COSC 366
Intro to Computer Security

Lecture 03
Basic Security 02

Dr. Suya

Fall 2024

Today's Class

- Security concepts 101
- Think like a defender

Announcements

- Assignment is posted, due Sep 9th
- Class schedule for assignments, projects and (tentative) lecture topics are posted
- Reduced to 3 projects, 4 assignments: as a result, grade distribution is adjusted as follows

Homework (4 in total): 40%
Projects (3 in total): 30%
Midterm Exam: 10%
Final Exam: 20%
Bonus points: answer questions with PointSolution

Think like A Defender

- If you only remember these things, you'll be okay (mostly):
  - Assume the adversary knows *everything* - don't try to hide!
  - You *can* plan for attacks - built in security controls
- Security is not free, but it is *essential* - you must weigh the cost to an organization with the benefit of added engineering efforts

Can It Be Hacked?

- Your grandma's computer?
- A computer with no connectivity in an underground vault that no one has keys to?
- Your home router?
- Your voice-controlled assistants (VCA)?

Dolphin Attack (CCS'17)

- A completely inaudible attack that modulates voice commands on ultrasonic carriers >20kHz
- Can be demodulated and interpreted by VCA
- Attacks
  - Activating Siri to make a FaceTime call on iPhone
  - Activating Google Now to switch the phone to airplane mode
  - Manipulating the navigation system on an Audi

Dolphin Attack (CCS'17)

CommanderSong (USENIX'18)

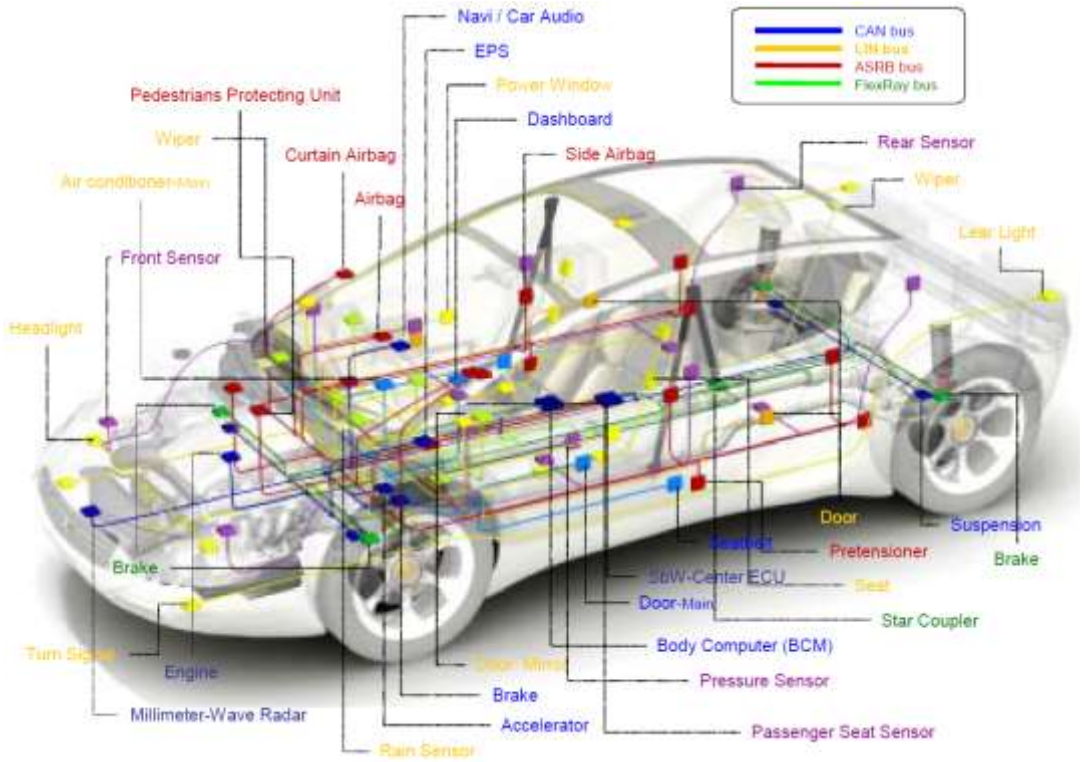
- Does not need an ultrasonic device or transducer
- Embed voice commands into songs that can affect a large number of users
- Leverages vulnerabilities in machine learning models

Can It Be Hacked?

- A Ford Model T?

Can It Be Hacked?

- A 2014 Jeep Cherokee (Uconnect)?
  https://www.youtube.com/watch?v=MK0SrxBC1xs



https://www.youtube.com/watch?v=MK0SrxBC1xs

Can It Be Hacked?

- Police bodycams?

