CSF 434/534: Advanced Network and System Security

Week 08 - Review

Michael Conti

Department of Computer Science and Statistics
University of Rhode Island

Sources: Professor Messer's CompTIA SY0-501 Security+ Course Notes

Development Life-Cycle Models

Development Life-Cycle Models

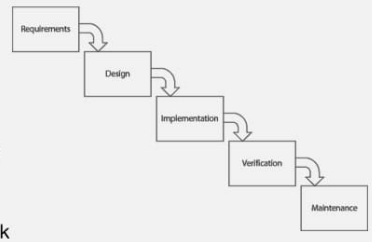
Building an application

- Systems development life cycle
  - Or application development life cycle
- Many ways to get from idea to app
  - And many moving parts
  - Customer requirements
  - Keep the process on schedule
  - Stay in budget
- There's no "best way"
  - But it helps to have a framework
  - There are many options

Development Life-Cycle Models

Waterfall

- Sequential design process
  - First step, then second step, then third step...
  - Considered to be the traditional model
  - Many different flavors of waterfalls
- The waterfall
  - Requirements: Document the request
  - Analysis: Build models and business rules
  - Design: Pick a software architecture
  - Coding: Development and integration work
  - Testing: Debug the application
  - Operations: Install and support the application



Development Life-Cycle Models

Agile

- Ready, shoot, aim. Repeat.
  - It's better to get moving than to wait around
- Everyone works together
  - Co-location and pair programming
- Get some code out there
  - An app under construction is better than slideware
- Customer collaboration - Constant communication
- Quick response to change - Development is continuous

Secure DevOps

Secure DevOps

DevOps

- Development and Operations
  - Bring together the two sides of the house
- Create and deploy
  - Speed, availability, and security
- Emphasis on automation and monitoring
  - Integration, testing, release, and manage
- Shrink deployment cycles
  - And increase the frequency of deployments
- How do you do this safely?

Secure DevOps

Security automation

- Automation is relatively inexpensive
  - It's automated, so run them early and often
- Functional security tests
  - Login, logout, ensure a secure platform
- Test against known vulnerabilities
  - Misconfigurations, weak SSL ciphers, etc.
- Penetration testing
  - Test the OS and application services
- Test the application
  - Manipulate the application to get unexpected results

Secure DevOps

Continuous integration

- Code is constantly written
  - And merged into the central repository many times a day
- So many chances for security problems
  - Security should be a concern from the beginning
- Basic set of security checks during development
  - Documented security baselines as the bare minimum
- Large-scale security analysis during the testing phase
  - Significant problems will have already been covered

Secure DevOps

Immutable systems

- Update an application every week for a year
  - It's nothing like the original deployment
  - You couldn't rebuild it if you had to
  - Configuration drift
- Immutable systems
  - Locked down and unable to change
- To update an application, a new iteration is deployed
  - The entire iteration is up to date and tested
- Much stronger security posture
  - Test and validate without worrying about a change

Secure DevOps

Infrastructure as code (IAC)

- Cloud computing
  - Relies on automation
- If you can automate it, you can quickly and safely deploy it
  - Except for the server hardware, switches, routers, firewalls, etc.
- Turn the infrastructure devices into code
  - Virtualize everything
  - Focus on what the application needs, rather than building the application based on available infrastructure
- A clearly defined infrastructure
  - Security is a known quantity

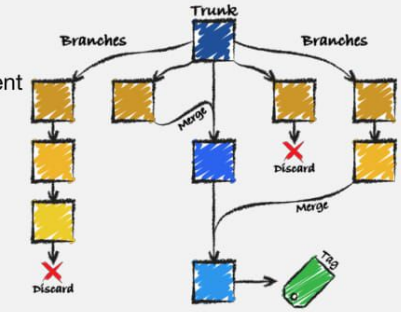
Version Control and Change Management

GitHub

Version Control and Change Management

The constant of change

- There will always be modifications, bug fixes, new features, and security patches
  - The important part is how you manage change
  - It's only chaotic if you allow it to be
- Changes during application development
  - Requires version control
- Changes to production
  - Formal change management



