CS615 - Aspects of System Administration

System Security

Department of Computer Science
Stevens Institute of Technology
Jan Schaumann
jschauma@stevens.edu
https://stevens.netmeister.org/615/

How to secure a Linux system

https://is.gd/96cQgL

This lecture

What I won't tell you:

- How to make your system "secure".
- How to break into other systems.
- Everything you need to know.

What I will tell you:

- What you need to know to *start looking*.
- What concepts are critical to understand.
- What conceptual pitfalls you are likely to encounter.
- A few *always* and *never*s.

This lecture

Security is not an end-goal.

Security is a trade-off property you may attempt to increase resilience against specific risks.

Where/how does 'security' come into play?

Where/how does 'security' come into play?

Lecture 02 (Filesystems, Disks, Storage)

- storage model (DAS, NAS, SAN, Cloud)
- partitions / mount options
- filesystem features (permissions, access control lists)
- DoS on disk space
- firmware compromise on hard drives

Lecture 03 (Software Installation Concepts)

- software package management and updates
- VMs, containers, etc.
- patch management
- package integrity checking

Where/how does 'security' come into play?

Lecture 04 (Multiuser Fundamentals)

- privileges and trust models
- authentication methods, multi-factor authentication
- file access controls
- raising privileges

Lecture 05 / 06 (Networking)

- protocols and visibility of data on different layers
- tcpdump can read all packets
- location of attacker on network implies capabilities
- network censorship

Where/how does 'security' come into play?

Lecture 07 (DNS; HTTP)

- If you control the DNS, you control the domain
- DNS registrars as attack points
- use of DNS as another channel for host verification (SSHFP records)
- trustworthiness of DNS (DNSSEC)
- HTTP as the universal entry into any network
- code execution context (CGI vs. server-side vs. client-side)
- content control and inspection capabilities of e.g. CDNs

Where/how does 'security' come into play?

Lecture 08 (SMTP, HTTPS)

- observation of packets via `tcpdump(1)`
- email as attack methods (spam, phishing)
- email privacy implications
- SMTP plain text vs. opportunistic encryption
- mail abuse and spam
- recipient and sender authentication, open relays
- TLS authentication
- PKI, Certificate Authorities
- protocol downgrade and MitM attacks

Where/how does 'security' come into play?

Lecture 09 (Writing System Tool)

- automation as a defensive weapon
- using the wrong tool for the job => writing insecure code
- understanding language / framework pitfalls
- simplicity reduces attack surface
- all code has bugs

Where/how does 'security' come into play?

Lecture 10 (Backup and Disaster Recovery, Monitoring)

- disasters include security breaches
- data loss as a risk
- safety of backups (encrypted backups?)
- incident detection via events, metrics, and context
- sensitive data in logs
- outsourcing monitoring services

Lecture 11 (Configuration Management)

- role based access control
- inherent trust, full control
- CAP theorem may impact security controls

What is security?

```
security

 NOUN:

    Freedom from risk or danger; safety.
```

What is risk?

```
risk

 NOUN:

  The possibility of suffering harm or loss; danger.
```

Suffering harm or loss of *what*?

- access to data
- integrity of data
- availability of services
- reputation
- monetary loss due to any of the above
- monetary loss due to physical items of actual value
- ...

How to determine *risk*

"Risk Assessment"

- identify *assets* (that which you wish to protect, what you *value*)

How to determine *risk*

"Risk Assessment"

- identify *assets*
- identify *threats* (possible dangers to your assets, bad things that might happen)

How to determine *risk*

"Risk Assessment"

- identify *assets*
- identify *threats*
- identify *vulnerabilities* (weaknesses in a system, component, protocol, ...)

How to determine *risk*

"Risk Assessment"

- identify *assets*
- identify *threats*
- identify *vulnerabilities*
- determine *likelihood of damage* (considering mitigating or exacerbating factors)

How to determine *risk*

"Risk Assessment"

- identify *assets*
- identify *threats*
- identify *vulnerabilities*
- determine *likelihood of damage*
- estimate *cost of recovery* (including recovery of data, immediate revenue loss, replacing physical items, ...)

