CSE 127 Computer Security

Stefan Savage, Fall 2026

Lecture 2: Threat Modelling and Risk

Logistics

PA1 is assigned (check the assignments section of the Web page)

- Will be due Oct 6th at 11:59pm Pacific Time
- Please start ASAP and let us know if you have any trouble setting it up (note slightly different instructions for folks with ARM-based Macs [M1/M2/etc])

  i.e., Please don't tell us the night before its due that you have trouble getting the environment set up.

  If you're having install problems, please check and post to Piazza so we can look for any commonality
- The discussion section is your friend here!

Important tip: if this doesn't seem *trivial* to you, you should go to discussion section tomorrow!

Finally, late days: you get three late days **total**

- Each provides a contiguous 24 hour period of forgiveness from a deadline; you don't need to ask to use them
- If you are working with a partner, both you and your partner will be charged for a late day; you **BOTH** need to have late days available for **either** of you to use them

Today: Lecture Objectives

Develop a mental framework for:

- Thinking about security risks
- Evaluating potential mitigation options
- Analyzing tradeoffs

- Some bonus material about **locks** and the mental mistakes we make

Threat Model

Threat Model defines ***Security Goals***: What are we trying to protect from whom?
- ***Assets*** and ***Attackers***

One definition of a secure system
- "System that remains dependable in the face of malice" -Ross Anderson

What does "remains dependable" mean?
- ***Confidentiality***, ***Integrity***, and/or ***Availability*** ***(C.I.A.)*** of particular system, component, or data are preserved
- These are the ***Assets*** we need to protect (both at rest [stored] and in flight [being transmitted])
  "An attacker must not be able to compromise the [Confidentiality | Integrity | Availability] of…"
- Assets have value: Must understand value in order to decide how much to spend on protection

What does "malice" mean?
- Malice from whom? Curious roommate? Criminal? Law enforcement? Foreign military?
- These are potential ***Attackers***
  Defined by ***Capability*** and ***Intent***

Confidentiality

Prevention of unauthorized access to information

- Unauthorized parties can't view protected information
- aka Secrecy

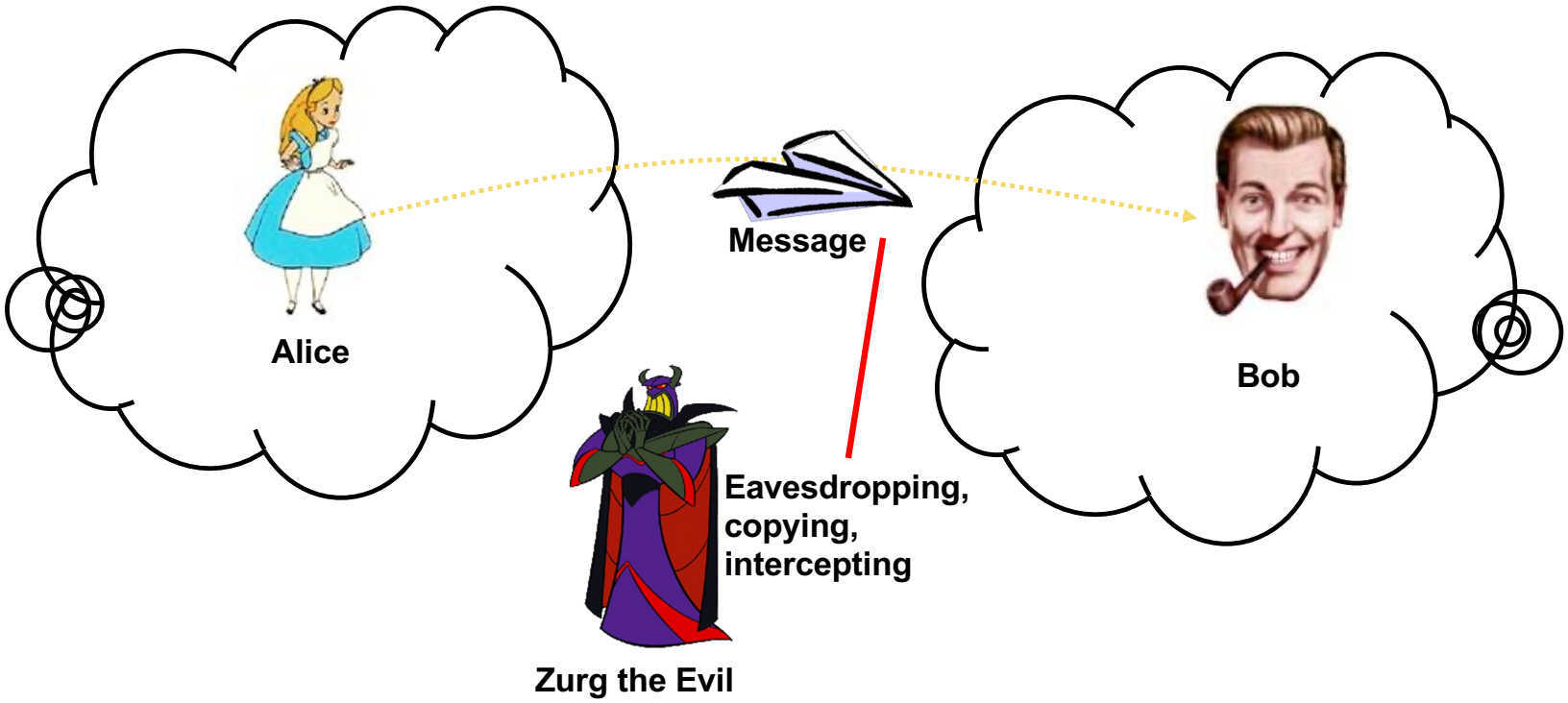
Examples

- Someone reading your private e-mail

What are some other scenarios where you want confidentiality?

Confidentiality

Protection of information; secrecy



Integrity (& Authenticity)