Version Control and Change Management

Version control

- Create a file, make a change, make another change, etc.
  - Track those changes, revert back to a previous version
- Commonly used in software development
  - But also in operating systems, wiki software, and cloud-based file storage
- Useful for security
  - Compare versions over time
  - Identify modifications to important files
- A security challenge
  - Historical information can be a security risk

Version Control and Change Management

Change management

- Change control
  - A formal process for managing change
  - Avoid downtime, confusion, and mistakes
- Nothing changes without the process
  - Plan for a change
  - Estimate the risk associated with the change
  - Have a recovery plan if the change doesn't work
  - Make the change

Version Control and Change Management

Change management and security

- Every change has a security component
  - Each change must be evaluated separately
- Install security patches
  - Ideally makes the systems more secure
- Application update
  - New version, new code, new security concerns
- Change to the application instance
  - New servers, updated middleware, etc.
  - Everything must be evaluated together

Provisioning and Deprovisioning

Provisioning and Deprovisioning

Provisioning

- Deploy an application
  - Web server, database server, middleware server, user workstation configurations, certificate updates, etc.
- Application software security
  - Operating system, application
- Network security
  - Secure VLAN, internal access, external access
- Software deployed to workstations
  - Check executables for malicious code, verify security posture of the workstation

Provisioning and Deprovisioning

Orchestration

- Automation is the key to cloud computing
  - Services appear and disappear automatically, or at the push of a button
- Entire application instances can be instantly provisioned
  - All servers, networks, switches, firewalls, and policies
- Instances can move around the world as needed
  - Follow the sun
- The security policies should be part of the orchestration
  - As applications are provisioned, the proper security is automatically included

Provisioning and Deprovisioning

Deprovisioning

- Dismantling and removing an application instance
  - All good things
- Security deprovisioning is important
  - Don't leave open holes, don't close important ones
- Firewall policies must be reverted
  - If the application is gone, so is the access
- What happens to the data?
  - Don't leave information out there

Secure Coding Techniques

Secure Coding Techniques

Secure coding concepts

- A balance between time and quality
  - Programming with security in mind is often secondary
- Testing, testing, testing
  - The Quality Assurance (QA) process
- Vulnerabilities will eventually be found
  - And exploited

Secure Coding Techniques

Error and exception handling

- What happens when an error occurs?
  - "Graceful" exceptions
- Network connection fails, server hangs, database unavailable
  - Think of every possible problem
- Mishandled exceptions can allow execution of code
  - The bad guys love this
- Avoid default messages
  - You'll give away the underlying architecture

Secure Coding Techniques

Input validation

- What is the expected input?
  - Validate actual vs. expected
- Document all input methods - Forms, fields, type
- Check and correct all input (normalization)
  - A zip code should be only X characters long with a letter in the X column
  - Fix any data with improper input
- The fuzzers will find what you missed
  - Don't give them an opening

Secure Coding Techniques

Normalization

- *Normalization* is an initial step in the input validation process
- Step of creating the canonical form of a string before processing
- Eliminates the variance in input form like case, format and input encoding
- Supporting libraries exist for normalizing input

Secure Coding Techniques

Stored procedures

- SQL databases
  - Client sends detailed requests for data
  - 'SELECT * FROM wp_options WHERE option_id = 1'
- Client requests can be complex
  - And sometimes modified by the user
  - This would not be good
- Stored procedures limit the client interactions
  - 'CALL get_options'
  - That's it. No modifications to the query are possible.
- To be really secure, use only stored procedures
  - The application doesn't use any SQL queries

Secure Coding Techniques

Code signing

- An application is deployed
  - Users run application executable or scripts
- So many security questions
  - Has the application been modified in any way?
  - Can you confirm that the application was written by a specific developer?
- The application code can be digitally signed by the developer
  - Asymmetric encryption
  - A trusted CA signs the developer's public key
  - Developer signs the code with their private key
  - For internal apps, use your own CA

Secure Coding Techniques

Encryption

- If you can see the source code, you can easily look for security holes
  - Source code is closely guarded
  - Development platforms should use encryption
- If you're sending data over the network, it should be encrypted
  - Easy to grab data from the air
  - Encrypt important data
  - Some operating systems do this anyway