How to determine *risk*

"Risk Assessment"

- identify *assets*
- identify *threats*
- identify *vulnerabilities*
- determine *likelihood of damage*
- estimate *cost of recovery*
- estimate *cost of defense* (objectively, without consideration of your budget; include partial defense or mitigating strategies)

How to determine *risk*

"Risk Assessment"

- identify *assets*
- identify *threats*
- identify *vulnerabilities*
- determine *likelihood of damage*
- estimate *cost of recovery*
- estimate *cost of defense*

A *risk* is the *likelihood* of a *threat* successfully exploiting a *vulnerability* and the *estimated cost* (or potential damage) both in the short and long term you may incur as a result.

How to determine *risk*

Never waste resources on unspecified, vague risks or FUD.

Always remember that risks are *scoped* and *specific*.

How do we secure a system?

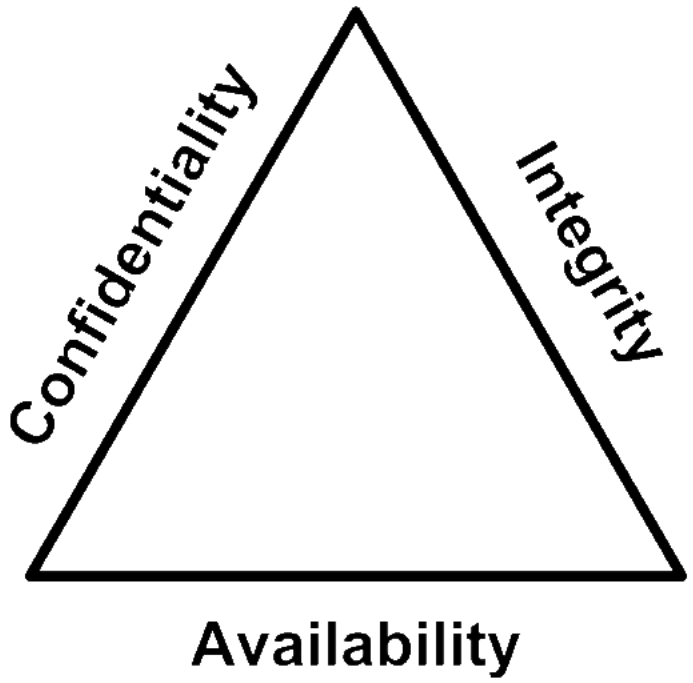
You can't "secure" a system; you can only minimize specific risks by e.g. closing an attack vector, eliminating a vulnerability, reducing the attack surface, or changing the economics of the adversary.

Threat Model

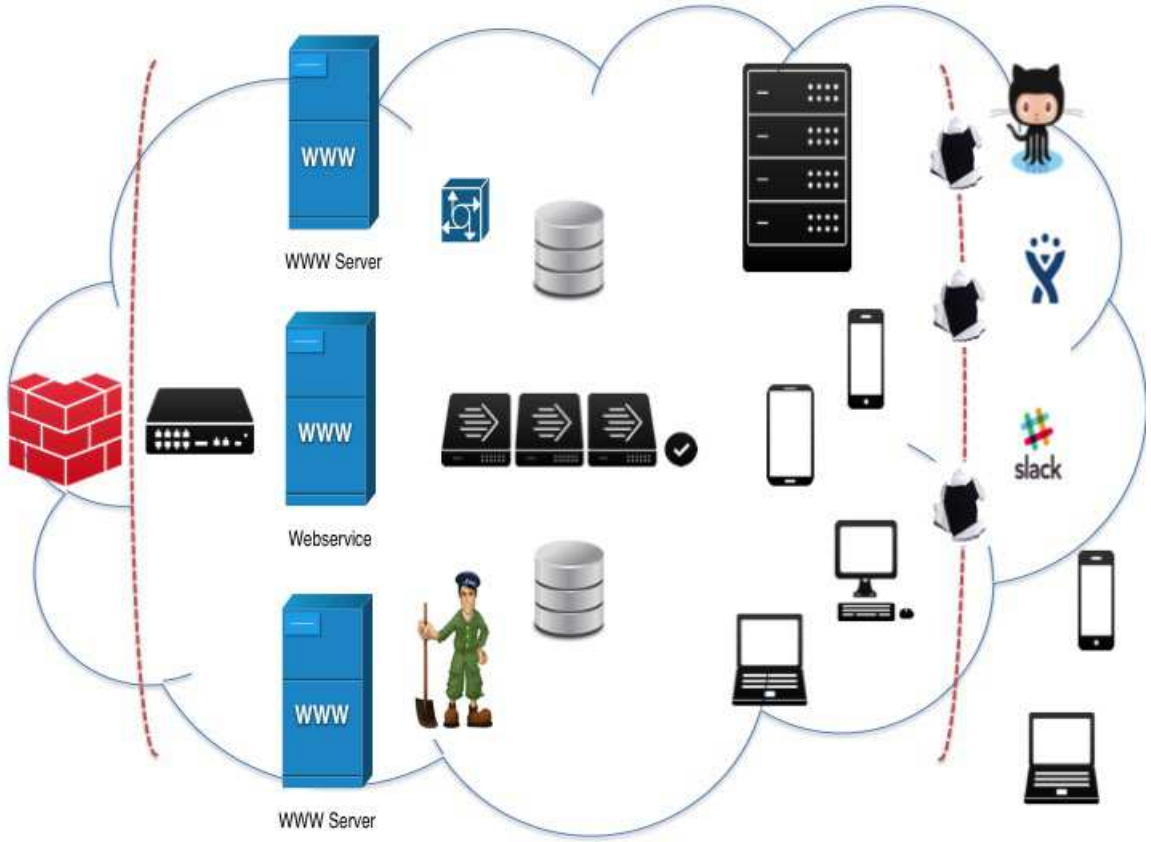
For each system/component/product/service/...