Prevention of unauthorized modification of information, process, or function

- Unauthorized parties can't modify protected information or process

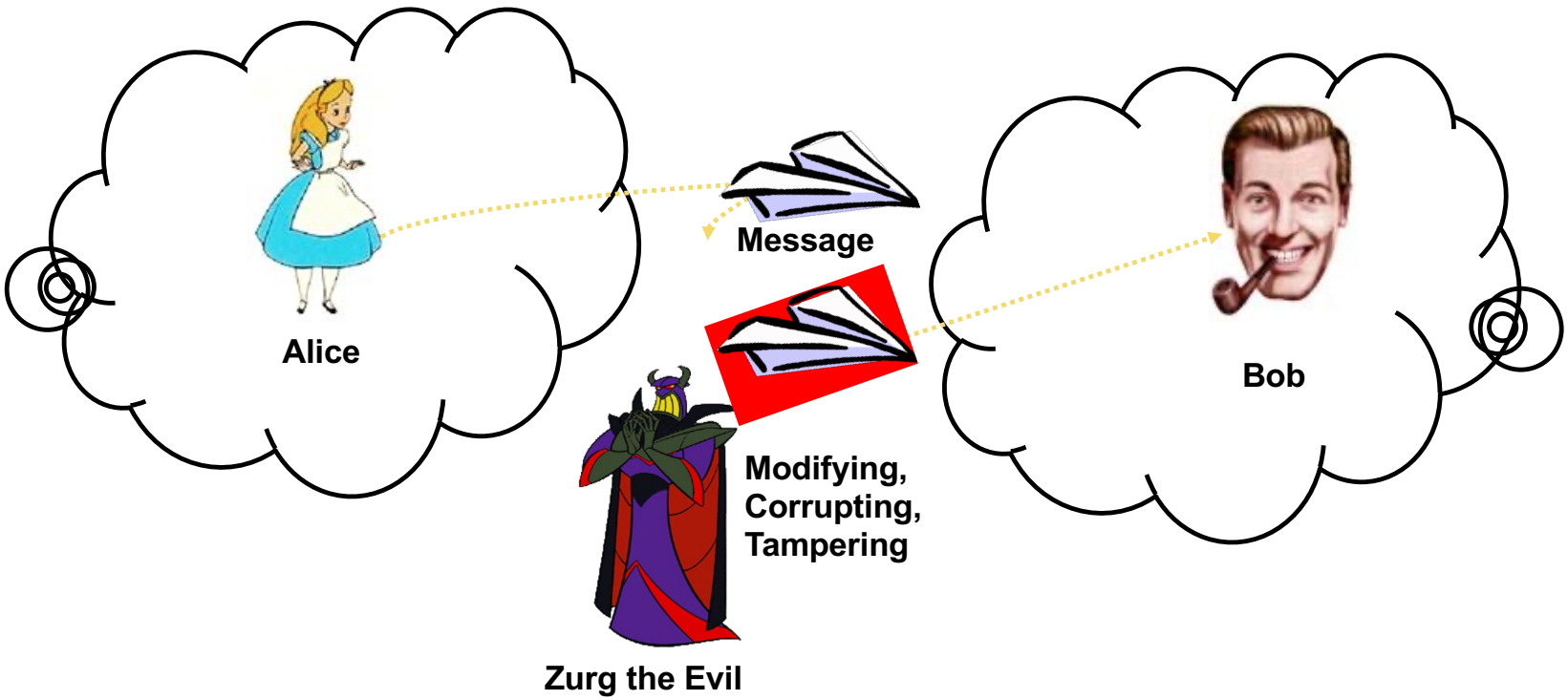
Examples

- Increasing your bank account balance without depositing money
- Getting snacks from vending machine without paying for them

What are some other examples?

Integrity

Prevention of unauthorized changes



Integrity (& Authenticity)

Prevention of impersonation of another **principal**

- aka origin integrity

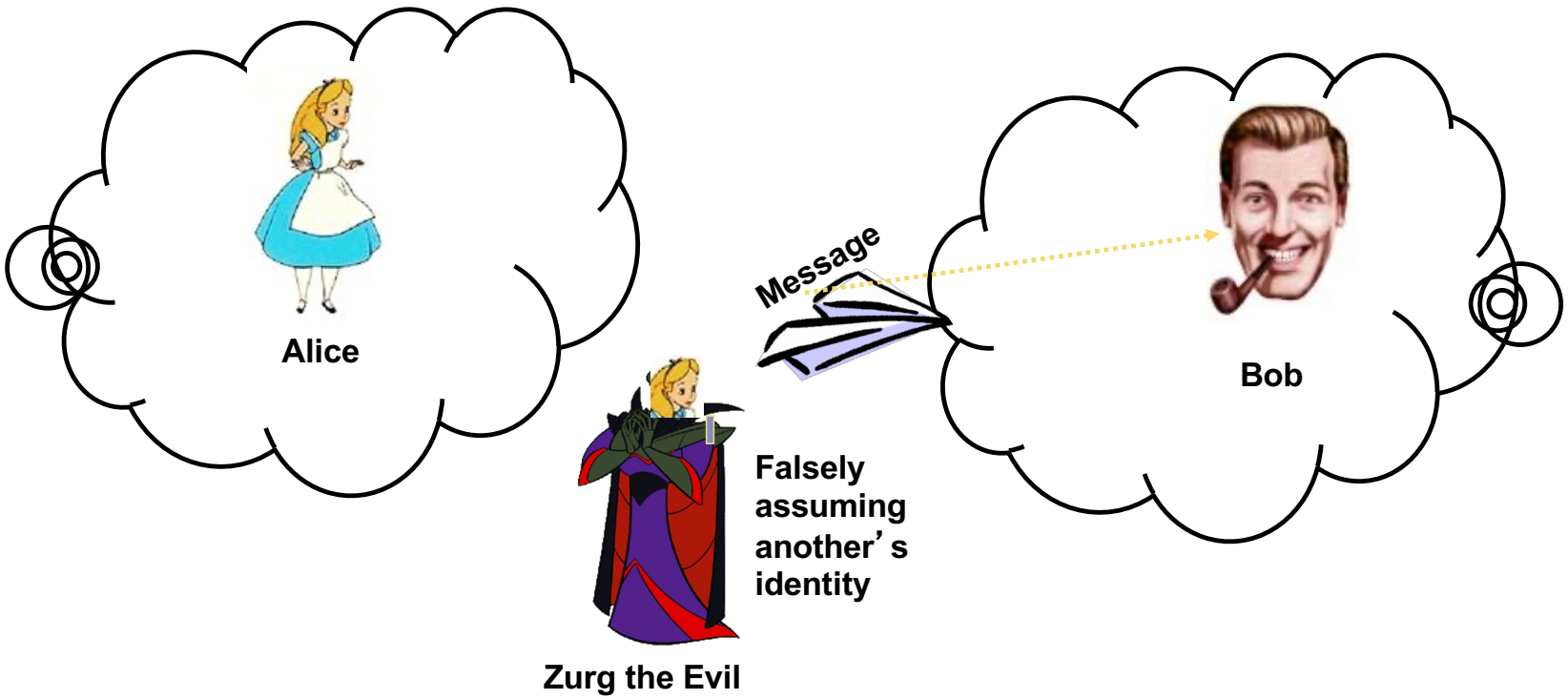
Example

- Getting money from someone else's bank account using their credentials

What are some other examples?

