ECE560
Computer and Information Security

Fall 2020

Endpoint security

Tyler Bletsch
Duke University

Overview

How do you configure **endpoints** for better security?

1. Updates
2. Correct settings
3. Reduce attack surface
4. Limit privilege
5. Isolate components
6. Good authentication
7. Backups
8. Monitor things
9. Endpoint security software

(Not covered: network-level security. That's next.)

#1: Updates

- Ensure updates are **enabled** and **automatic**
- Updates patch out discovered vulnerabilities
- The top tier attackers will use ***zero-day*** **vulnerabilities** (ones that are not known or patched against, i.e. you have *zero days* of notice)
  - But most attackers are not top-tier!
- *Most* vulnerabilities exploited already have patches available!

The only reason not to do this is laziness or stupidity.

#2: Correct settings

- Check over **all** settings carefully
- Common mistakes:
  - Kept the default password
  - Forgot to have encryption enabled where possible (HTTP->HTTPS)
    - Didn't use proper CA-signed certs – just lived with warnings!
  - Permissions set too permissively by default
  - One-time setup facility left enabled
  - Service meant to listen local-only (127.0.0.1) set to global (0.0.0.0)
  - Windows: TONS of settings in Active Directory (a whole field of study)

- Most of the above fall into the later categories, but you gotta do it for *all* the applications/tools you have. Think systematically!

#3: Reduce attack surface

- Turn off or remove **everything** you don't **need;** install **only** the packages you **need** – no more!
- Examples
  - Windows: Turn of the dozens of needless services (e.g. file sharing server).
  - Linux: Don't pick a GUI version for a web server.
  - Need a basic web server for static content? Don't pick the "LAMP stack" wizard that installs Apache/MySQL/PHP –install a web server daemon specifically focused on static content (e.g. nginx).
- Special note – **Firewalls**:
  - Firewall: A software or hardware system that selectively filters packets
  - Endpoint firewall: Runs in OS of a server/workstation
    - Set to allow only the flows you need (e.g. a database only needs to get connections from its known clients, not all IPs)
  - Network firewall: Runs on network device (physical or virtual)
    - We'll cover this later

#4: Limit privilege

- Configure permissions to the minimum needed.
- Examples (Linux in blue, Windows in pink)
  - **Disable root/Administrator** login (privilege via sudo/UAC only)
  - Services run as **unprivileged accounts** without sudo/admin access (default for most packages in Linux)
  - Inspect **filesystem permissions** and ensure it's as restrictive as feasible
  - Enable and configure **Mandatory Access Control (MAC)**: selinux/apparmor for Linux
  - Use **application whitelisting** so only specific programs can execute (via selinux or AppLocker)

#5: Isolate components

- Apply the strongest feasible form of isolation between parts
- Forms of isolation (from weakest to strongest)
  - Separate **process** (just a different memory space)
  - Separate **user accounts** (different filesystem permissions)
  - Separate **container** (separate kernel namespace)

    E.g. Docker containers (discussed later)
  - Separate **virtual machine** (separate OS kernel)
  - Separate **physical machine** (separate hardware, network can be filtered)
  - **Airgapped** physical environment (no network connectivity at all)

- Note: cost of hardware and labor is correlated with level of isolation
  - Tradeoff!

#6: Good authentication

- Apply good **password practices** and **multi-factor authentication**

- (We already discussed this...so now do it!)

#7: Backups

- Take periodic backups of all relevant data. A backup solution must:
  1. Record changes to data **over time**
     - If I just have the most recent copy, then I just have the most recently corrupted copy. *RESULT: MIRRORING ISN'T BACKUP!!!!*
  2. Have a copy at a **separate physical location**
     - If all copies are in one place, then a simple fire or lightning event can destroy all copies
  3. Must be **automatic**
     - When you get busy, you'll forget, and busy people make the most important data
  4. Require **separate credentials** to access
     - If one compromised account can wipe primary and secondary, then that account is a single point of failure
  5. Be **unwritable** by anyone except the backup software (which ideally should live in the restricted backup environment)
     - If I can cd to a directory and change backups, then the same mistake/attack that killed the primary can kill the backup
  6. Reliably **report** on progress and **alert** on failure
     - I need to know if it stopped working or is about to stop working
  7. Have periodic **recovery tests** to ensure the right data is being captured
     - Prevent "well it apparently hasn't been backing up properly all along, so we're screwed"

