What Every Developer Should Know about Computer Security

Tom Wheeler

What I'm Going to Cover

- I'll explain what computer security is and why it matters
- I'll teach you how to think about it
- I'll describe some common attacks & protection measures
- There is a *lot* to cover in this area... and limited time
  - I will focus on what programmers can generally affect
  - Not system administrators or network administrators
  - No IDS, firewalls, DNS exploits, DDOS attacks, etc.

What is Computer Security?

Practical UNIX and Internet Security said it best:

> *A computer is secure if you can depend on it and its software to behave as you expect.*

What is Computer Security?

Not just about wily attackers attacking you from afar, but also:

- Disgruntled employees
- Fully-gruntled but inept employees
- Unscrupulous vendors
- Flaky hardware
- Natural disasters
- Godzilla!

Why Is Computer Security Important?

Software systems are increasingly:

- Interconnected with other systems
- Exposed to a wider audience
- Poorly documented
- Administered by people who don't understand them
- Complex & brittle
- Essential to a company's core business

Why Is Computer Security Important?

There's too much to lose.

Computer security is not someone else's problem. It's yours.

And Now for the Bad News

- Your systems will never **be secure** (and still be useful)
- All you can hope is that they're *more secure* than before
- It's infeasible to protect against every threat
- You must analyze which ones warrant your attention

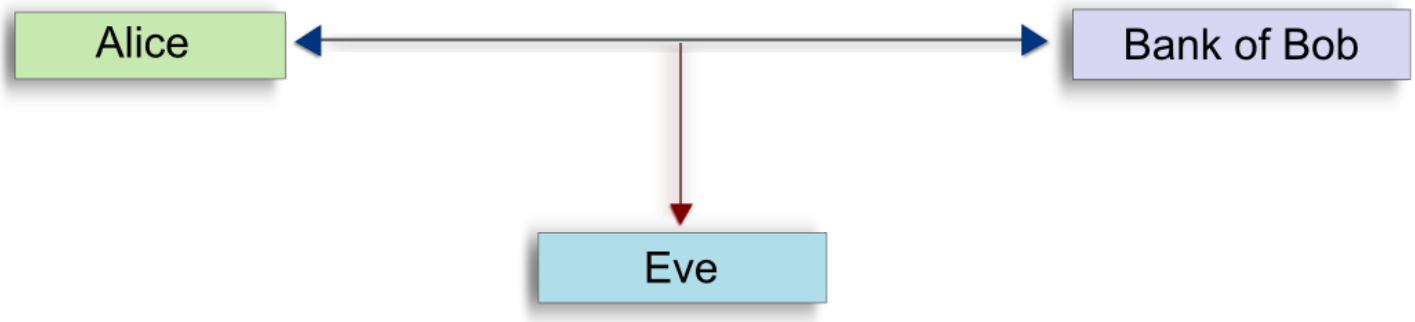
Concepts

Computer security usually involves some combination of the following concepts:

Concept: Confidentiality

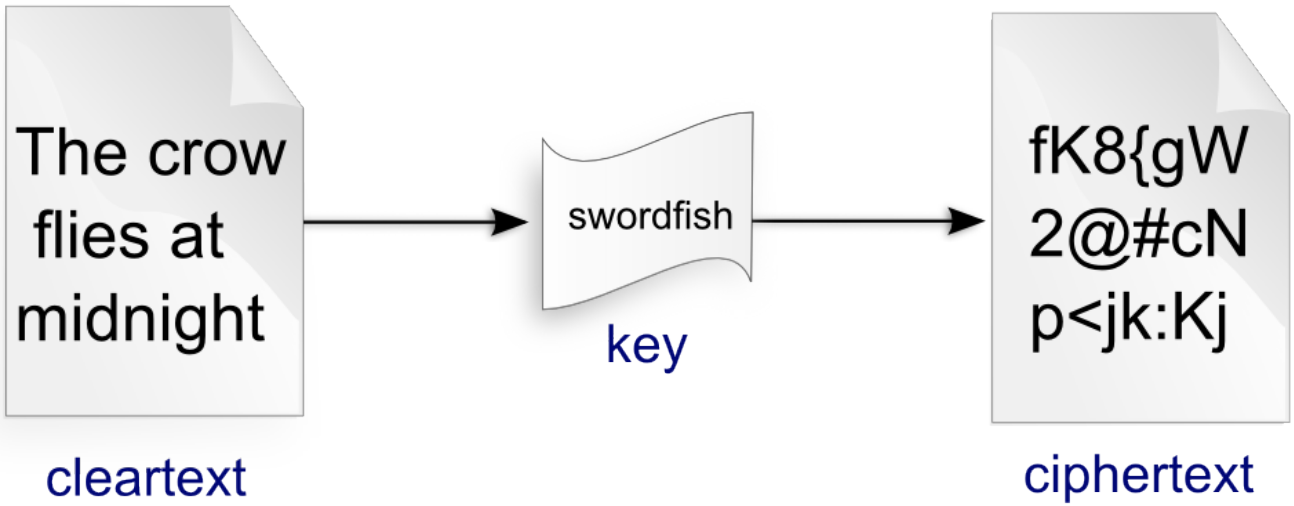
Data should be accessible only by the intended parties.