Authenticity

Identification and assurance of origin



Availability

Prevention of unauthorized *denial of service* to others

- Unauthorized parties can't prevent authorized users from accessing system

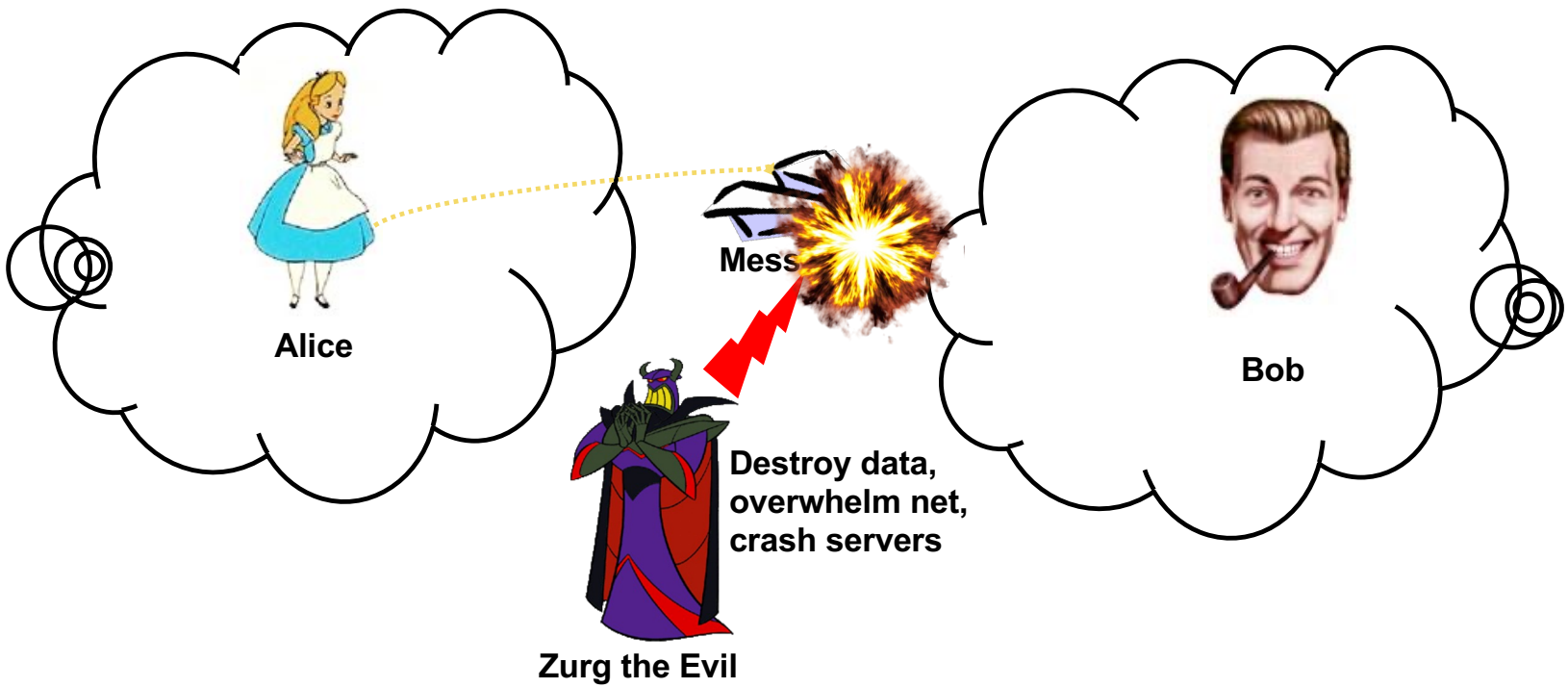
Examples

- Putting superglue in the ATM card slot
- Network denial of service attacks (e.g., packet floods)

What are other examples of denial of service?

Availability

Ability to use information or resources desired



C.I.A. + Privacy

Privacy: A person's right or expectation to control the disclosure of his/her personal information, including activity metadata

What is the difference between privacy and secrecy?

- Secret from whom?
  - Secrecy is about explicitly hiding information from third-parties
  - Privacy is about not being observed/monitored (including via public data)
- Activity metadata
  - e.g., what can you figure out about a person just from their location history?

C.I.A. + Privacy

Which security property is violated if someone …

- unplugs your alarm clock while you're sleeping?
- changes the time on your alarm clock?
- Watches you through your window via a telescope?

Vulnerabilities

Weaknesses that could be exploited to cause damage to assets (typically where CIA + Privacy are not enforced as intended)

- Default password left intact ("Admin")
- Bad password chosen ("password")
- Implementation flaws in software
- Debug interface left open to network
- Cryptography based on *weak* keys

In particular, look to ***assumptions*** in system

Known vulnerabilities (e.g., documented in public forums, NVD) vs unknown vulnerabilities (0-days)

Threat Model

Threat Model defines ***Security Goals***: What are we trying to protect from whom?

- ***Assets*** and ***Attackers***

  Defined by ***Capability*** and ***Intent***

Attackers/Adversaries

Types

- Individual
  - Outsider
  - Insider
  - Trusted/Privileged Insider
- Group
  - Ad hoc
  - Established
- Organization
  - Competitor
  - Supplier
  - Partner
  - Customer
- State or State-sponsored actor

Capabilities

- Time
- Money
- Training
- Access

Motivation/Intent

- Curiosity
- Fame
- Money
- National interest

Trusted Computing Base: what you trust

Trusted Computing Base

- Set of systems/components/people/entities that your security depends on
- Security dependency

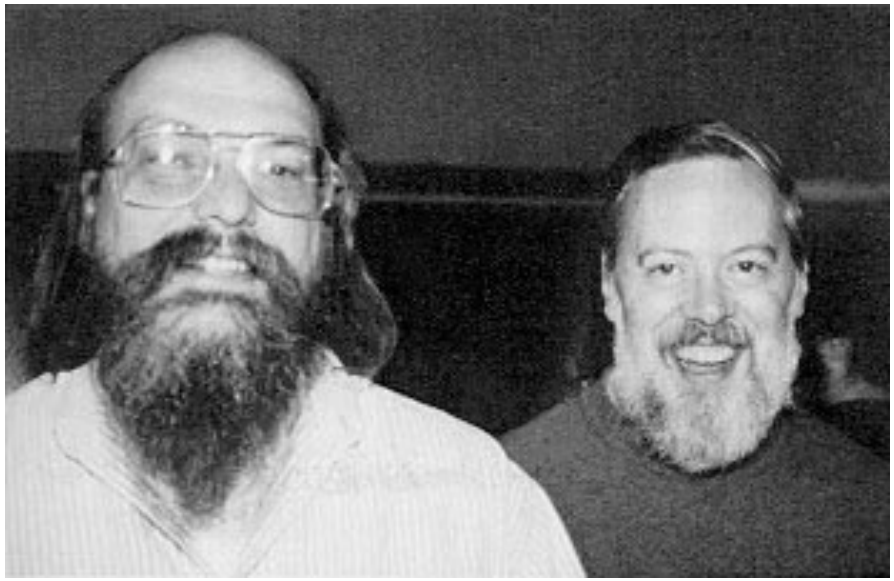
You *need* to trust something

- Trusted != Trustworthy
- Reflections on Trusting Trust

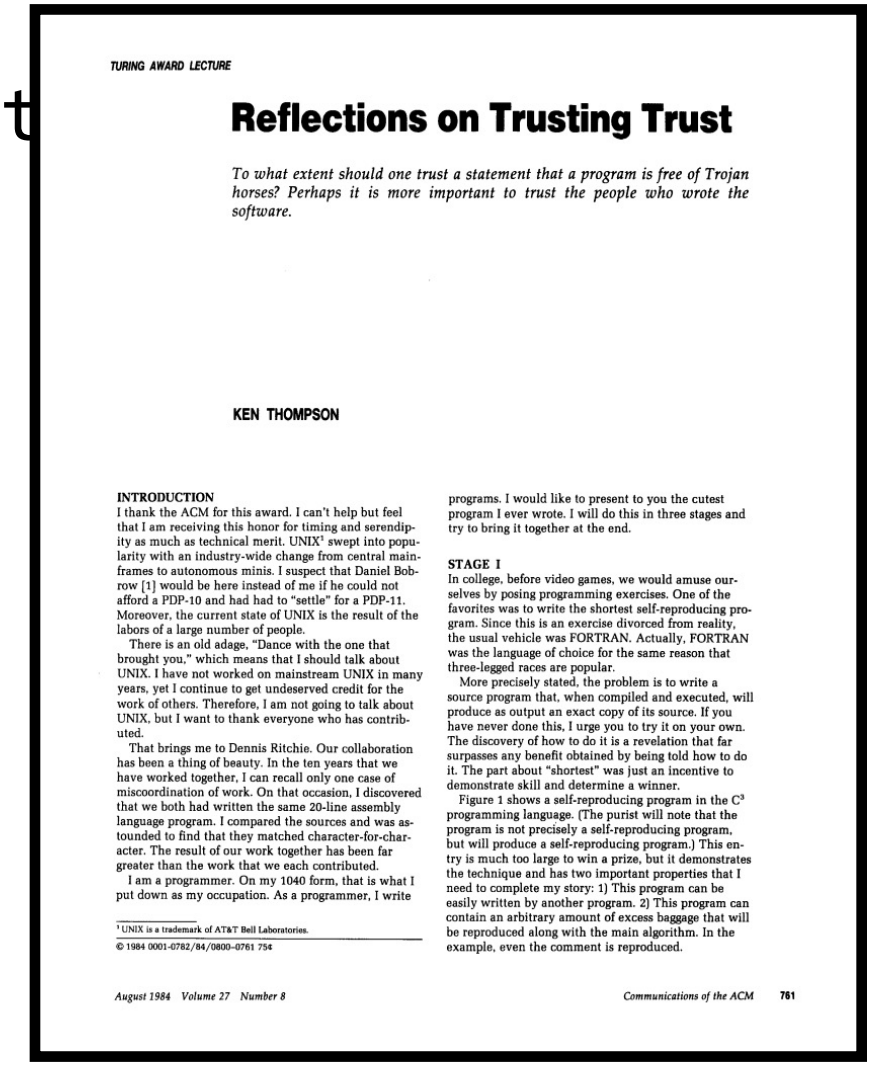
  How much do you need to trust?