Secure Coding Techniques

Obfuscation / camouflage

- Obfuscate
  - Make something normally understandable very difficult to understand
- Take perfectly readable code and turn it into nonsense
  - The developer keeps the readable code and gives you the chicken scratch
  - Both sets of code perform exactly the same way
- Helps prevent the search for security holes
  - Makes it more difficult to figure out what's happening
  - But not impossible

Code reuse/dead code

Code reuse / dead code

Code reuse/dead code

- Code reuse
  - Use old code to build new applications
  - Copy and paste
- If the old code has security vulnerabilities, reusing the code spreads it to other applications
  - You're making this much more difficult for everyone
- Dead code
  - Calculations are made, code is executed, results are tallied
  - The results aren't used anywhere else in the application
- All code is an opportunity for a security problem
  - Make sure your code is as alive as possible

Code reuse / dead code

Validation points

- Server-side validation
  - All checks occur on the server
  - Helps protect against malicious users
  - Bad guys may not even be using your interface
- Client-side validation
  - The end-user's app makes the validation decisions
  - Can filter legitimate input from genuine users
  - May provide additional speed to the user
- Use both
  - But especially server-side validation

Code reuse / dead code

Memory management

- As a developer, you must be mindful of how memory is used
  - Many opportunities to build vulnerable code
- Never trust data input
  - Malicious users can attempt to circumvent your code
- Buffer overflows are a huge security risk
  - Make sure your data matches your buffer sizes
- Some built-in functions are insecure
  - Use best practices when designing your code

Code reuse / dead code

Third-party libraries and SDK's

- Your programming language does everything
  - Almost
- Third-party libraries and software development kits
  - Extend the functionality of a programming language
- Security risk
  - Application code written by someone else
  - Might be secure. Might not be secure.
  - Extensive testing is required
- Balancing act
  - Application features vs. unknown code base

Code reuse / dead code

Data exposure

- So much sensitive data
  - Credit card numbers, social security numbers, medical information, address details, email information
- How is the application handling the data?
  - No encryption when stored
  - No encryption across the network
  - Displaying information on the screen
- All input and output processes are important
  - Check them all for data exposure

Code Quality and Testing

Code Quality and Testing

Static code analyzers

- Static Application Security Testing (SAST)
  - Help to identify security flaws
- Many security vulnerabilities found easily
  - Buffer overflows, database injections, etc.
- Not everything can be identified through analysis
  - Authentication security, insecure cryptography, etc.
  - Don't rely on automation for everything
- Still have to verify each finding
  - False positives are an issue

Code Quality and Testing

Dynamic analysis (fuzzing)

- Send random input to an application
  - Fault-injecting, robustness testing, syntax testing, negative testing
- Looking for something out of the ordinary
  - Application crash, server error, exception
- 1988 class project at the University of Wisconsin
  - "Operating System Utility Program Reliability"
  - Professor Barton Miller
  - The Fuzz Generator

Code Quality and Testing

Fuzzing engines and frameworks

- Many different fuzzing options
  - Platform specific, language specific, etc.
- Very time and processor resource heavy
  - Many, many different iterations to try
  - Many fuzzing engines use high-probability tests
- Carnegie Mellon Computer Emergency Response Team (CERT)
  - CERT Basic Fuzzing Framework (BFF)
  - http://professormesser.link/bff

Code Quality and Testing

Stress testing

- The software works with one user
  - What about 1,000 users?
- Inadvertent results can occur at load
  - Unintended error messages
  - Application details and versions displayed to the user
  - Kernel and memory dumps
- Extensive automation options
  - Automate individual workstations
  - Simulate large workstation loads
  - Extensive reporting

Code Quality and Testing

Sandboxing

- A bit different than the developer sandbox
  - A different sandbox at a different playground
- Test environment looks and works exactly like production
  - No production systems are used
  - No production data is used
- QA can fuzz, overload, and try to break the sandboxed environment
  - You can't hurt anything in the sandbox