- identify *what* you're protecting
- identify *from whom* you're protecting it
  - identify *goals* of the attacker
  - identify *motivation* of the attacker
  - identify *capabilities* of the attacker
- identify threats you cannot defend against (within this system or in general)

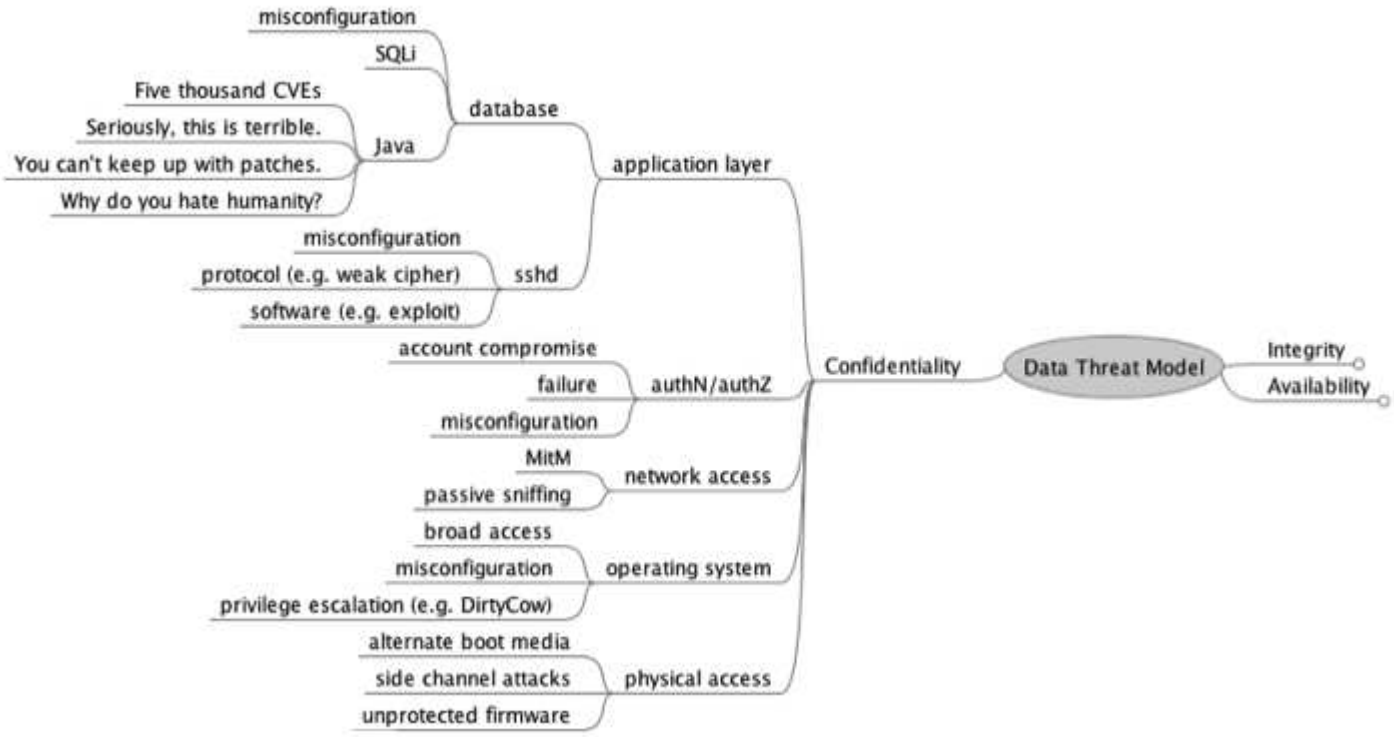
CIA Triad



Threat Model



Threat Model



https://www.netmeister.org/blog/threat-model-101.html.html

Threat Model

Your adversaries are determined human actors with specific goals.