Reflections on Trusting Trust

1984 Turing Award Lecture



Ken Thompson + Dennis Ritchie



TURING AWARD LECTURE

Reflections on Trusting Trust

To what extent should one trust a statement that a program is free of Trojan horses? Perhaps it is more important to trust the people who wrote the software.

KEN THOMPSON

INTRODUCTION

I thank the ACM for this award. I can't help but feel that I am receiving this honor for timing and serendipity as much as technical merit. UNIX[1] swept into popularity with an industry-wide change from central mainframes to autonomous minis. I suspect that Daniel Bobrow [1] would be here instead of me if he could not afford a PDP-10 and had had to "settle" for a PDP-11. Moreover, the current state of UNIX is the result of the labors of a large number of people.

There is an old adage, "Dance with the one that brought you," which means that I should talk about UNIX. I have not worked on mainstream UNIX in many years, yet I continue to get undeserved credit for the work of others. Therefore, I am not going to talk about UNIX, but I want to thank everyone who has contributed.

That brings me to Dennis Ritchie. Our collaboration has been a thing of beauty. In the ten years that we have worked together, I can recall only one case of miscoordination of work. On that occasion, I discovered that we both had written the same 20-line assembly language program. I compared the sources and was astounded to find that they matched character-for-character. The result of our work together has been far greater than the work that we each contributed.

I am a programmer. On my 1040 form, that is what I put down as my occupation. As a programmer, I write programs. I would like to present to you the cutest program I ever wrote. I will do this in three stages and try to bring it together at the end.

STAGE I

In college, before video games, we would amuse ourselves by posing programming exercises. One of the favorites was to write the shortest self-reproducing program. Since this is an exercise divorced from reality, the usual vehicle was FORTRAN. Actually, FORTRAN was the language of choice for the same reason that three-legged races are popular.

More precisely stated, the problem is to write a source program that, when compiled and executed, will produce as output an exact copy of its source. If you have never done this, I urge you to try it on your own. The discovery of how to do it is a revelation that far surpasses any benefit obtained by being told how to do it. The part about "shortest" was just an incentive to demonstrate skill and determine a winner.

Figure 1 shows a self-reproducing program in the C[3] programming language. (The purist will note that the program is not precisely a self-reproducing program, but will produce a self-reproducing program.) This entry is much too large to win a prize, but it demonstrates the technique and has two important properties that I need to complete my story: 1) This program can be easily written by another program. 2) This program can contain an arbitrary amount of excess baggage that will be reproduced along with the main algorithm. In the example, even the comment is reproduced.

[1] UNIX is a trademark of AT&T Bell Laboratories.

© 1984 0001-0782/84/0800-0761 75¢

Basic thought experiment…

Do you trust the software you're running?

- What if adversary put "backdoor" in the binary (i.e., so it does something bad sometimes)

Check the source code?

- But how do you know the version you're running came from that source code?

Recompile the source code after checking it?

- But what if the compiler is compromised and it inserts a backdoor when it compiles (i.e., that isn't in the program's ource code)?

Check the compiler source code?

- How do you know the compiler binary you're running is from that code?

Compile the compiler from source?

- But how do you know that the compiler you're using doesn't insert a backdoor into the binary you created?

Etc… at some point you have to trust something and build off that trust

Aside: XZ Utils Backdoor – CVE-2024-3094

In February 2024, a Microsoft employee found a backdoor in XZ, a popular compressor for Unix-like operating systems

- The backdoor was found in compressed test files that enabled full remote-code execution on the machine using the compromised xz via sshd

  Github core source code was fine, but release tarballs were compromised

- The backdoor was introduced *in the supply chain* – a developer (Jia Tan) who contributed to XZ for *years* introduced the patch in obfuscated test files

  Primary guess right now is state actor, but we're not sure

Trust/Security Boundary & Attack Surface

Security Boundary

- Perimeter around components of same trust level
- Any data or signals coming in from outside is untrusted and potentially malicious

Attack Surface

- Set of interaction points across a security boundary
- Parts of your system handling input from or otherwise interacting with less trusted and potentially malicious entities

  Port, Inter-procedure Call, API, parser, etc.
- Some *highly sensitive* systems are even "air-gapped" to minimize the attack surface

Threat Model

Your very first question in any security discussion should be

What's the <u>threat model</u>?

Do not argue about attacks or defenses without understanding the threat model

The threat model defines the problem to be solved.

- If there is no consensus on the problem, there will be no consensus on solutions
- Also… consider if threat model is ***reasonable*** for actual situation

Threat Model Example: Personal Items

Asset

- TV, jewelry

Security Boundary

- Access to house (door, locks, windows, etc)
- Who has access? spouse, roommate

Attacker

- Roommate, thief?
- Do you have to worry about state-sponsored attackers?

Threat Model

A cautionary note

- Your threat model is <u>your</u> problem scope.
- Attackers ***don't care*** about your threat model.
- Just because an asset or an attacker are outside your threat model, does not mean that they do not exist.

  It just means that you have explicitly decided that you will not address them in your solution.

Risk Assessment

Now that we know what we want to protect and from whom, we can reason about what risks attackers can pose to our assets

Security is rarely binary

Everything has some measure of risk

- A rough, informal scale: from Very Low to Very High
- Calculated as some combination of:
  - Likelihood – the probability that a security threat will materialise
  - Impact – the adverse affects that will occur if a threat does materialise

We evaluate security risk relative to value

- Is this risk worth taking?
- Similar to health and financial risks

Risk Assessment (idealized)

Risk assessment/risk analysis process for assessing risk in a system:

1. Start by understanding system requirements
2. Identify assets and attackers
3. Establish security requirements
4. Evaluate system design
5. Identify threats and classify risks
6. Address identified risks

1. Understand system requirements

You can't make a system secure if you don't know what it is supposed to do

What is the system supposed to do?

- What is the security boundary?
- Look for vague or open-ended requirements. These will likely be interpreted in different ways by different architects and developers.

What is the lifecycle? How will it be deployed?

2. Identify assets and attackers

Who are the stakeholders?

- Who has a vested interest in the system running correctly/securely.

What needs to be protected?

- What is the value? To whom?

From whom does it need to be protected?

3. Establish security requirements

Think about the CIA+P classification

"An attacker must not be able to compromise the [Confidentiality | Integrity | Availability] of…"

4. Review system design

You can't make a system secure if you don't understand how it works

What are the different components in the system?
- How will the components communicate?
- How will they store and/or pass data?