Code Quality and Testing

Model verification

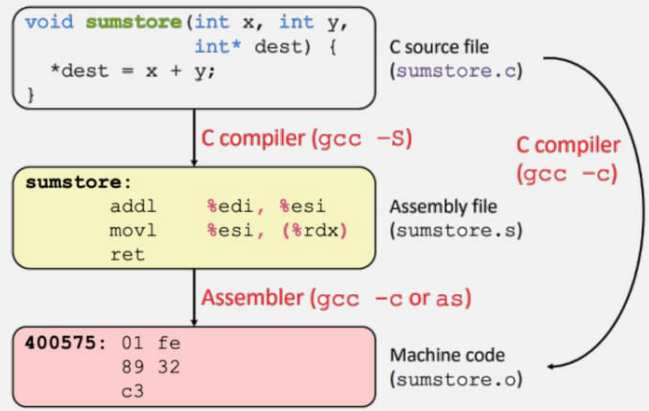
- Verification and Validation (V&V)
  - You started development with a set of requirements
- Verification
  - Does the software work properly?
  - Are there any bugs to address?
  - Are we building the product right?
- Validation
  - Did you meet the high level requirements?
  - Are we building the right product?

Code Quality and Testing

Compiled vs. runtime code

- Compiled code
  - You don't see the source code
  - The application is an executable compiled from the source
  - The compiled code is specific to an operating system and CPU
  - Logical bugs can be identified at compile time
- Runtime code
  - Source code is usually viewable
  - The code instructions execute when the application is run
  - No opportunity to find compile-time errors, so errors are detected during or after the execution

Code Quality and Testing



Cloud and Virtualization Overview

Cloud and Virtualization Overview

Virtualization

- One computer, many operating systems
  - Mac OS X, Windows 7, Linux Ubuntu, all at the same time!
- Separate OS, independent CPU, memory, network, etc.
  - But really one computer
- Host-based virtualization
  - Your normal desktop plus others
- Standalone server that hosts virtual machines
  - Enterprise-level
- Been around since 1967 - IBM mainframe virtualization

Cloud and Virtualization Overview

The hypervisor

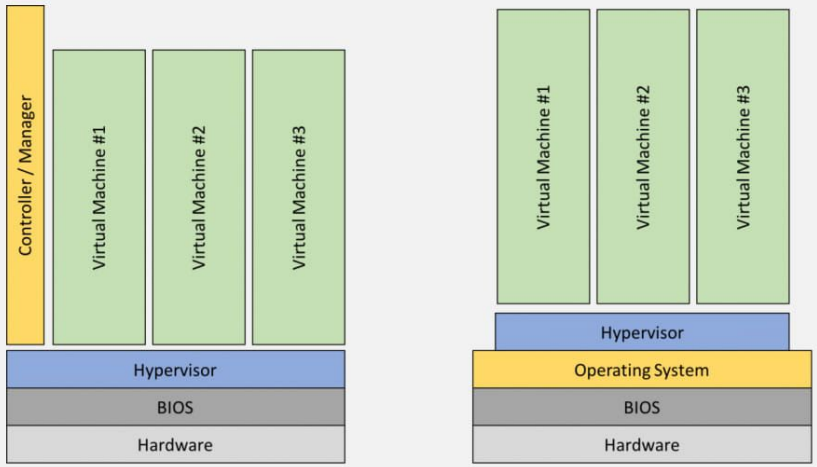
- Virtual Machine Manager
  - Manages the virtual platform and guest OSes
- May require a CPU that supports virtualization
  - Can improve performance
- Hardware management - CPU, networking, security

Cloud and Virtualization Overview

Hypervisors

- Type I - Bare metal, Embedded, Native
  - Run directly with hardware, no additional OS needed
- Type II - Run on a host OS
- Application containerization
  - Run an application without launching an entire VM
  - Everything you need to run the app is in the image (container/cell)

Cloud and Virtualization



Virtualization Security

Virtualization Security

VM sprawl avoidance

- Click a button and you've built an infrastructure
  - It becomes almost too easy to build instances
- The virtual machines are sprawled everywhere
  - You aren't sure which VMs are related to which applications
  - It becomes extremely difficult to deprovision
- Formal process and detailed documentation
  - You should have information on every virtual object