Threat actors have their own risk profile, -tolerance, and cost/benefit calculations.

Imperatives

Constantly seek to reduce your attack surface.
Identify and eliminate attack vectors.

You can't do this alone:
lead by example, seek allies.

Imperatives

Never think you're the only one who understands or cares about security.

Always consult with subject matter experts, especially those *not* on your team.

Defense in Depth

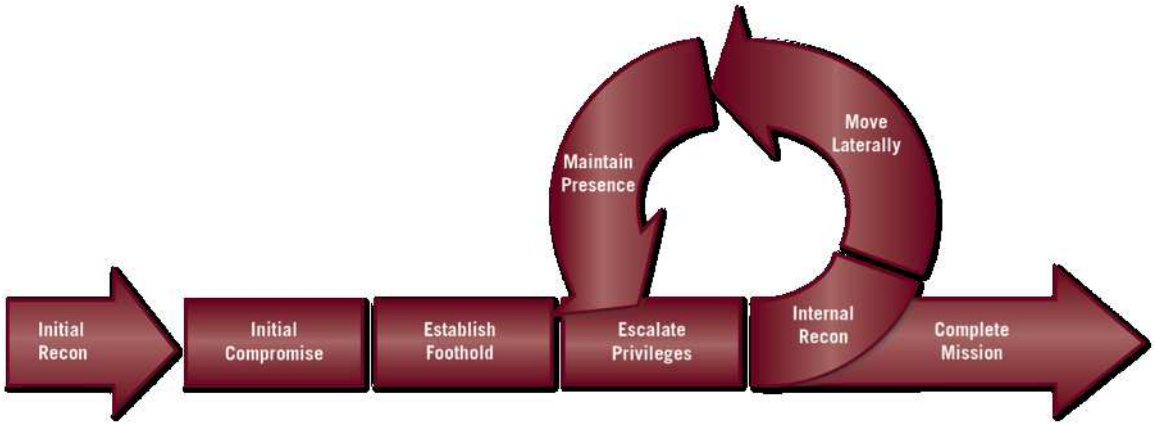
Security is like an onion: the more layers you peel away, the more it stinks.

Never assume any one protection mechanism is sufficient.

Always assume the other protections you deployed can be circumvented or broken.

The Attack Life Cycle

...or why you need to think about *Zero Trust*.



http://is.gd/6sREQh

https://www.netmeister.org/blog/attack-life-cycle.html

Zero Trust

- not a product, but a model
- assume all networks are hostile
- users and clients are mutually authenticated
- actions are explicitly authorized
- authorizations are tightly scoped
- privileges are role based and time limited

Cryptography

Cryptography can help mitigate *some* of the risks *sometimes*.

Cryptography

Cryptography can help mitigate *some* of the risks *sometimes*.

It may provide security in the areas of:

- Secrecy or Confidentiality
  - *Did/could anybody else see (parts of) the message?*

Cryptography

Cryptography can help mitigate *some* of the risks *sometimes*.

It may provide security in the areas of:

- Secrecy or Confidentiality
  - *Did/could anybody else see (parts of) the message?*
- Accuracy or Integrity
  - *Was the message (could it have been) modified before I received it?*

Cryptography

Cryptography can help mitigate *some* of the risks *sometimes*.

It may provide security in the areas of:

- Secrecy or Confidentiality
  - *Did/could anybody else see (parts of) the message?*
- Accuracy or Integrity
  - *Was the message (could it have been) modified before I received it?*
- Authenticity
  - *Is the party I'm talking to actually who I think it is / they claim they are?*

Cryptography

Note:

- *Never* write your own crypto or invent your own protocol.
- *Authentication* != *Authorization*
- cryptography does not handle authorization
- you generally need all three: confidentiality, integrity, authenticity
- cryptography cannot prevent against incorrect use
  – usability is hard!

Know your threat model!