Draw a block diagram of the system
- Indicate security boundaries and information flow
- Describe protocol interactions

Again, look for ambiguity and hidden assumptions

This is only the starting point
- The design will evolve as risks are identified and mitigated

Later, make sure that implementation matches the design

5. Identify threats and classify risks

Using the security goals and current design, identify ***Threats*** to security

- Possible ways in which attackers (as defined by your threat model) could exploit vulnerabilities in system design or implementation to compromise the assets (as defined by your threat model)

For each threat, determine likelihood and impact and convert to a risk "score"

- (this is inherently ad hoc, but it's the best we know how to do)

Identifying Threats

Adopt an adversarial mindset

- An attacker does not have to obey any rules established by the defender.
  - Requirements are for builders, not breakers.
  - It doesn't matter what is *supposed* to happen.
    - Review the code to see what actually happens.
- Assume that the untrusted parts of the system work in the worst possible way – i.e. assume adversarial control.
  - That includes any external components, like storage, etc.
- Look for assumptions about data coming across the security boundary and try to violate them.
  - E.g., if the documentation says that commands have to have a certain format, then evaluate what can happen when the format is incorrect.
  - E.g., if the documentation says that some commands are supposed to be executed in a certain order, then evaluate what can happen when the order is wrong.
  - Etc

Identifying Threats

Strategies

- Start with the assumption that there is yet another way to attack the system
  - You just need to find it
- Attack the weakest link
  - Focus on parts that are ambiguous or seem overly complex.
  - Look for ambiguity in requirements and design and see if different parts of the system were implemented using different interpretations.
- Be curious, ask stupid questions.
- Consider the security goals one-by-one and look for different types of attacks against that goal.
  - Attack Trees
- Consider the types of attack one-by-one and look for different ways that these attacks can affect a goal.
  - STRIDE

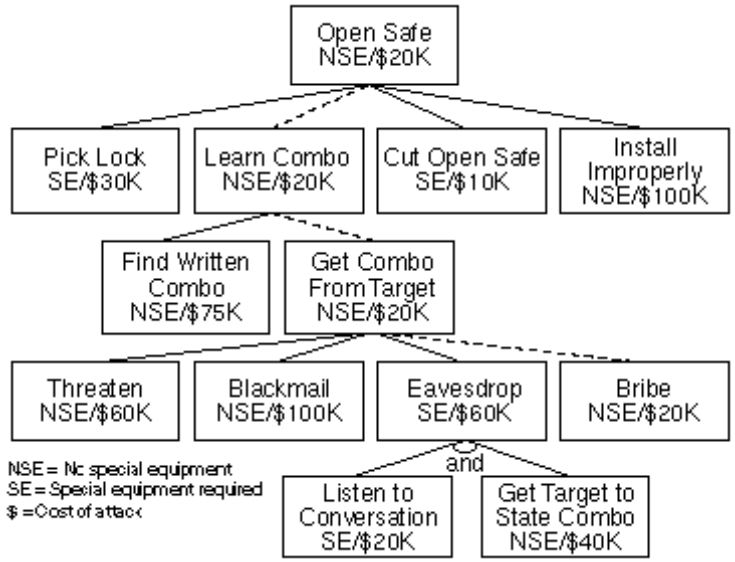
Identifying Threats

Attack Trees

- Start with an attacker's goal as the root of a tree

  "Read secret …" or "Modify value…"

- Different ways of achieving a goal (sub-goals) become child nodes
- Can be annotated with likelihood, cost, etc.



https://www.schneier.com/academic/archives/1999/12/attack_trees.html

Identifying Threats

STRIDE: Microsoft-developed model

- Start with possible attack types and evaluate how they can violate security goals
- Spoofing identity.

  An example of identity spoofing is illegally accessing and then using another user's authentication information, such as username and password.
- Tampering with data.

  Data tampering involves the malicious modification of data. Examples include unauthorized changes made to persistent data, such as that held in a database, and the alteration of data as it flows between two computers over an open network, such as the Internet.
- Repudiation.

  Repudiation threats are associated with users who deny performing an action without other parties having any way to prove otherwise—for example, a user performs an illegal operation in a system that lacks the ability to trace the prohibited operations. Nonrepudiation refers to the ability of a system to counter repudiation threats. For example, a user who purchases an item might have to sign for the item upon receipt. The vendor can then use the signed receipt as evidence that the user did receive the package.
- Information disclosure.

  Information disclosure threats involve the exposure of information to individuals who are not supposed to have access to it—for example, the ability of users to read a file that they were not granted access to, or the ability of an intruder to read data in transit between two computers.
- Denial of service.

  Denial of service (DoS) attacks deny service to valid users—for example, by making a Web server temporarily unavailable or unusable. You must protect against certain types of DoS threats simply to improve system availability and reliability.
- Elevation of privilege.

  In this type of threat, an unprivileged user gains privileged access and thereby has sufficient access to compromise or destroy the entire system. Elevation of privilege threats include those situations in which an attacker has effectively penetrated all system defenses and become part of the trusted system itself, a dangerous situation indeed.

https://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx

Assessing Risk

We can "score" identified risk based on combination of Likelihood and Impact:

Risk assessment is subjective

- Use your knowledge and common sense to make sure that the risk ranking is reasonable.

		Impact				
		Very Low	Low	Moderate	High	Very High
Likelihood	Very Low	Very low	Very Low	Very Low	Low	Low
	Low	Very Low	Very Low	Low	Low	Moderate
	Moderate	Very Low	Low	Moderate	Moderate	High
	High	Low	Low	Moderate	High	Very High
	Very High	Low	Moderate	High	Very High	Very High

https://csrc.nist.gov/publications/detail/sp/800-30/rev-1/final

6. Address identified risks

Once risk is identified and classified we must decide what to do about it

Avoid: Remove component that creates the risk

- Example: Disable the feature

Mitigate: Add measures that decrease likelihood or impact

- Example: Better defense mechanisms

Transfer: Make it someone else's problem

- Example: Insurance

Accept: Do nothing

- There's always residual risk we must accept
- https://www.youtube.com/watch?v=9IG3zqvUqJY

  Pwnie-nominated performance https://pwnies.com/

Risk Acceptance

It is impossible to offer functionality without risk.

Residual risk is left over when all the controls and protections have been installed.

Decision to accept risk should be made by someone who has sufficient authority to do so.

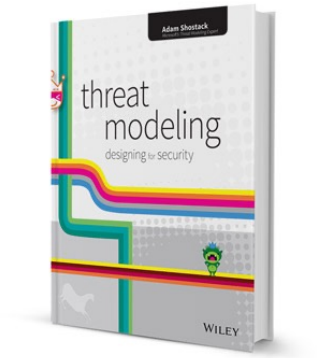
Remember: accepting the risk does not make it go away.

- Regardless of why you choose to accept it
- If there's no money or time to deal with a threat, does not make the risk disappear

Additional Resources

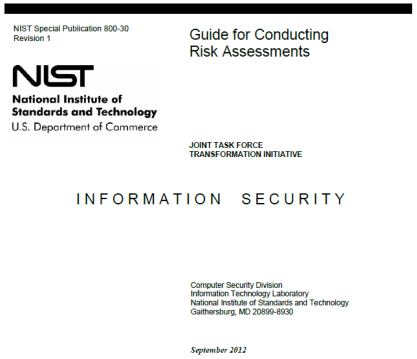
Threat Modeling: Designing for Security

- by Adam Shostack
- https://threatmodelingbook.com/



NIST Guide for Conducting Risk Assessments

- SP 800-30 Rev. 1
- https://csrc.nist.gov/publications/detail/sp/800-30/rev-1/final


NIST Special Publication 800-30
Revision 1

Guide for Conducting
Risk Assessments

NIST
National Institute of
Standards and Technology
U.S. Department of Commerce

JOINT TASK FORCE
TRANSFORMATION INITIATIVE

INFORMATION SECURITY

Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8930

September 2012

Review

Threat Model defines ***Security Goals***: What are we trying to protect from whom?