Typically implemented with cryptography.



Cryptography in 60 Seconds

- Cryptography deals with ciphers
- Kerckhoffs's Principle: Only the key needs to be secret
- Modern crypto is all mathematics
- Symmetric / Asymmetric / Hybrid

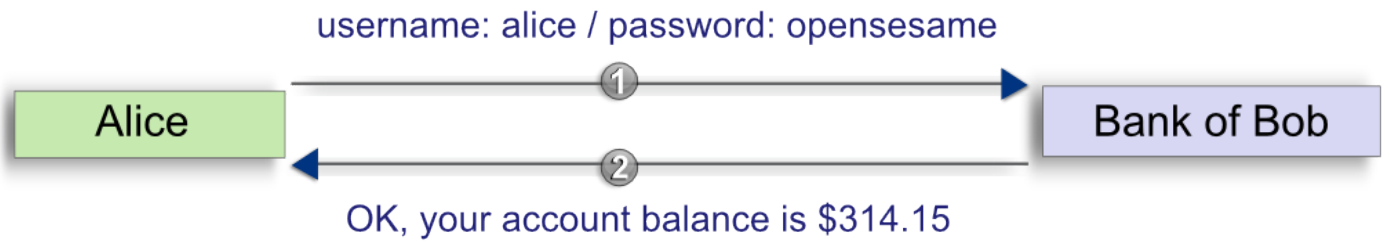


Concept: Authentication

Participants must prove their identity.

Typically implemented with one or more of the following:

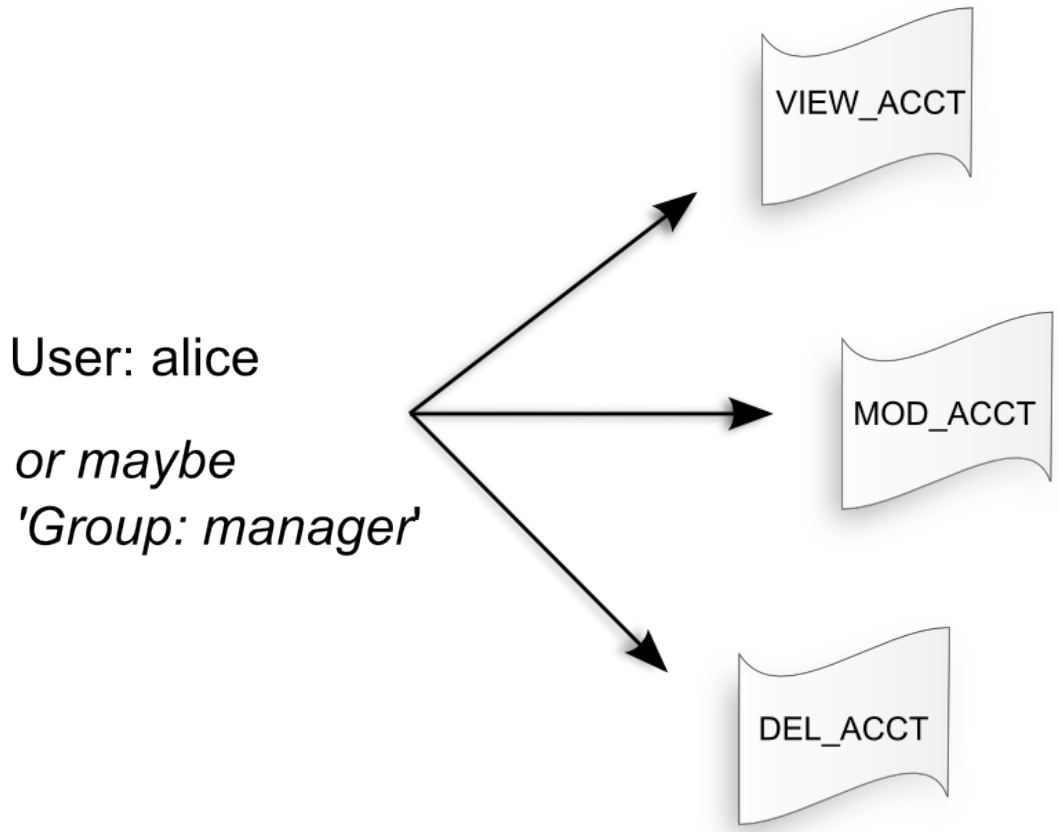
- What you know
- What you have
- What you are



Concept: Authorization

An actor must be granted permission to perform an action.