Basic Security Concepts: Confidentiality

- Alice and Bob agree on a way to transform plain text into ciphertext
- transformed data is sent over insecure channel
- Alice and Bob are able to reverse transformation

Different approaches:

- secret key cryptography (example: *DES*)
  - Alice and Bob share a secret key (e.g. WEP, WPAPSK, ...)
- public key cryptography (example: *RSA*)
  - Alice has a private and a public key (e.g. TLS, SSH, PGP, ...)
  - data encrypted with her private key can only be decrypted by her public key and vice versa
  - public key can be shared with anybody (via insecure means)

Confidentiality

```
$ curl http://random-place-on-the-internet/some-file | sudo bash
```

Confidentiality

```
$ curl http://random-place-on-the-internet/some-file | sudo bash
```

vs.

```
$ curl -k https://random-place-on-the-internet/some-file | sudo bash
```

Threats to Confidentiality

- lack of *authenticity*
- key exchange
- lack of key rotation
- key disclosure

Never store secrets in code!
Always use a key management system.

Basic Security Concepts: Integrity

In order to protect against forgery or data manipulation, provide some sort of digest or checksum (often a one-way hash). Popular choices:

- `5f4dcc3b5aa765d61d8327deb882cf99` (MD5)
- `5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8` (SHA-1)
- `5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62 a11ef721d1542d8` (SHA256)
- `b109f3bbbc244eb82441917ed06d618b9008dd09b3befd1b5 e07394c706a8bb980b1d7785e5976ec049b46df5f1326af5a 2ea6d103fd07c95385ffab0cacbc86` (SHA512)

Note: MD5 and SHA1 are no longer be acceptable choices for cryptographic integrity.

See also:
`https://latacora.singles/2018/04/03/cryptographic-right-answers.html`

Integrity

```
$ curl -k https://random-place-on-the-internet/some-file | sudo bash
```

Integrity

```
$ curl -k https://random-place-on-the-internet/some-file | sudo bash
```

vs.

```
$ curl -k https://random-place-on-the-internet/some-file >file
$ curl -k https://random-place-on-the-internet/some-file.md5 >file.md5
$ diff <(md5 <file) <(file.md5) && cat file | sudo bash
```

Basic Security Concepts: Integrity

Examples: host based IDS, package manager signatures

Some possible threats:

- collisions in algorithm
- lack of *authenticity* (Where did I get the checksum?)
- lack of *integrity* (Was the checksum tampered to match the (tampered) data?)
- “verification” with compromised tools
- “rainbow tables” / internet search engines allow for easy reverse lookup of un-salted hashes.

Basic Security Concepts: Hashing Passwords

Never confuse *hashing* and *encryption*!

Never encrypt your users' passwords to store them – *always* hash them.

Always salt your hashes.

Always use adaptive or key-stretching functions such as *e.g.*: `argon2`, `bcrypt`, `PBKDF2`, `scrypt`

Here, $DK = KDF(key, salt, iterations)$, where $salt$ and $iterations$ are public.

Basic Security Concepts: Authenticity

Three general ways of proving that you are who you say you are:

- something you know
- something you have
- something you are

Basic Security Concepts: Authenticity

Three general ways of proving that you are who you say you are:

- something you know
  - secret handshake, password
  - can (easily) be given to and used by somebody else
- something you have
- something you are

Basic Security Concepts: Authenticity

```
NetBSD/amd64 (SERVER) (console)

login: jschauma
password: **********************************
NetBSD 7.0.2 (SERVER) #2: Tue Jan 24 02:33:13 EST 2017

Welcome to NetBSD!
hostname$
```

Basic Security Concepts: Authenticity

Three general ways of proving that you are who you say you are:

- something you know
  - secret handshake, password
  - can (easily) be given to and used by somebody else
- something you have
  - physical items: smart card, RSA token, ...
  - private keys
  - can (easily) be given to and used by somebody else
- something you are

Basic Security Concepts: Authenticity

```
$ ssh-keygen -l -f /dev/stdin <<<$(aws ec2 get-console-output \
        i-0990f1eb069c853c4 | grep ^ecdsa)
256 19:af:35:01:0b:2a:ee:3d:30:0f:69:11:cc:55:7c:20 (ECDSA)
$ ssh -i ~/.ssh/myawskey ec2-54-227-16-184.compute-1.amazonaws.com
The authenticity of host 'ec2-54-227-16-184.compute-1.amazonaws.com
(54.227.16.184)' can't be established.
ECDSA key fingerprint is 19:af:35:01:0b:2a:ee:3d:30:0f:69:11:cc:55:7c:20.
Are you sure you want to continue connecting (yes/no)?  yes
NetBSD 7.0.2 (SERVER) #2: Tue Jan 24 02:33:13 EST 2017

Welcome to NetBSD!
hostname$
```