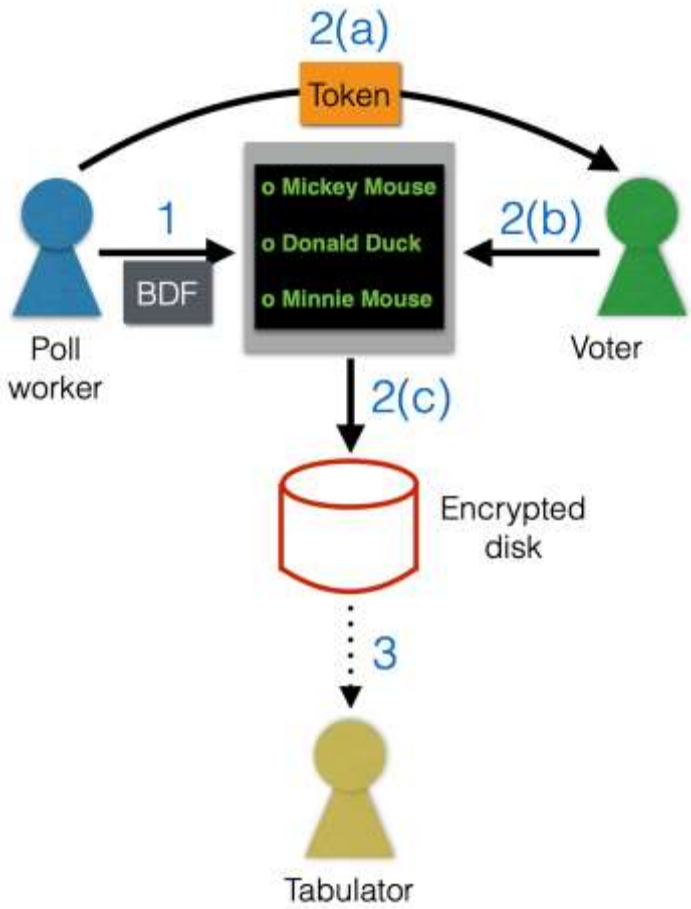
Hacking Police Body Cameras

Security Mindset

- To anticipate attackers, we must be able to think like them
- The ability to view a large, complex system and be able to reason about:
  - What are the potential security threats?
  - What are the hidden assumptions?
  - Are the explicit assumptions true?
  - How can we mitigate the risks of the system?

E-voting Analysis

- Summarize the system as clearly and concisely as possible

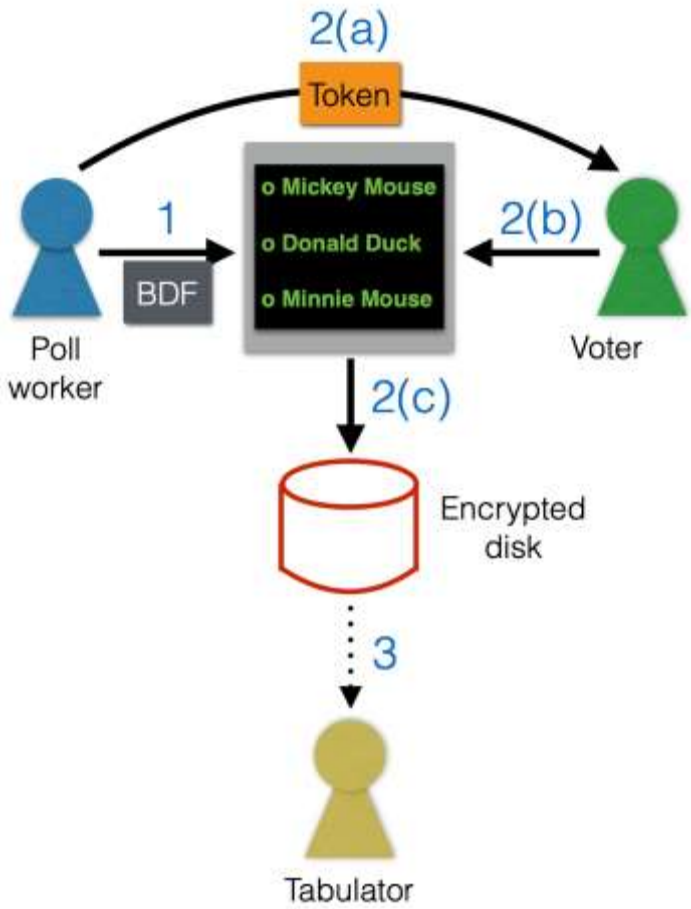


E-voting Analysis

1. Pre-election phase
   - Poll worker loads a "ballot definition file" (defines who's running, colors on the screen, and many more things) on the voting machines with, e.g., USB