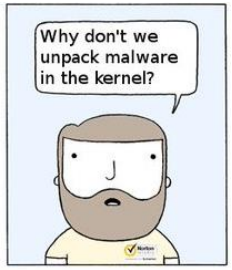
#8: Monitor things

- Record and analyze logs
- Set up logs for*:
  - multiple failed login attempts, especially for critical accounts. This includes cloud aggregator services like Office 365 or GSuite
  - successful logins to your CMS and changes to any of the files in it (if you don't change them often)
  - changes to your log configurations
  - password changes
  - 2FA requests that were denied
  - anti-malware notifications
  - network connections going in and out of your network
- Use a log centralizing system (e.g. syslog) and some form of analysis (e.g. logwatch).

* Quoting from New Zealand CERT's Top 11 Cyber Security tips

#9: Endpoint security software

- Deploy endpoint security (e.g., so-called “antivirus” software)
- Basic “antivirus” can be good – catches common malware by signature analysis
  - Ensure good quality to avoid making things worse!



<

Symantec/Norton
(link to vulnerability disclosure)

Microsoft Windows Defender
(link to article on useful security enhancement)

- Host-based Intrusion Detection System (HIDS)
  - Monitor host configuration and activity
  - Detect attacks by known signature or by anomalous behavior
  - Send reports and/or trigger alerts

Conclusion

- A 2010 study by the Australian Defense Signals Directorate found that implementing just SOME of these things (updates, avoiding administrator privilege, and application whitelisting) would have prevented **85%** of intrusions reported in that year.

- So do them.

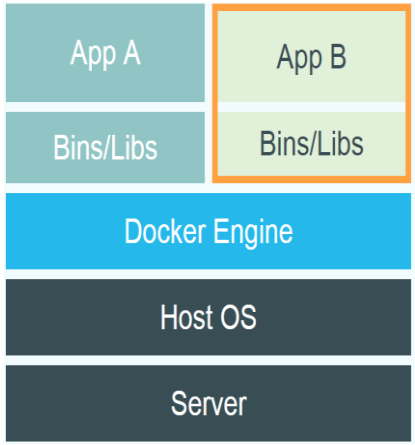
Further reading on hardening strategies is available from the Australian Cyber Security Centre:
Strategies to Mitigate Cyber Security Incidents

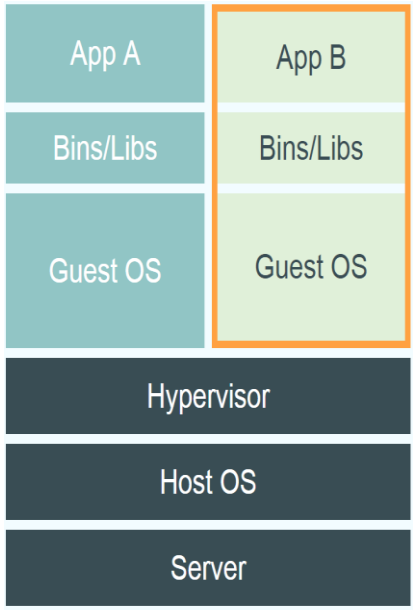
Quick intro to Docker

Adapted from "Linux Containers and Docker"
by Keke Chen, Wright State University.

Introduction

- Linux containers (LXC) are “lightweight” VMs
- Docker is a commoditized LXC technique that dramatically simplifies the use of LXC
- Containers versus VMs:





Adapted from “Linux Containers and Docker” by Keke Chen, Wright State University.