Basic Security Concepts: Authenticity

Three general ways of proving that you are who you say you are:

- something you know
  - secret handshake, password
  - can (easily) be given to and used by somebody else
- something you have
  - physical items: smart card, RSA token, ...
  - private keys
  - can (easily) be given to and used by somebody else
- something you are
  - physical, physiological or behavioral traits
  - cannot (easily or at all) be given to or used by somebody else
  - cannot (easily or at all) be changed once compromised

Basic Security Concepts: Authenticity

Authenticity

```
$ curl -k https://random-place-on-the-internet/some-file >file
$ curl -k https://random-place-on-the-internet/some-file.md5 >file.md5
$ diff <(md5 <file) <(file.md5) && cat file | sudo bash
```

Authenticity

```
$ curl -k https://random-place-on-the-internet/some-file >file
$ curl -k https://random-place-on-the-internet/some-file.md5 >file.md5
$ diff <(md5 <file) <(file.md5) && cat file | sudo bash
```

vs.

```
$ curl -k https://random-place-on-the-internet/some-file >file
$ curl -k https://random-place-on-the-internet/some-file.sig >file.sig
$ gpg --verify file.sig file && cat file | sudo bash
```

Basic Security Concepts: Authenticity

Some possible threats:

- lack of *confidentiality*
- lack of *integrity*
- reliance on fragile infrastructure
- usability
- conflation with *authorization*

Principle of Least Privilege

Black Team Links:

`https://is.gd/y0GkhU`, `https://is.gd/ZBnl1P`

Principle of Least Privilege

Never run services as root; *always* use a dedicated account.

Never log in as root; *always* use `sudo(1)`.

Never rely on implicit privileges; *always* grant access explicitly.

Never grant permanent overly broad access; *always* use periodic access renewal and Role Based Access Controls (RBAC).