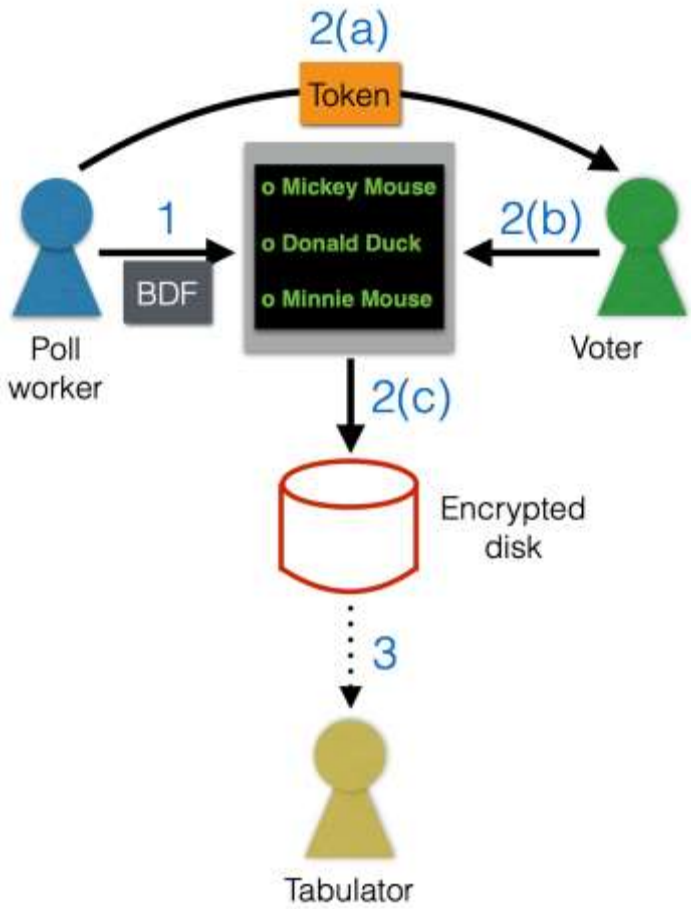
2. Voting phase
   - Voter obtains a single-use token
   - Voter uses the token to interactively vote
   - Vote stored encrypted on disk
   - Voter token canceled