Container technique

- Linux kernel provides the “control groups” (cgroups) functionality (cgroup-tools)
  - allows limitation and prioritization of resources (CPU, memory, block I/O, network, etc.) without the need for starting any VM
- “namespace isolation” functionality
  - The “unshare” command
  - allows complete isolation of an applications' view of the operating environment, including process trees, networking, user IDs and mounted file systems.
  - Possible to have a new “init” process for each container

Adapted from “Linux Containers and Docker” by Keke Chen, Wright State University.

Unique features

- Containers running in user space
- Each container has
  - Own process space
  - Own network interface
  - Own /sbin/init (coordinates the rest of the boot process and configures the environment for the user)
  - Run stuff as root
- Share kernel with the host
- No device emulation. The system software and devices are provided by the image

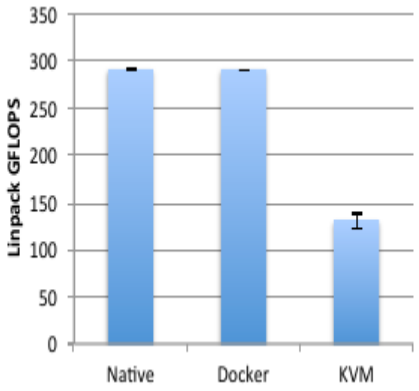
Check the namespace isolation…

- Pid namespace
  - Type “ps aux| wc –l” in host and the container
- Mnt namespace
  - Type “wc –l /proc/mounts” in both
- Net namespace
  - Install net-tools
  - Type “ifconfig”
- hostname namespace
  - “hostname”
- ipc namespace
  - Type “ipcs”
- User namespace
  - UID 0-1999 in the first container mapped to UID 10000 – 11999 in host
  - UID 0-1999 in the 2nd container mapped to UID 12000 – 13999 in host

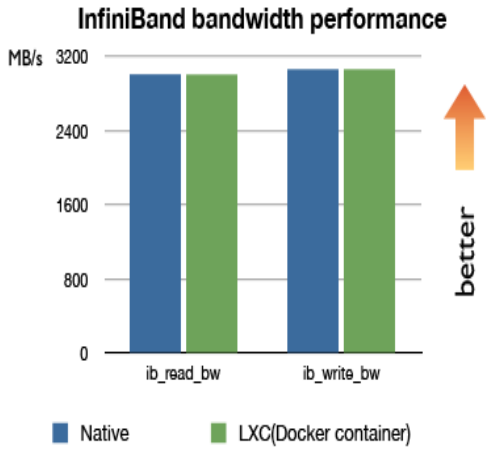
Adapted from “Linux Containers and Docker” by Keke Chen, Wright State University.

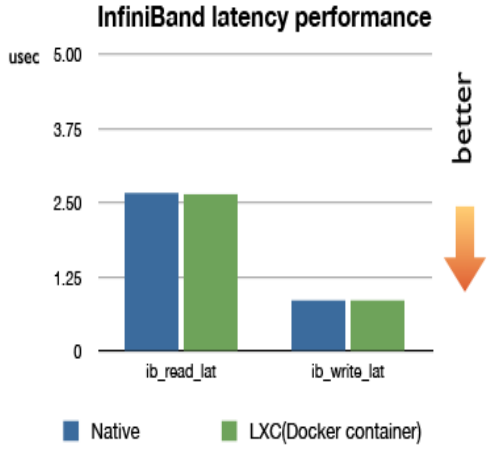
Almost no overhead

- processes are isolated, but run straight on the host
- CPU performance = native performance
- memory performance = a few % shaved off for (optional) accounting
- network performance = small overhead; can be reduced to zero



Linear algebra





Networking

Adapted from "Linux Containers and Docker" by Keke Chen, Wright State University.

What is docker: summary

- Open Source engine to commoditize LXC
- Uses copy-on-write for quick provisioning
- Allows one to **create and share** *images*
- **Standard format** for containers
- Standard, *reproducible* way to *easily* build *trusted* images (Dockerfile, Stackbrew...)