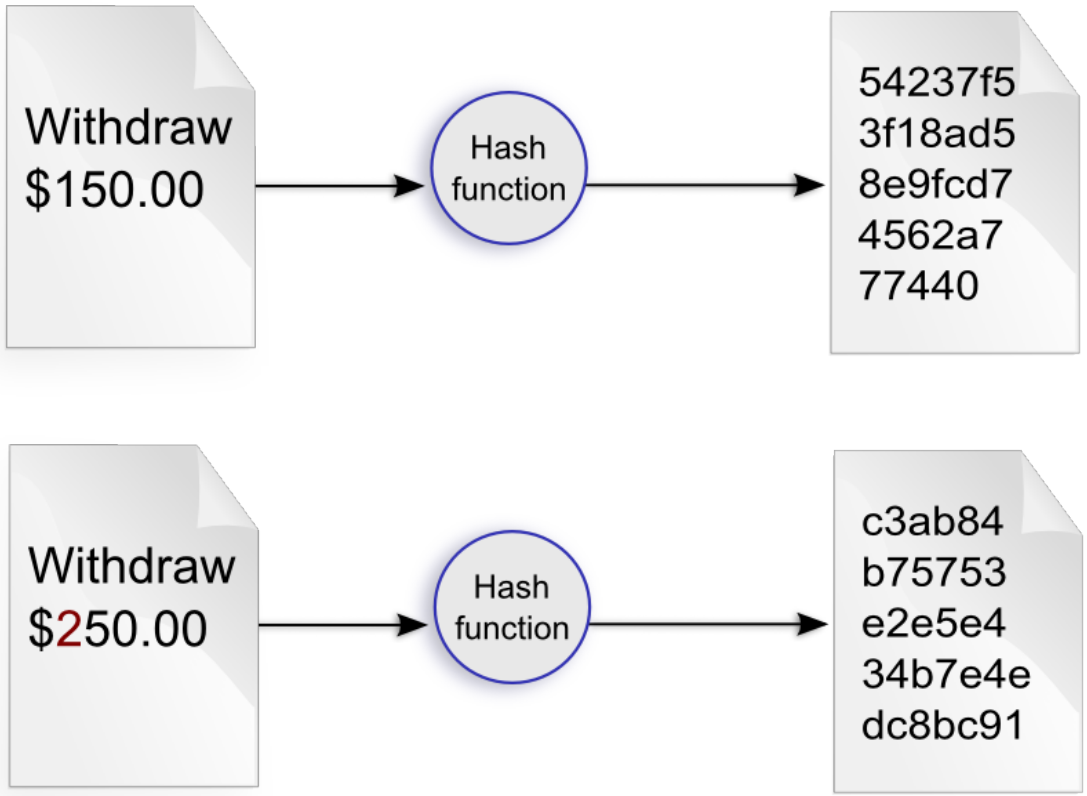
Depends on authentication and is typically implemented with access control lists.



Concept: Data Integrity

The data must be unmodified in storage/transit.