3. Post-election phase
   - Stored votes decrypted and transported to tabulator
   - Tabulator counts and announces vote



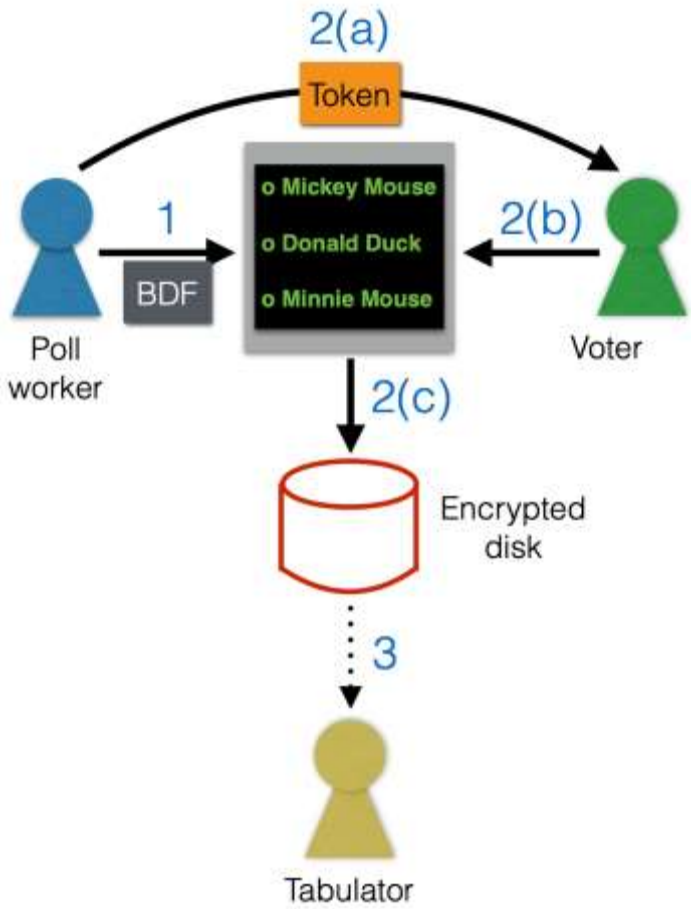
E-voting Analysis

- Identify the assets/goals of the system



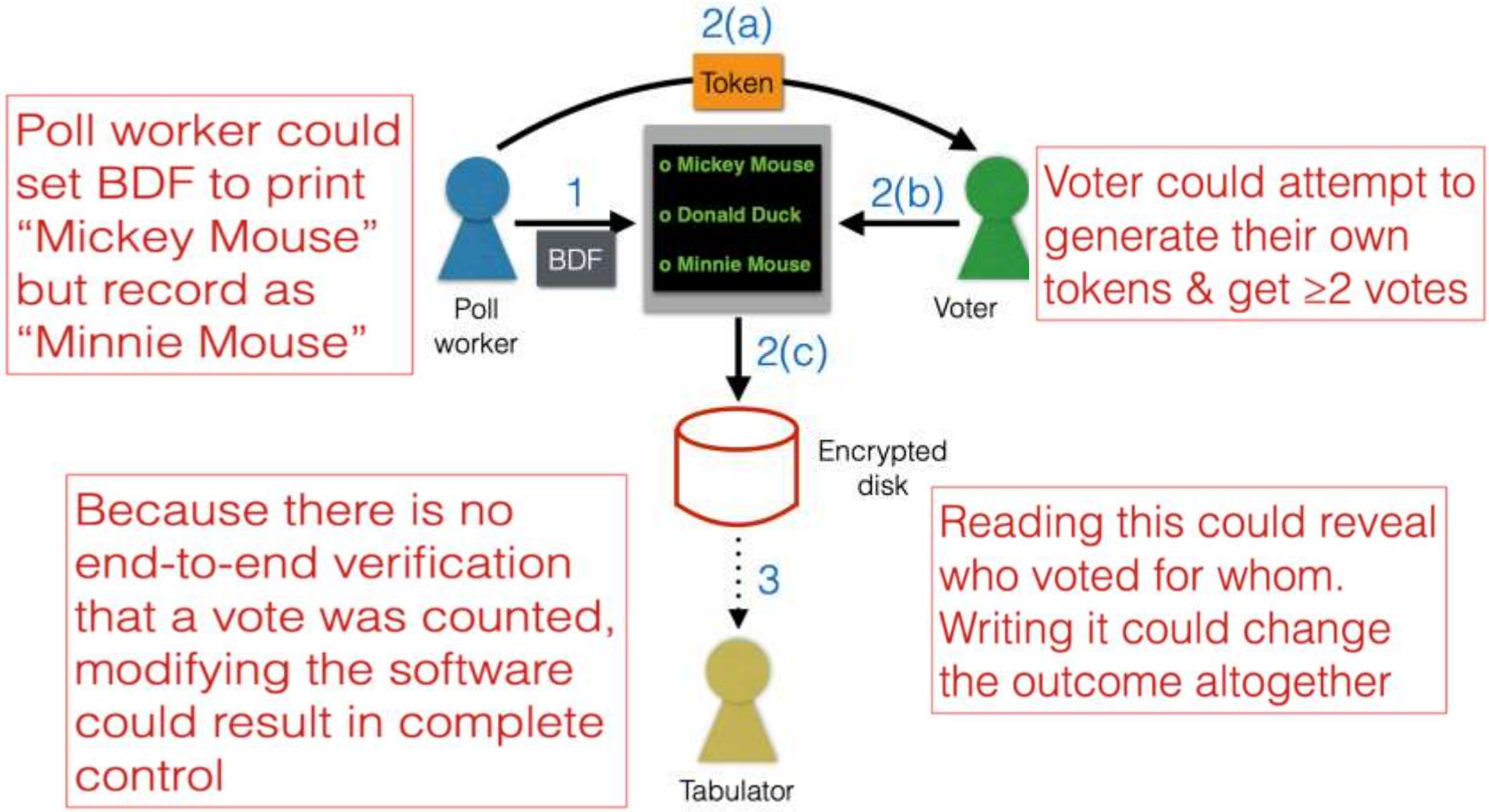
E-voting Analysis

- Confidentiality
  - No one knows for whom any given voter voted (except for the voter)
- Integrity
  - Every voter's vote counted once
  - No voter's vote changed
- Availability
  - Everyone has the ability to cast their vote
- Usability
  - Easy for the voter to vote (correct language, good UI)
  - Easy for the tabulator to count votes