- ***Assets***: ***Confidentiality***, ***Integrity***, and/or ***Availability*** ***(C.I.A.)*** of particular system, component, or data are preserved
- ***Attackers***: ***Intent*** and ***Capability***

Security is almost never binary, everything has some measure of ***Risk***

- Calculated as some combination of ***Likelihood*** and ***Impact***
- Is this risk worth taking?

Review

Risk assessment/risk analysis process for assessing risk in a system.

Adversarial Mindset

- An attacker does not have to obey any rules established by the defender.
- Assume that the untrusted parts of the system work in the worst possible way.
- Violate assumptions.
- An attack is possible, you just need to find it.

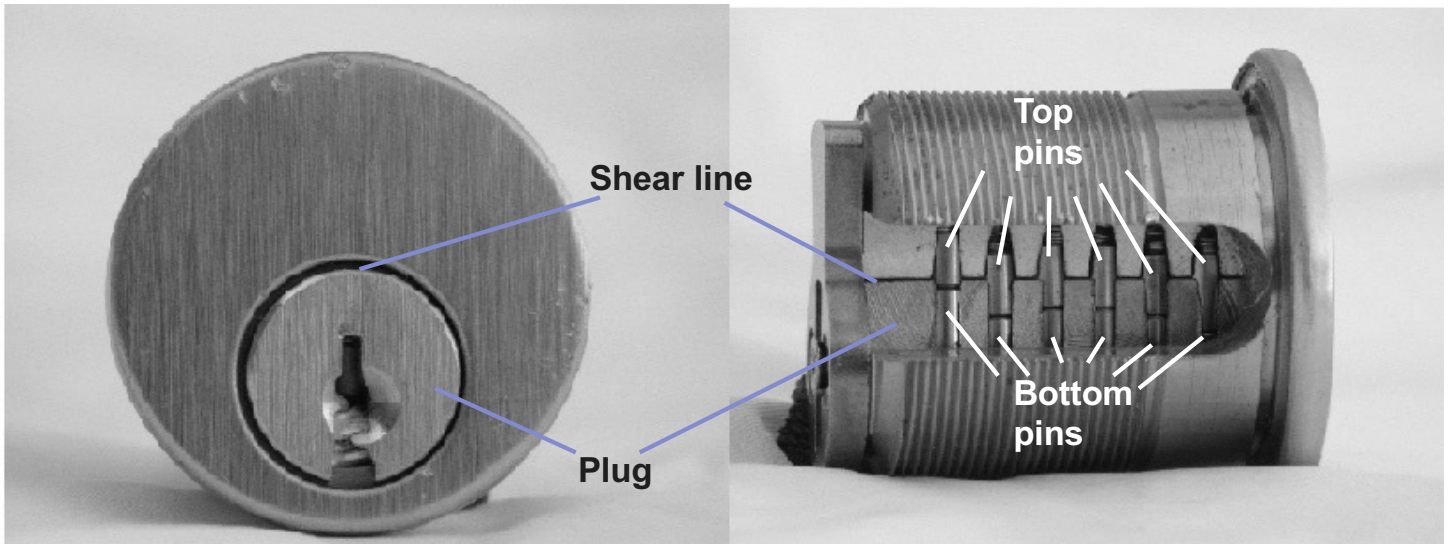
Once risk is identified and classified we must decide what to do about it:

- Avoid, Mitigate, Transfer, Accept

Bonus: using the adversarial mindset

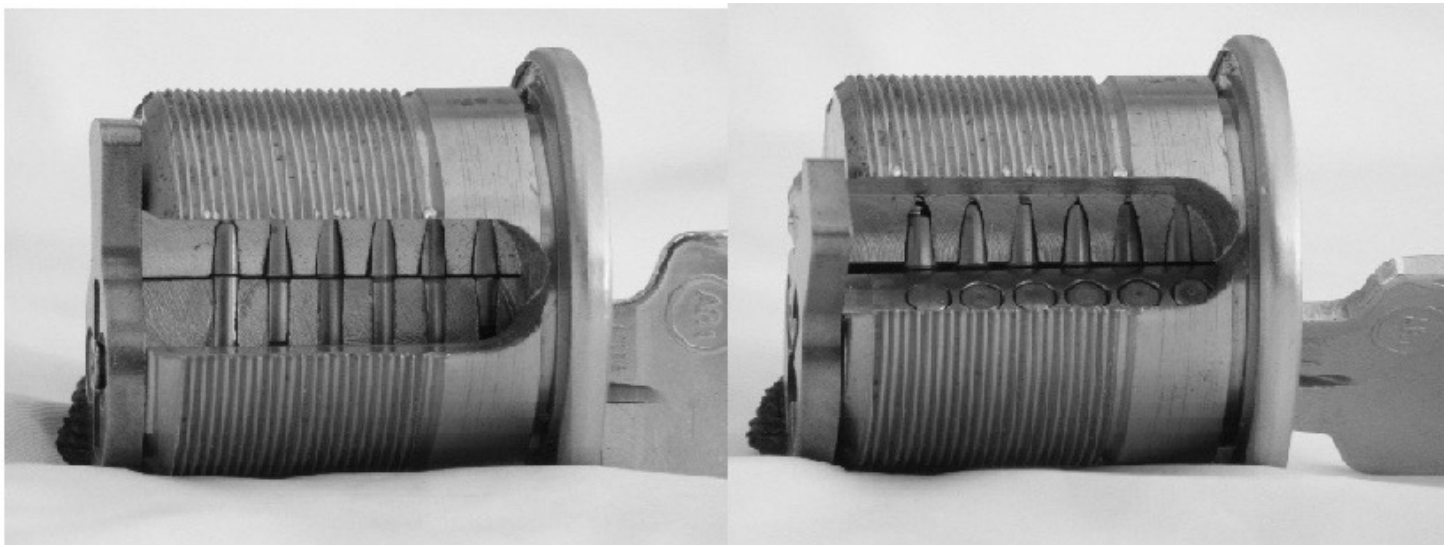
We depend on physical locks to secure our property