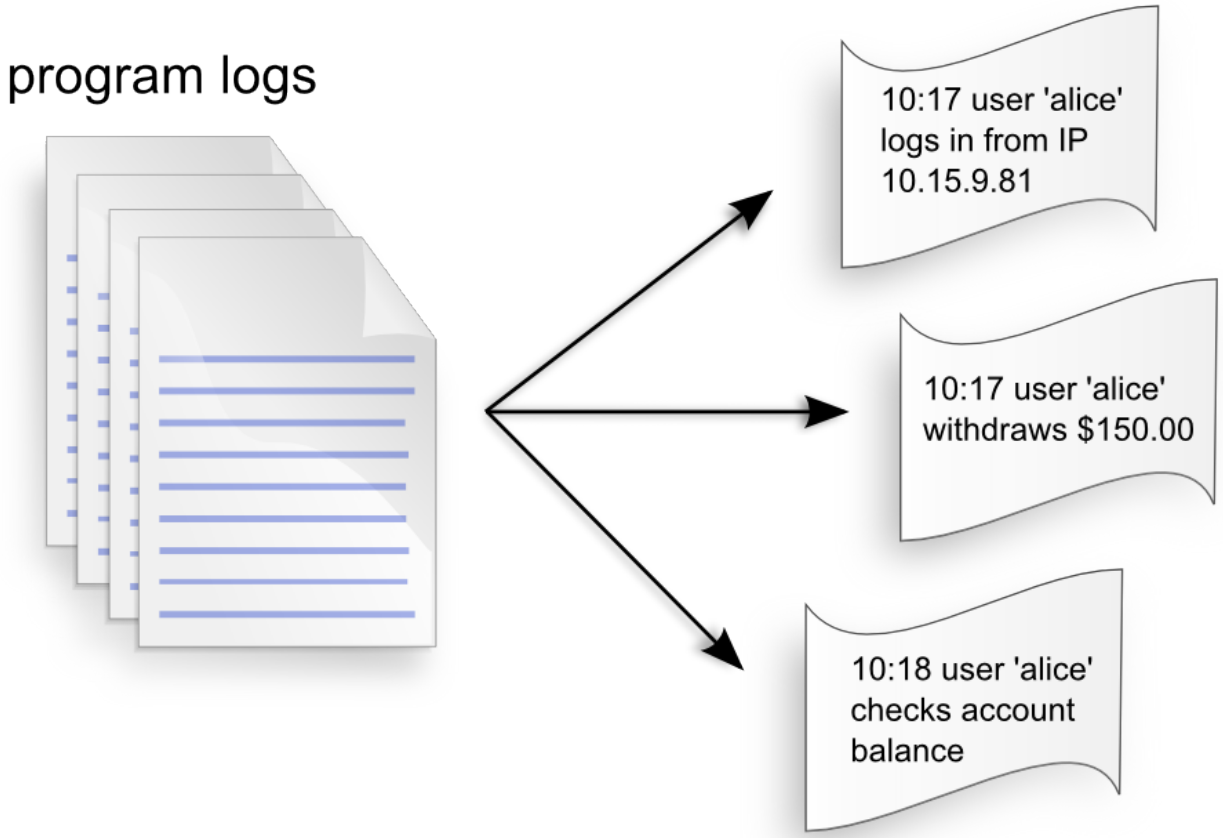
Typically implemented with cryptographic hash functions.



Concept: Accountability

It must be possible to investigate a sequence of actions later.

Typically implemented with some variation of logging.



Concept: Non-repudiation

An actor must not later deny having performed some action.

Typically implemented with some variation of digital signatures

Threat Analysis

- What do you need to protect?
- From what/whom are you trying to protect it?
- How much would a compromise cost?
  - Staff time
  - Lost revenue during downtime
  - Fines or regulatory action
  - Loss of client trust
  - Negative media attention
- What is the likelihood for each of these threats?
- Budget ~ (cost of attack) * (likelihood of attack)
- Probably not worth protecting against Godzilla

Threat Analysis Example

I'll define an example software development scenario. Please consider the following:

- What do you need to protect?
- What are some possible attacks?
- Who might instigate those attacks?
- Which attacks seem most likely?
- Which security concepts seem most relevant here?
- What possible protection measures might you implement?

Threat Analysis Example

A major online auction site has hired you to create a REST API to be used in a mobile app. They want at least the following features:

- View/search auctions
- Bid on auctions
- Premium members can be notified when outbid
- Specify the shipping address upon winning
- Submit payment using a credit card upon winning

Common Attacks: User Input

- Cross-site scripting (XSS / XSRF)
- OS command injection
- SQL injection
- Path traversal
- Buffer overflow
- Bypassing client-side validation
- Defenses

Common Attacks: Authentication

- Social engineering
- Dictionary attack
- Brute force
- Rainbow tables
- Defenses

Common Attacks: Eavesdropping

- Network sniffing (Hubs/switches/gateway impersonation)
- Defenses

Common Attacks: Miscellaneous attacks

- Man in the middle attacks
- Temp file name prediction
- Defenses

Some Security Tips

- Use layers of security: avoid a single point of failure
- Don't trust user input: carefully extract what you need
- Principle of least privilege
- Bare minimum install / disable unneeded services
- Without physical security, there is **no** security

More Security Tips

- Provide info sparingly (e.g. error messages)
- Have a disaster recovery plan and test it periodically
- Don’t rely only on "security through obscurity"
- Never make up your own crypto algorithms

Conclusion

- Questions?