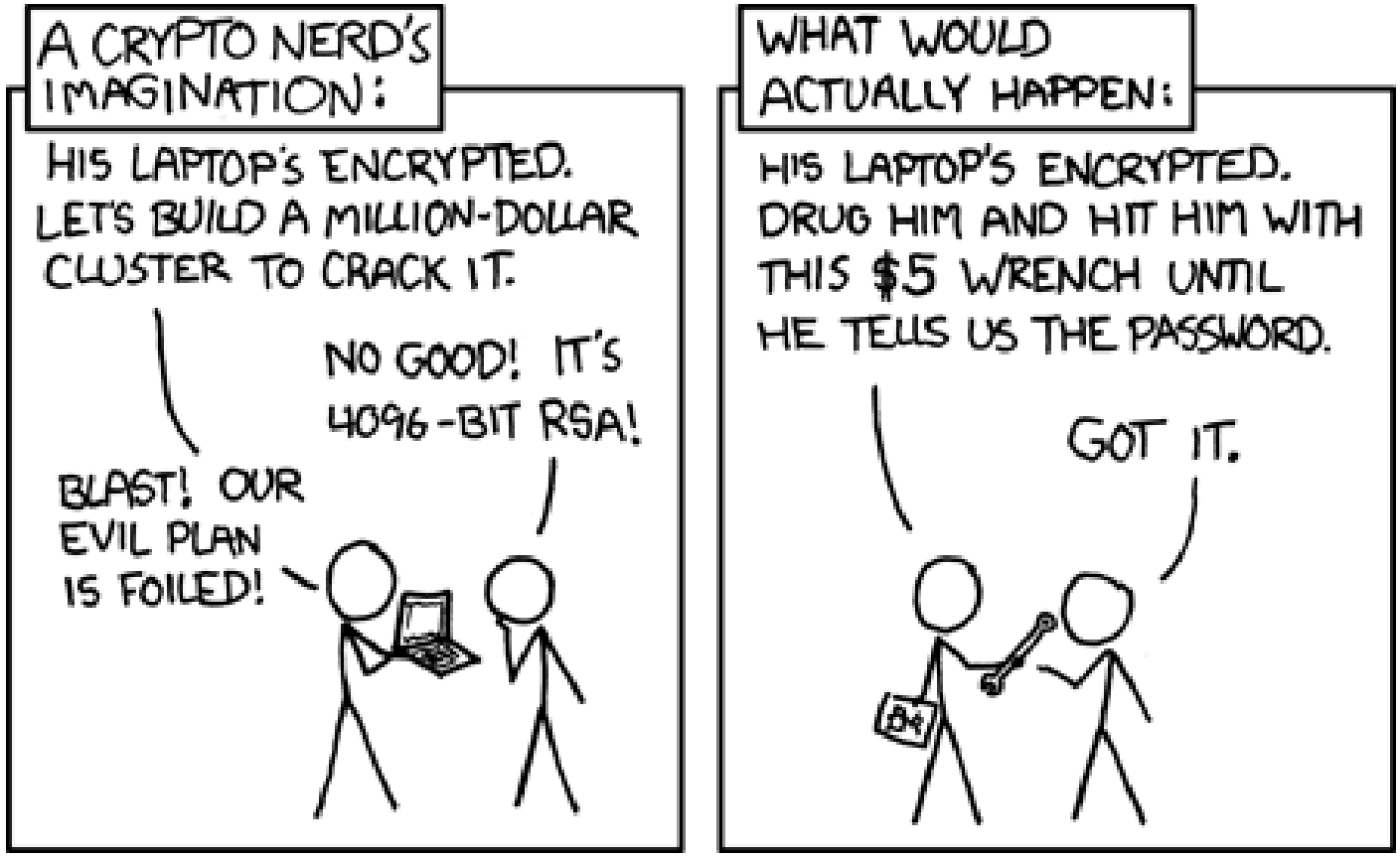
It's not just 1s and 0s

System security is not restricted to *software* security.

It's not just 1s and 0s

The thing that makes security difficult is not the software or hardware components. It's the human component.

It's not just 1s and 0s



Secure by default

Users care about usability, not about security.

Secure by default

Users will not change their default settings.

Secure by default

Users will not change their default settings.

(Unless a less secure option is available.)

Secure by default

https://is.gd/L2H9k4

Hooray!

5 Minute Break

(Some) Classes of Vulnerabilities

- memory management
  - use of uninitialized memory
  - buffer overflow / stack smashing
  - use-after-free / dangling pointer
- input validation
  - code and command injections
  - format attacks
  - Little Bobby Tables (`https://www.xkcd.com/327/`)
- race contitions
  - non-atomic TOCTOU
  - symlink attacks

(Some) Classes of Vulnerabilities

- privilege escalation and confusion
  - XSS, CSRF
  - setuid with untrusted environment
- social engineering
  - phishing
  - watering hole attacks
- brute-force attacks
  - namespace iteration
  - denial of service
- information disclosure
  - MitM
  - insufficient permissions
  - lack of encryption, authN, authZ

Security Fallacies and Pitfalls

Security by Obscurity

Security Fallacies and Pitfalls

Know what you're doing.

Never blindly apply nor dismiss a security mechanism.

Always know which threat you're mitigating.

Security Fallacies and Pitfalls

Perfect is the Enemy of the Good

(Differentiate between futile efforts and raising the bar.)

Security Fallacies and Pitfalls

One in a million is next Tuesday.

`https://is.gd/Isb20K`

Security Fallacies and Pitfalls

"Any person can invent a security system so clever that she or he can't think of how to break it."

Schneier's Law `https://is.gd/hW82dt`

Security Fallacies and Pitfalls

Don’t invent your own crypto.

(Seriously, don’t.)

Security Fallacies and Pitfalls

Complexity is the worst enemy of security.

(The more secure you make something, the less secure it becomes.)

Whom do you trust?

Reflections on Trusting Trust

https://is.gd/RUX4zY

Outsourcing Services

- you trust the provider/vendor to honor the agreement
- you “hope” they won’t change their agreement (once invested, changing back is hard; vendor lock-in)
- you trust the provider/vendor to keep their infrastructure safe
- you trust the provider/vendor’s employees
- you are ok with the traffic going across the public internet

Outsourcing Services

- you trust the provider/vendor to honor the agreement
- you “hope” they won’t change their agreement (once invested, changing back is hard; vendor lock-in)
- you trust the provider/vendor to keep their infrastructure safe
- you trust the provider/vendor’s employees
- you are ok with the traffic going across the public internet

Bottom-line: are you increasing or decreasing your attack surface?