Virtualization Security

VM escape protection

- The virtual machine is self-contained
  - There's no way out - Or is there?
- Virtual machine escape
  - Break out of the VM and interact with the host operating system or hardware
- Once you escape the VM, you have great control
  - Control the host and control other guest VMs
- This would be a huge exploit
  - Full control of the virtual world

Virtualization Security

Escaping the VM

- March 2017 - Pwn2Own hacking contest
  - You pwn it, you own it - along with some cash
- JavaScript engine bug in Microsoft Edge
  - Code execution in the Edge sandbox
- Windows 10 kernel bug
  - Compromise the guest operating system
- Hardware simulation bug in VMware
  - Escape to the host
- Patches were released soon afterwards

Cloud Deployment Models

Cloud Deployment Models

Platform as a Service (PaaS)

- No servers, no software, no maintenance team
  - Someone else handles the platform, you handle the product
- You don't have control of the data, people, or infrastructure
- SalesForce.com is an example of PaaS

Software as a service (SaaS)

- On-demand software, no local installation
- Used for common business functions such as payroll services
- Data and applications are centrally managed
- Gmail and Google Docs is an example of SaaS

Cloud Deployment Models

Infrastructure as a service (IaaS)

- Sometimes called Hardware as a Service (HaaS)
- Equipment is outsourced
- You are still responsible for the overall device and application management - and security
- Web hosting and email services

Cloud Deployment Models

- Private - A virtualized data center
- Public - Available to everyone over the Internet
- Hybrid - A mix of public and private
- Community - Organizations share the same resources

Security in the Cloud

Security in the Cloud

On-premises, hosted, and cloud

- On-premises - Your applications are on local hardware
  - Your servers are in your data center in your building
- Hosted - Your servers are not in your building
  - They may not even be running on your hardware
  - Usually a specialized computing environment
- Cloud - Entire application instances can be created and torn down on-demand
  - Resources are available as needed

Security in the Cloud

Cloud storage

- Data is available anywhere, anytime, on any device
  - If you have a network, you have your data
- Integrates with your enterprise authentication
  - Use your network login to access your data
  - Can include two factor authentication (2FA)
- Encryption is required
  - Data is not under your direct control
  - Strong encryption mechanisms are critical

Security in the Cloud

VDI (Virtual Desktop Infrastructure)

- Virtualize the user's desktop and run it in the data center
  - Sometimes called VDE (Virtual Desktop Environment)
- All of the computing power is in the data center
  - The end-user hardware is a "virtual desktop"
  - Relatively small computing requirements on the client workstation
  - The end-user operating system becomes less important
- Enhanced security
  - Centralized and easier to manage
  - Changes can be tightly controlled
  - The data never leaves the data center

Security in the Cloud

Cloud access security broker (CASB)

- Clients are at work, data is in the cloud
  - How do you keep everything secure?
  - The organization already has well-defined security policies
- How do you make security policies work in the cloud?
  - Integrate a CASB
  - Implemented as client software, local security appliances, or cloud-based security solutions
- Visibility - Determine what apps are in use
  - Are they authorized to use the apps?
- Compliance - Are users complying with HIPAA? PCI?
- Threat prevention
  - Allow access by authorized users, stop attacks
- Data security
  - Ensure that all data transfers are encrypted
  - Protect the transfer of PII with DLP

Security in the Cloud

Security as Service (SECaaS)

- Instead of managing your own security solution, move it to the cloud
  - Pay for what you use, Scale as needed
- Continuously monitoring
  - Uniformly applies to all traffic
- Anti-virus/anti-malware signatures are constantly updated
  - Block emerging threats without deploying updates

Resiliency and Automation

Resiliency and Automation

Automation and scripting

- Plan for change - Implement automatically
- Automated courses of action
  - Many problems can be predicted
  - Have a set of automated responses
- Continuous monitoring
  - Check for a particular event, and then react
- Configuration validation
  - Cloud-based technologies allow for constant change
  - Automatically validate a configuration before going live
  - Perform ongoing automated checks