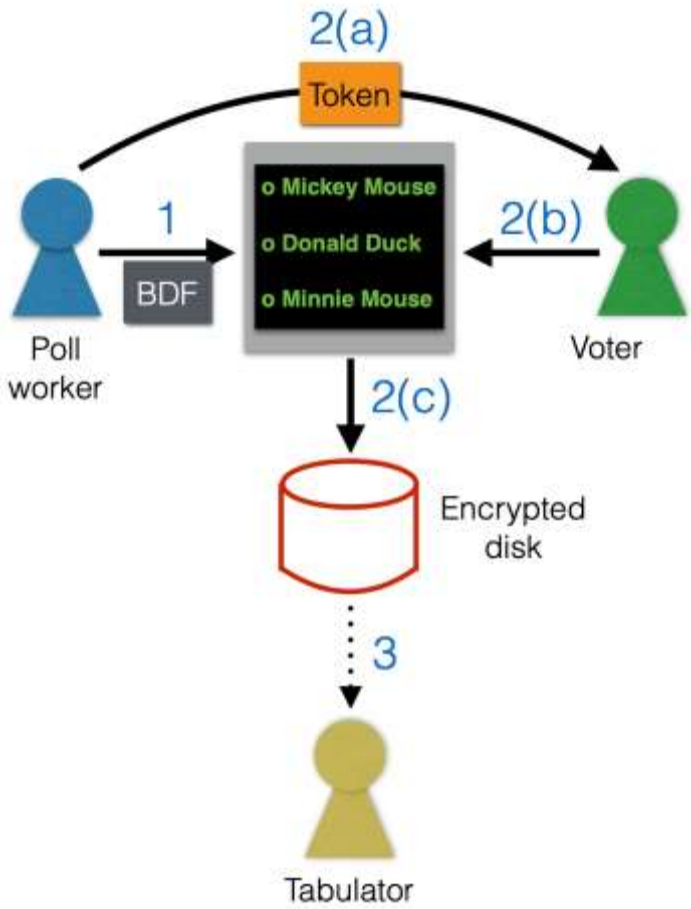
E-voting Analysis

- Identify the adversaries and threats



E-voting Analysis

- Identify the vulnerabilities



E-voting Analysis

- Ballot definition files are not authenticated

- How do we know they're from the election board?

- Can redefine "Candidate A" as "Candidate B"

- Viruses

- Tokens are not authenticated

- How do we know they're not user-generated?

- Possible to make your own and vote multiple times.

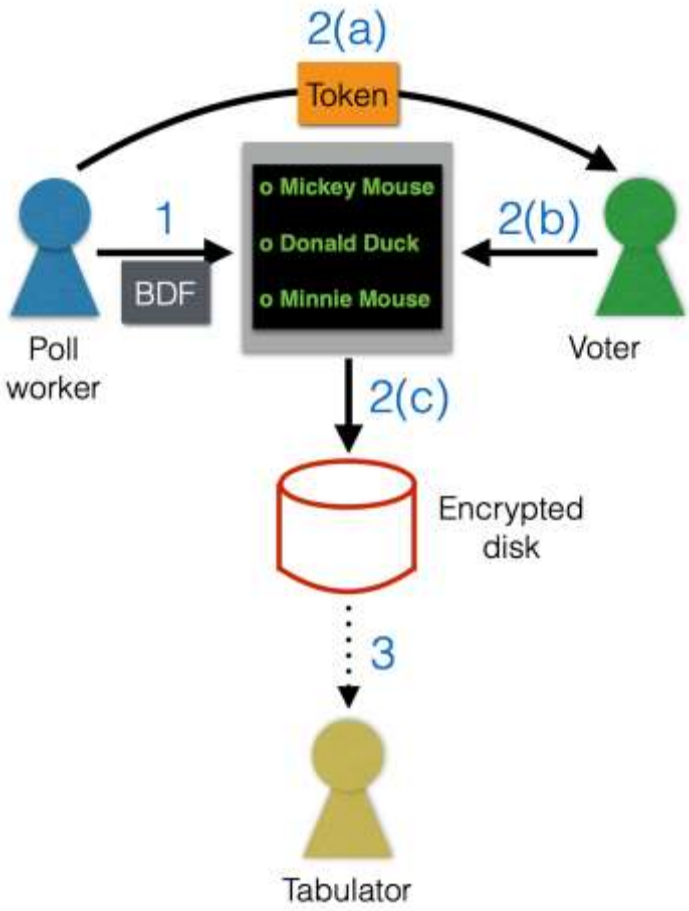
- Specific software vulnerabilities

- Every machine has the same encryption key

- Break one, and they all fall

- Votes are stored in the order casted

- If one can view the data unencrypted, this violates our confidentiality goal



Think like A Defender

- Threat Modeling
- Security Planning

Threat Modeling

- Who are the adversaries
- What they are capable of

Security Planning

- Documentation and planning are an important aspect designing secure programs/systems
- Plans are broken down into security goals, mechanism and controls
  - Security goals should be self explanatory at this point….
  - A control is high level way you are going to guarantee the security goal
  - A mechanism is how the control is implemented
- Example:
  - Goal - Integrity of user authentication
  - Control - Limiting unsuccessful login attempts - The system limits a user to 5 consecutive invalid logon attempts during a 5 minute period and automatically locks the account for 5 minutes when the limit is exceeded
  - Mechanism - The system uses the *Fail2Ban* utility

Fundamental Dilemma of Computer Security

- ***Security-unaware*** users have specific security requirements but usually no security expertise
- This means a few things

  - Users may be bad at enumerating their goals
  - Designing an appropriate control that works against an adversary
  - Figuring out how to correctly implement a mechanism

- Usability vs. Security
  - Example of a completely secure but useless spam filter

We Need Standards

- As a result standards exist that provide "cookbooks" for a variety of scenarios
- Generally if your boss demands you do X, it is because a standard told them so….
- Actually provides a good starting point….
- Does not mean you cannot do more than the standard!