Always make a conscious decision; *never* blindly follow the promises without understanding the trade-offs.

Bug Bounty Programs

https://is.gd/2Eb26A, https://is.gd/WAqf6O, https://is.gd/Tyu7Hu

https://www.netmeister.org/blog/bug-bounty.html

Embrace Automation

Vulnerabilities are dense.

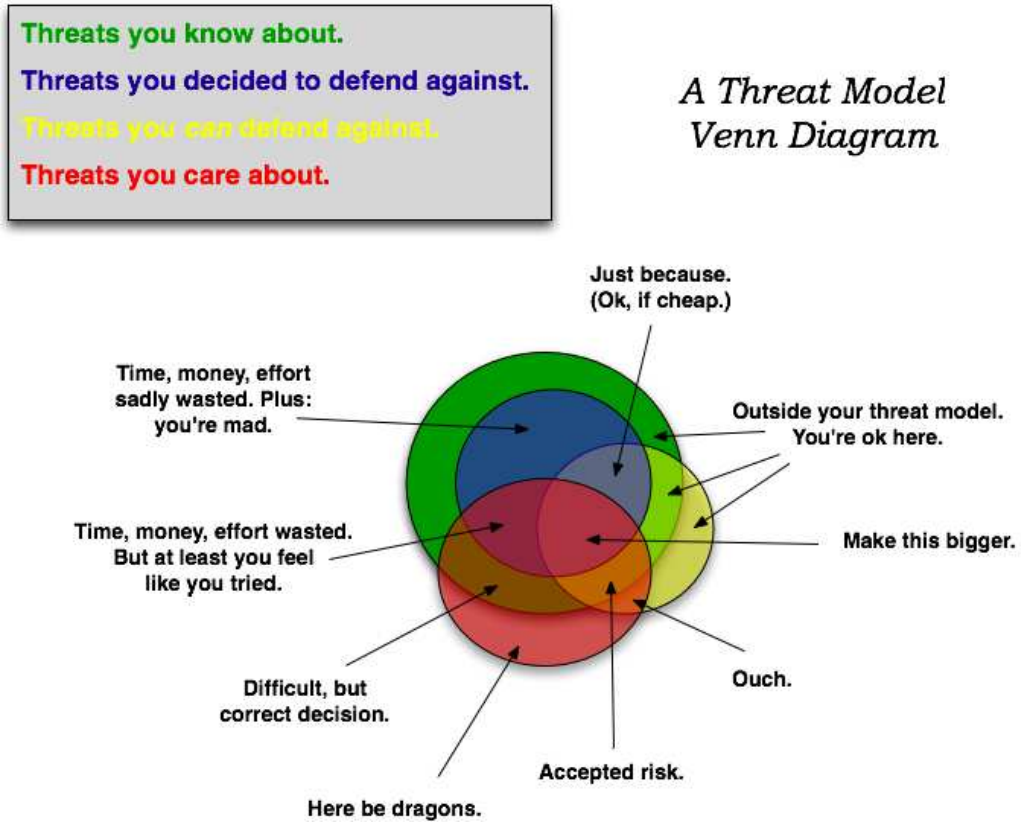
Eliminate *classes* of attacks, not individual flaws.

Build Robust Infrastructures and Service

Your endpoint security model should assume the network is compromised;
your network security model should assume the endpoint is.

Both in fact are.

Know Your Threat Model



https://www.netmeister.org/blog/threat-model-101.html.html

Last Words of Advice: Don't be lazy!

- keep your asset inventory accurate
- don't shell out; parametrize arguments and `exec(3)`
- don't trust the environment
- use multi-factor authentication
- use a password manager
- use a key management system
- rotate your secrets frequently
- `curl -k` is a (contagious) symptom
- don't MitM your own users
- disable Flash; use an ad-blocker
- sign your software, configs; verify all signatures
- ensure secure defaults (e.g. umask, shell history, ...)

Where/how does 'security' come into play?

Additional Reading

https://www.slideshare.net/zanelackey/attackdriven-defense
https://www.netmeister.org/blog/moving-the-needle.html
https://www.netmeister.org/blog/attack-life-cycle.html
https://www.netmeister.org/blog/threat-model-101.html
https://twitter.com/jschauma/status/713118376550404096
https://t.co/DRHbEKXod8
https://danielmiessler.com/study/security_and_obscurity/
https://is.gd/sGnRVL
https://www.netmeister.org/blog/two-questions.html