Resiliency and Automation

Templates

- Cloud orchestration relies on templates
  - Define the basic structure of an application instance
  - The blueprint of your online services
- Application instance requires a web server and a database server
  - Web server needs a specific Apache HTTP server version, a specific PHP version, SSL certs, firewall with predefined rules, brute force. monitoring, etc.
- The template is just the start
  - Each application instance needs to be customized
  - Orchestrate with scripts and APIs

Resiliency and Automation

Master Image

- Instead of building the server each time, create a customized image
  - Build the perfect deployment
  - Save it as a master image
- You'll still need to make changes when deploying
  - IP addresses, firewall rules, licensing updates
- You'll have to keep the master image updated
  - Security patches, operating system updates, etc.
  - This can be administratively challenging

Resiliency and Automation

Non-persistence

- The cloud is always in motion
  - App instances are constantly built and torn down
- Snapshots capture the current configuration and data
  - Preserve the complete state of a device, or just the configuration
- Revert to known state
  - Fall back to a previous snapshot
- Rollback to known configuration
  - Don't modify the data, but use a previous configuration
- Live boot media
  - Run the operating system from removable media

Resiliency and Automation

Elasticity and scalability

- Elasticity - Provide resources when demand requires it
  - Scale down when things are slow
- Host availability
  - New server deployed with a few mouse clicks
- Virtualization integrates a layer of orchestration
  - Automate the deployment and movement of virtual hosts
- Servers can be added or moved to other data centers
  - All of the management systems follow the servers

Redundancy, Fault Tolerance, and High Availability

Redundancy, Fault Tolerance, and High Availability

Distributive allocation

- The bad guys are looking for your data
  - Most people keep it in the data center
- Web servers, database servers, middleware, security devices, monitoring systems
  - Many devices are required to maintain an application instance
- Don't keep everything in one place
  - Critical assets, data, and other system should be in different places
  - Makes it more difficult to target and exploit an application instance
  - A distributive allocation

Redundancy, Fault Tolerance, and High Availability

Redundancy and fault tolerance

- Maintain uptime
  - The organization continues to function
- No hardware failure - Servers keep running
- No software failure - Services always available
- No system failure - Network performing optimally

Redundancy, Fault Tolerance, and High Availability

High availability

- Redundancy doesn't always mean always available
  - May need to be enabled manually
- HA (high availability)
  - Always on, always available
- May include many different components working together
  - Watch for single points of failure

Redundancy, Fault Tolerance, and High Availability

Redundancy and fault tolerance (cont.)

- Redundant hardware components
  - Multiple devices, load balancing power supplies
- RAID
  - Redundant Array of Independent Disks
- Uninterruptible power supplies (UPS)
  - Prepare for the disconnections
- Clustering
  - A logical collective of servers (downtime is futile)
- Load balancing
  - Shared load across components

Redundancy, Fault Tolerance, and High Availability

RAID Level	Description	Details
RAID 0	Striping without parity	High performance, no fault tolerance
RAID 1	Mirroring	Duplicates data for fault tolerance, but requires twice the disk space
RAID 5	Striping with parity	Fault tolerant, only requires an additional disk for redundancy
RAID 0+1, RAID 1+0, RAID 5+1, etc.	Multiple RAID types	Combine RAID methods to increase redundancy

Physical Security Controls

Physical Security Controls

Proper Lighting

- More light means more security
  - Bad guys avoid the light
  - Easier to see when lit
  - Non IR cameras can see better
- Specialized design
  - Consider overall light levels
  - Lighting angles may be important - Facial recognition
  - Avoid shadows and glare

Physical Security Controls

Signs

- Clear and specific instructions
  - Keep people away from restricted areas
  - Consider visitors
- Consider personal safety
  - Fire exits
  - Warning signs (i.e., chemicals, construction)
  - Medical resources
- Informational
  - In case of emergency, call this number

Physical Security Controls

Fencing

- Build a perimeter
  - Usually very obvious
  - May not be what you're looking for
- Transparent or opaque
  - See through the fence (or not)
- Robust
  - Difficult to cut the fence
- Prevent climbing
  - Razor wire
  - Build it high

Physical Security Controls

Rack monitoring and security

- Monitoring systems
  - Environmental sensors
  - Webcams
  - Integration with enterprise monitoring systems
- Security
  - Closed racks
  - Locks (with keys!)
  - Fences and gates