How physical locks work



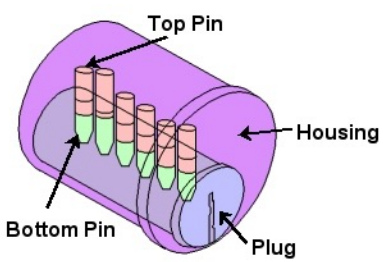
courtesy Matt Blaze

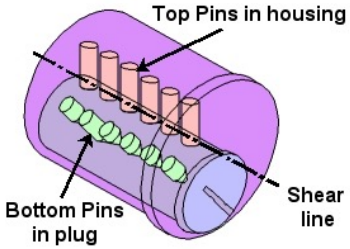
How physical locks work

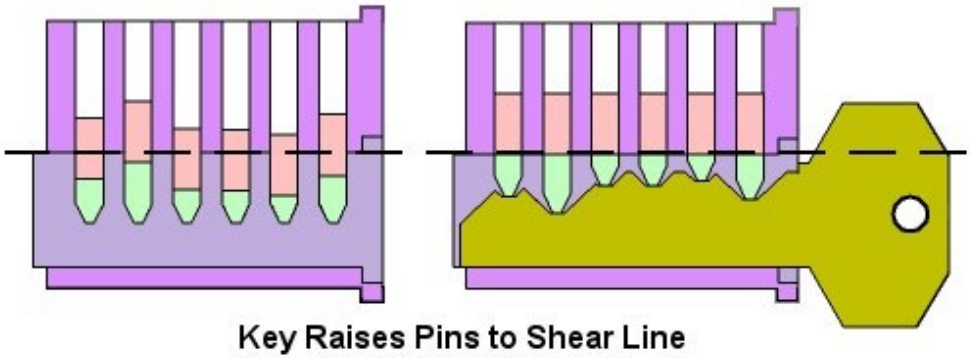


courtesy Matt Blaze

Another visualization







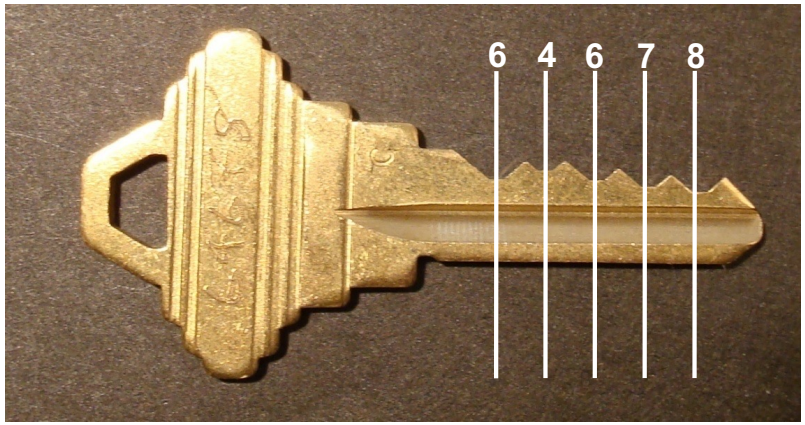
Shared secrets

There is a shared secret between the lock and the key… its shape

In fact, it's a digital code

Bitting codes

- A key can be precisely described with a discrete code
  - Cuts at regular intervals (4-6 cuts)
  - Depth of cuts quantized in standard fashion (typically 6-9 bins)
  - 4-6 digits sufficient to describe most keys



Design assumptions

If you don't know the secret code, you can't open the lock

The secret code is secret

If you can't open the lock, everything is fine

Lock bypass via manipulation

Picking & Raking

Bumping

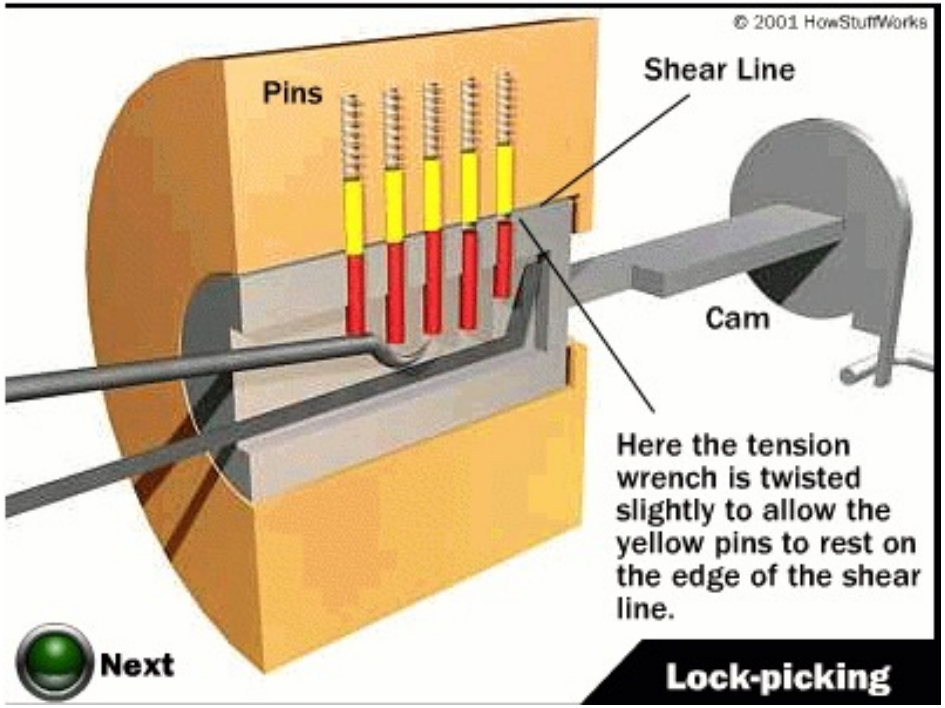
Picking

Two parts

- Tension wrench used to apply *slight* lateral force on plug
- Pick used to lift individual bottom pins to the shear line

Tension causes top pins to bind above shear line

Picking

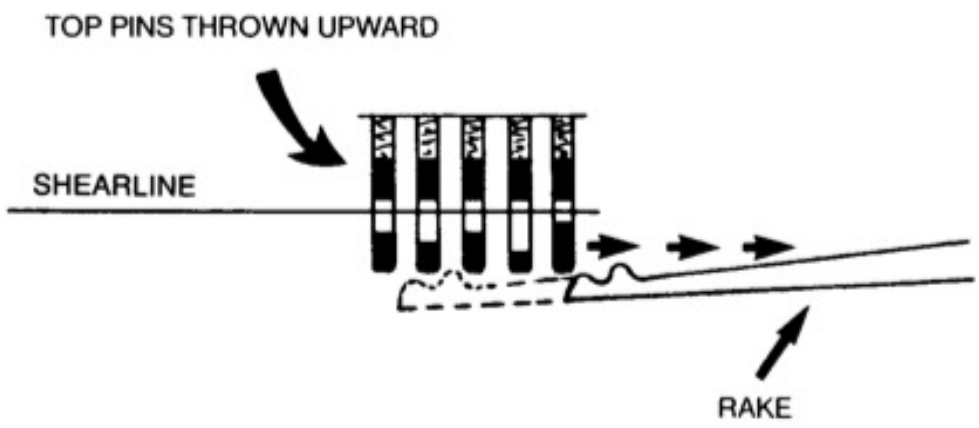

© 2001 HowStuffWorks
Pins
Shear Line
Cam
Here the tension wrench is twisted slightly to allow the yellow pins to rest on the edge of the shear line.
Next
Lock-picking

Raking

Similar idea, but less finesse…

Rake pick moved in and out quickly imparts force to bottom pins; driver pins bind

Quick & easy

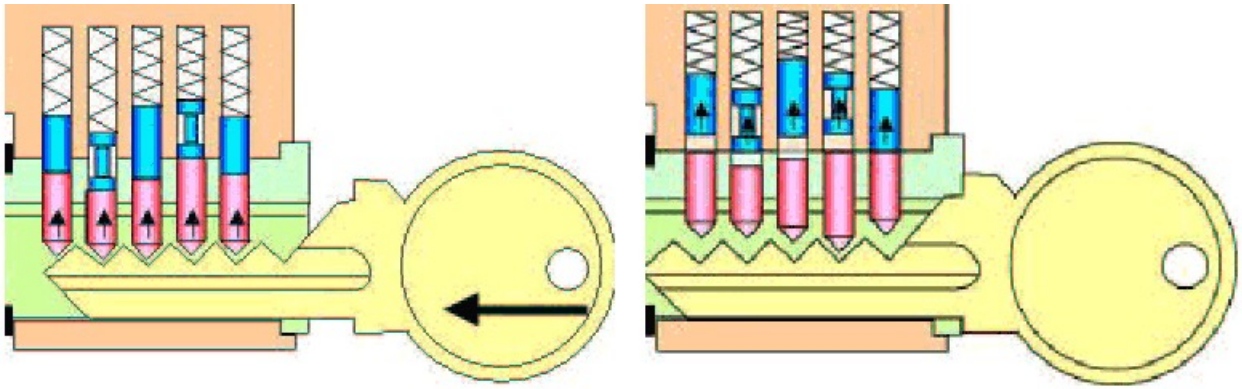


Bumping

Similar idea to raking, but does all pins in parallel; super easy to do

Max-depth key (bump key) used to impart force to bottom pins who transfer energy to driver pins (think billiards)

