” of the host) although a lot of work has been done.
- Good enough for many, perhaps most, purposes: Amazon, etc.

Zones, etc

- Fully supported by vendors, indeed encouraged by vendors.
- Doesn’t always help you with different OSes, but Docker.
- Might help with different OS versions (Solaris 11 can run Solaris 10 and Solaris 8 applications in zones, although it isn’t bug-for-bug compatible as it’s “really” a Solaris 11 kernel).
- Mobility is usually limited to moving cold systems, but there is active work on improving this.
- Cheap, lightweight, ideal if your stack fits it.
- Docker is changing the trade-offs as we speak.
- I think in 2017 a LAMP/etc application which isn’t deployed via Docker should be asked “why not?”

HMG Footnote

- *I have a medical textbook which has special "except if you're military and in the field" sections denoted with camouflage borders to the pages. I need something similar for "what about classified systems?"*

- It is highly unlikely that any of these isolation techniques are regarded as strong enough to separate different classifications, and unlikely that they are regarded as enough to separate different assets at the same classification at the higher levels without a lot of effort and the use of assured (ie, expensive) solutions.
- The segregation is just too hard to audit, either at the time of selection or on an on-going basis.

Network Virtualisation

- Virtualisation suites allow you to build virtual networks inside the virtualised environment, and run virtual firewalls and virtual intrusion detection systems.
- We'll talk about this after we've talked about firewalls and IDSes.