Physical Security Controls

Guards and access lists

- Security guard
  - Physical protection
  - Validates identification of existing employees
  - Provides guest access
- ID badge
  - Picture, name, other details
  - Must be worn at all times
- Access list
  - Physical list of names
  - Enforced by security guard

Physical Security Controls

Alarms

- Circuit-based
  - Circuit is opened or closed
  - Door, window, fence
  - Useful on the perimeter
- Motion detection
  - Radio reflection or passive infrared
  - Useful in areas not often in use
- Duress
  - Triggered by a person
  - The big red button

Physical Security Controls

Safe

- Secure your important hardware and media
  - Backups, laptops, hard drives
- Protection against the elements
  - Fire, water
- Difficult to steal
  - Very heavy
- Must be carefully managed
  - Don't share the combination
  - What happens when you lose the combination?

Physical Security Controls

Locking cabinets

- Data center hardware is often managed by different groups
  - Responsibility lies with the owner
- Racks can be installed together
  - Side-to-sides
- Enclosed cabinets with locks
  - Ventilation on front, back, top, and bottom

Physical Security Controls

Protected distribution

- Protected Distribution System (PDS)
  - A physically secure cabled network
- Protect your cables and fibers
  - All of the data flows through these conduits
- Prevent cable and fiber taps
  - Direct taps and inductive taps
- Prevent cable and fiber cuts
  - A physical denial of service (DoS)
- Hardened protected distribution system
  - Sealed metal conduit, periodic visual inspection

Physical Security Controls

Air gap

- Physical separation between networks
  - Secure network and insecure network
  - Separate customer infrastructures
- Most environments are shared
  - Shared routers, switches, firewalls
  - Some of these are virtualized
- Specialized networks require air gaps
  - Stock market networks
  - Power systems/SCADA
  - Airplanes
  - Nuclear power plant operations

Physical Security Controls

Mantraps

- All doors normally unlocked
  - Opening one door causes others to lock
- All doors normally locked
  - Unlocking one door prevents others from being unlocked
- One door open / other locked
  - When one is open, the other cannot be unlocked
- One at a time, controlled groups
  - Managed control through an area

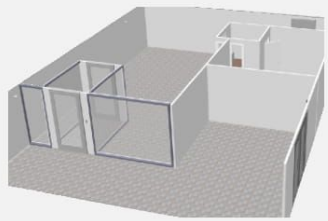
Physical Security Controls

Faraday cage

- Blocks electromagnetic fields
  - Discovered by Michael Faraday in 1836
- A mesh of conductive material
  - The cage cancels the electromagnetic field's effect on the interior
  - The window of a microwave oven
- Not a comprehensive solution
  - Not all signal types are blocked
  - Some signal types are not blocked at all
- Can restrict access to mobile networks
  - Some very specific contingencies would need to be in place for emergency calls

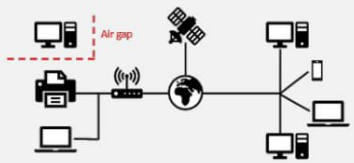
Physical Security Controls

Mantrap

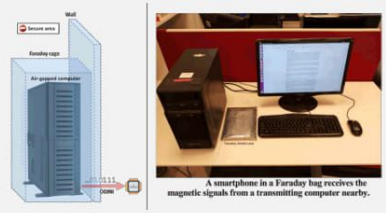


Protected Distribution

Airgap



Faraday Cages



Physical Security Controls

Door access controls

- Conventional
  - Lock and key
- Deadbolt
  - Physical bolt
- Electronic
  - Keyless
- Token-based
  - Magnetic swipe card or proximity reader
- Multi-factor
  - Smart card and PIN

Physical Security Controls

Biometrics

- Biometric authentication
  - Fingerprint, iris, voiceprint
- Usually stores a mathematical representation of your biometric
  - Your actual fingerprint isn't usually saved
- Difficult to change
  - You can change your password
  - You can't change your fingerprint
- Used in very specific situations
  - Not foolproof