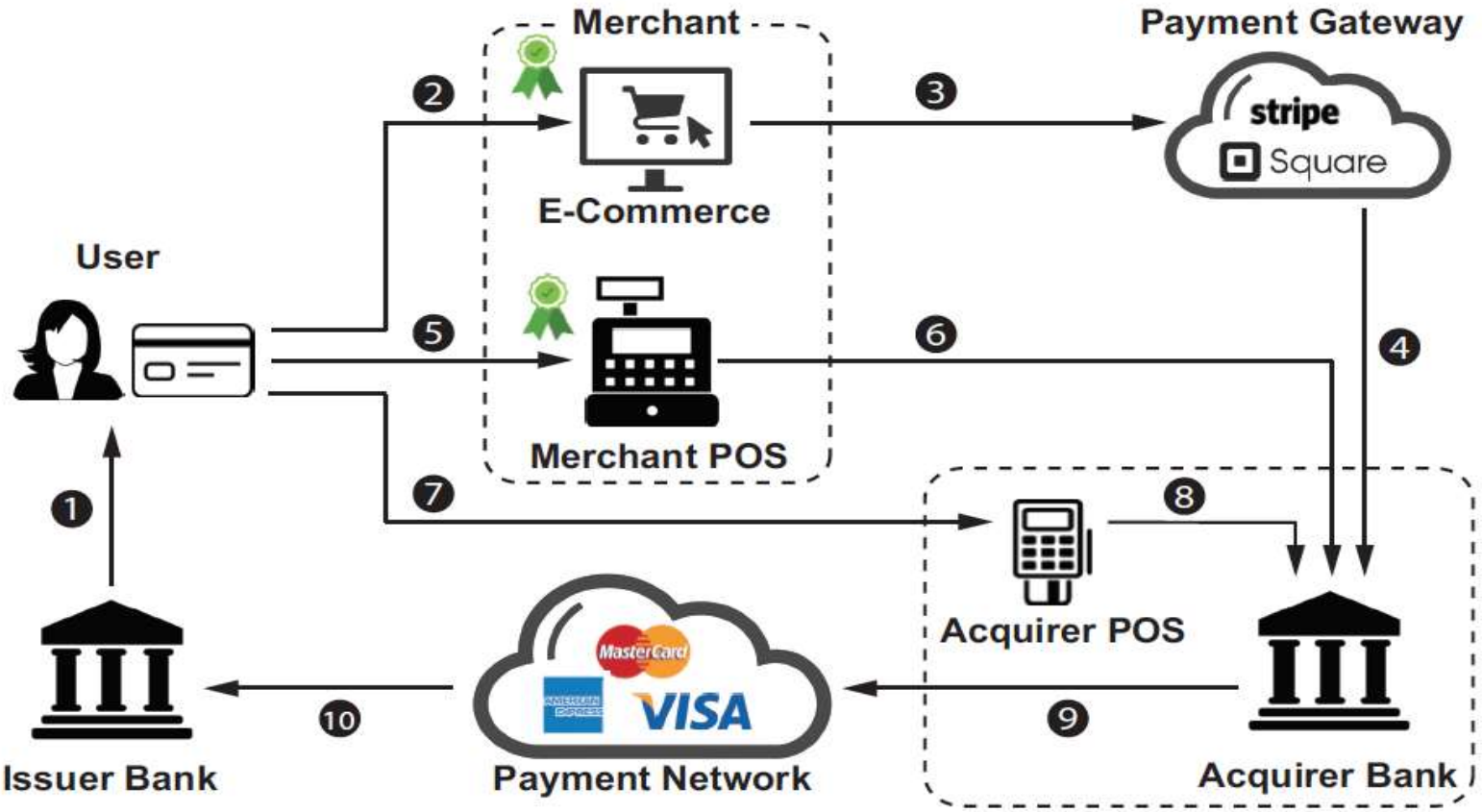
Government and Industry Standards

- NIST FIPS - Federal Information Processing Standards
  - FIPS 140-3: defines security requirements for cryptographic modules
- NIST SP - Special Publications
  - These publications provide guidelines, technical specifications for various aspects of cybersecurity
- NERC CIP – Critical Infrastructure Protection (SCADA)
- PCI DSS – Payment Card Industry Data Security Standard