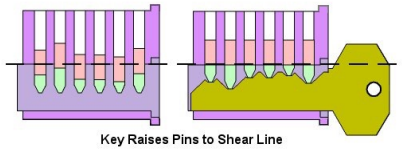
Bumping



Comb picks

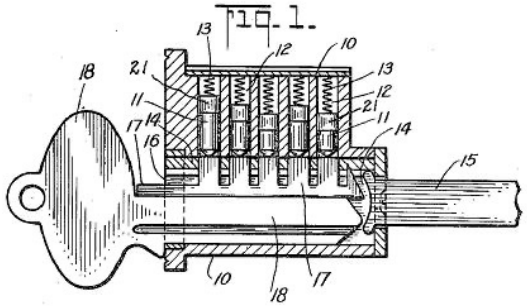
Assumption

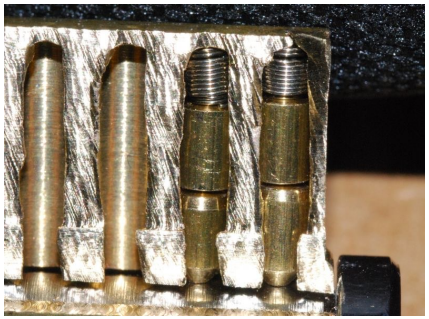
- The only way to open lock is by raising bottom pins to the right place so cylinder can rotate (top pins above shear line, bottom pins below)



Reality

- Some locks have enough room to compress both top and bottom pins above the shear line of the cylinder

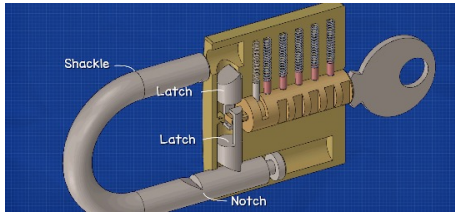




Padlock Shims

Assumption

- Turning the cylinder is the only way to unlock the clasp on the padlock



Reality

- Sometimes you can fit a thin piece of metal (shim) between the shackle and the lock and free it without turning the cylinder at all

Design assumptions

If you don't know the secret code, you can't open the lock

The secret code is secret

If you can't open the lock, everything is fine

Problem

The bitting code is only secret if the key is kept secure

What if I "borrow" your key?

Lock bypass via surreptitious duplication

Field casting

Decoding

The power of decoding

Code key cutting machine

+

Key
Blank

+ 64678 =

Key replica

April 4 2008 95/366 - House keys! on Flickr - Photo Sharing! - Windows Internet Explorer

http://flickr.com/photos/misssage/2389760360/

flickr

Home The Tour Sign Up Explore

You aren't signed in Sign In Help

Search

April 4 2008 95/366 - House keys!

Uploaded on April 5, 2008
by ♥Sage

♥Sage's photostream

785 uploads

This photo also belongs to:

Project 365/366 2008 (Set)

179 items

April 2008 (Set)

Done

Internet | Protected Mode: On

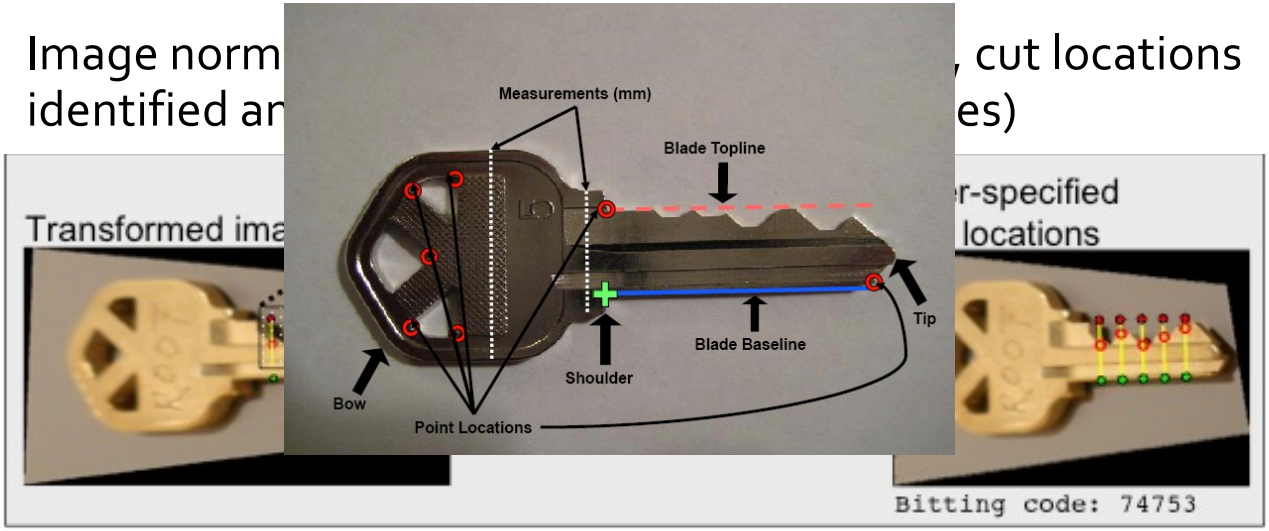
5
3
3
8
6

Optical decoding

- Decode keys semi-automatically from photos
- Traditional computer vision problem (photometry)
  - Normalize for scale and rotation

Sneakey: UCSD

- Reference key measured at control points
- User supplies correspondences between target key and reference image
- Image norm , cut locations identified ar es)



Works really well

Almost perfectly from up close photos
(e.g., cell phone cameras, etc)

But that's no fun… what would James Bond do?

CELESTRON

Where's the Key?
74753
SAFES,
SECURITY
Vol. I
2nd Ed.

Design assumptions

If you don't know the secret code, you can't open the lock

The secret code is secret

If you can't open the lock, everything is fine

Threat model example: home safety

What is the threat?

- Capabilities, resources, goals
- Faster than the bear or faster than the next guy?

What are all the ways the adversary might get access (the "attack surface")?

Aside: For those interested…

- Check out
  - Matt Blaze's work

    Safecracking for the Computer Scientist

    Cryptology and Physical Security: Rights Amplification in Master-Keyed Mechanical Locks

    Notes on Picking Pin Tumbler Locks,
  - MIT Guide to Lockpicking
  - Locksport International (http://locksport.com/)
  - Lockpicking lawyer videos (both youtube and tiktok)
- However…
  - ***NEVER*** pick a lock you do not own
  - ***ALWAYS*** know the local law about using such tools

For next time

We get technical…

- Would be good for you to refresh your memory about how function calls are implemented in normal imperative programming languages

Read *Smashing The Stack For Fun And Profit* by Aleph One

- http://phrack.org/issues/49/14.html#article

Next Lecture…

Control flow vulnerabilities: buffer overflows