Physical Security Controls

Barricades/bollards

- Prevent access
  - There are limits to the prevention
- Channel people through a specific access point
  - And keep out other things
  - Allow people, prevent cars and trucks
- Identify safety concerns
  - And prevent injuries
- Can be used to an extreme
  - Concrete barriers / bollards
  - Moats

Physical Security Controls

Token and cards

- Smart card
  - Integrates with devices
  - May require a PIN
- USB token
  - Certificate is on the USB device
- Hardware or software tokens
  - Generates pseudo-random authentication codes
- Your phone
  - SMS a code to your phone

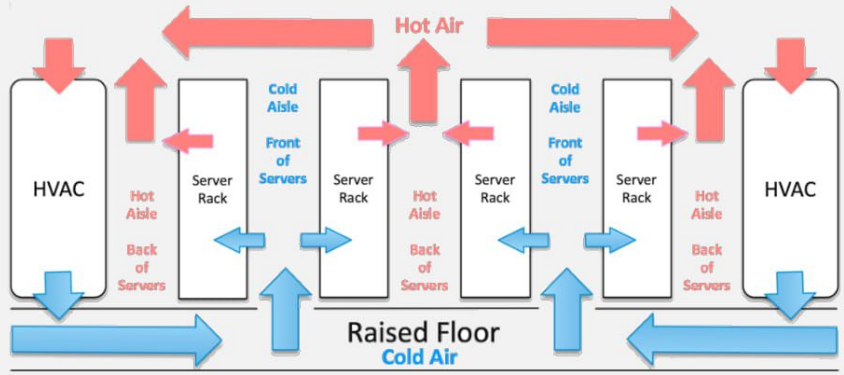
Physical Security Controls

HVAC

- Heating, Ventilating, and Air Conditioning
  - Thermodynamics, fluid mechanics, and heat transfer
- A complex science
  - Not something you can properly design yourself
  - Must be integrated into the fire system
- Data center should be separate from the rest of the building
  - Not too cold, not too hot
  - Overheating is a huge issue
- Closed-loop recirculating and positive pressurization
  - Recycle internal air, and air is pushed out

Physical Security Controls

Hot and cold aisles



Physical Security Controls

Fire suppression

- Electronics require unique responses to fire
  - Water is generally a bad thing
- Detection
  - Smoke detector, flame detector, heat detector
- Suppress with water
  - Dry pipe, wet pipe, pre-action
- Suppress with chemicals
  - Halon - No longer manufactured
  - Dupont FM-200 / American Pacific Halotron

Physical Security Controls

Cable locks

- Temporary security
  - Connect your hardware to something solid
- Cable works almost anywhere
  - Useful when mobile
- Most devices have a standard connector
  - Reinforced notch
- Not designed for long-term protection
  - Those cables are pretty thin

Physical Security Controls

Screen filters

- Control your input
  - Be aware of your surroundings
- Use privacy filters
  - It's amazing how well they work
- Keep your monitor out of sight
  - Away from windows and hallways
- Don't sit in front of me on your flight
  - I can't help myself

Physical Security Controls

Video surveillance

- CCTV (Closed circuit television)
  - Can replace physical guards
- Camera properties are important
  - Focal length - Shorter is wider angle
  - Depth of field - How much is in focus
  - Illumination requirements - See in the dark
- Often many different cameras
  - Networked together and recorded over time

Physical Security Controls

Logs

- Everything is logged
  - Entering the parking area
  - Identification upon entering the building
  - Badge assignment tracks door operation
- Correlate physical location with digital access
  - Someone logged into the console while in the room
- Need a formal process to collect and archive log information
  - Some of the logs are physical, some are digital
  - May fall under privacy laws

Physical Security Controls

Key management

- Cryptographic key management
  - Policies for the creation and protection of important keys
- Some keys should be physically separated from the network
  - Certificate Authority (CA) root key
  - If compromised, all keys must be replaced
  - One good reason for intermediate CAs
- The root CA certificate will only be used to sign other intermediate CAs

CSF 434/534: Advanced Network and System Security

Week 08 - Review

Michael Conti

Department of Computer Science and Statistics
University of Rhode Island

Sources: Professor Messer's CompTIA SY0-501 Security+ Course Notes