Cost and Risk Analysis

- Not all solutions to the same problem are the same
  - Some work better than others
  - Some cost more than others

Security Certification in PCI (CCS'19)



Security Certification in PCI

- PCI Security Standards Council requires all merchants and banks to be DSS (Data Security Standard) compliant
  - Merchants and banks must pass security tests and audits from external entities
  - Security tests are done by Approved Scanning Vendors (ASVs) and their scanners remotely
  - Security audits are done by Qualified Security Assessors (QSAs) on-site (>1M transactions)
  - Most small businesses receive certification after passing ASV scanning, large organizations need QSAs

Security Certification in PCI

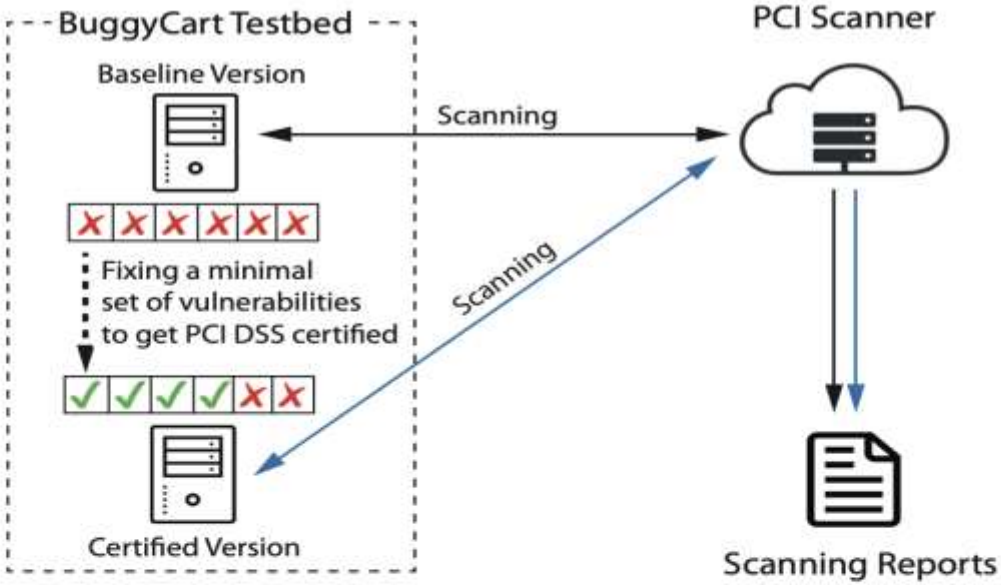


Figure 2: Illustration of the baseline scanning and the certified version. A PCI scanner iteratively scans the testbed. The initial scan (baseline) is on the original testbed with all 35 vulnerabilities. The certified version is the testbed version where the testbed successfully passes the scanning after we iteratively fix a minimal set of vulnerabilities in the testbed. In Table 3, we report the scanning results on both versions of the testbed for each scanner.

Security Certification in PCI

Table 2: Prices of PCI scanners and the actual costs.

PCI Scanners	Price	Spent Amount	PCI SSC Approved?
Scanner1	$2,995/Year	$0 (Trial)	Yes
Scanner2	$2,190/Year	$0 (Trial)	Yes
Scanner3	$67/Month	$335	No
Scanner4	$495/Year	$495	Yes
Scanner5	$250/Year	$250	Yes
Scanner6	$59/Quarter	$118	No
Scanner7	Unknown	N/A	Yes
Scanner8	$350/Year	N/A	Yes
Total	-	$1198	-

17/29 detected, 21 fixes

21/29 detected, 10 fixes

17/29 detected, 10 fixes

Security Certification in PCI

- Scanner 2 is the best
  - Detected 21/29 vulnerabilities and required to fix all 21
- Scanner 1 is the worst
  - Detected 17/29 vulnerabilities and required to fix only 10
  - Same as Scanner 4 but costs a lot more

Cost and Risk analysis

- How do we choose?
- Qualitative
  - "Rate" how costly attacks are and how well solutions mitigate attacks
  - Easier to do, but can lead to bad results
- Quantitative
  - Harder to accomplish (have to define a lot of things that are challenging to define)
  - Leads to results that are more useful from a biz perspective

Qualitative Example

- Open Web Application Security Project (OWASP) Top 10 Lists
  - A risk-awareness documents for web developers, security professionals, and organizations
  - Provides recommendations for protecting against different types of vulnerabilities with qualitative asseessments

Quantitative Example

- Annual Loss Expectancy (ALE)
  - How much does a vulnerability cost us?
  - Actuary sciences give us a way to quantify this:

*Annual Rate of Occurrence * Single Loss Expectancy = ALE*

Example:
- Vulnerability A happens twice a year, and costs $1k per incident
2 * $1,000 = $2,000 ALE
- Vulnerability B happens once every ten years, costs $10k per incident
0.1 * $10,000 = $1,000 ALE

Quantitative Example: DDoS

A DDoS incident costs $10k per incident, we expect 2 attacks per year.

$$ALE = \$10{,}000 * 2 = \$20{,}000$$

Defense X costs $1k / year, does not change the number of attacks per year, but reduces the cost of each incident to $5k.

Defense Y costs $19k / year, does not change the cost of an incident, but reduces the likelihood of an attack to once every 10 years.

Which is better?

Qualitative Example: DDoS

Defense X:

(Adjusted ALE) AALE = $5,000 * 2 = $10,000

$20,000 - $10,000 - $1,000 = $9,000 benefit (ALE - AALE - Cost)

Defense Y:

$10,000 * 0.1 = $1000

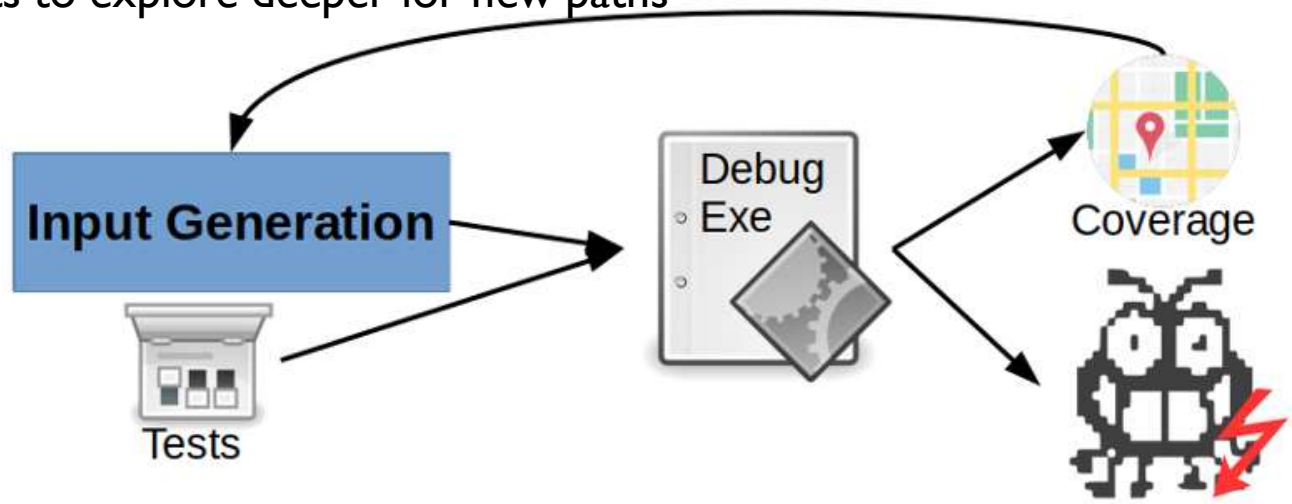
$20,000 - $1,000 - $19,000 = $0 benefit

Once we incorporate the desired mitigations, how can we check if there are any additional flaws or bugs?

Fuzzing

- Automated software testing technique that involves providing invalid, unexpected, or random data as inputs to a computer program

Code coverage guided: generate similar inputs to explore deeper for new paths



Log input and analyze the bugs

Pros and Cons of Fuzzing

- Pros
  - Provides results with little effort: automatic fuzzer is up and running with no interaction
  - Can reveal bugs missed in a manual audit
- Cons
  - Unable to find all bugs: miss bugs that don't trigger a full program crash
  - Unable to know how the software operates internally
  - Requires a lot of power consumption (similar to mining)

Types of Fuzzing

- Mutation-based (Dumb) fuzzing
  - Add anomalies to existing good inputs (e.g., test suite)
    - e.g., Large integers or strings, randomly flip bits
  - Easy to start, may be limited in diversity
- Generative (Smart) fuzzing
  - Generate inputs from specification of format, protocol, etc
  - Test cases are generated from protocol description (e.g., expected input formats)
  - Can have broader exploration, but can be complex to setup
- Evolutionary (Responsive) fuzzing
  - Leverage program feedback (e.g., coverage) to improve the input set
  - Maximize coverage, but complex to setup, resource intensive

Charlie Miller's 5-line Fuzzer

- In 2010, Charlie Miller fuzzed Adobe Acrobat, Apple Preview, PowerPoint, and Open Office by downloading PDF and PPT files and five lines of simple fuzzing:

```
numwrites = random.randrange(math.ceil((float(len(buf)) / FuzzFactor))) + 1
for j in range(numwrites):
    rbyte = random.randrange(256)
    rn = random.randrange(len(buf))
    buf[rn] = "%c"%(rbyte)
```

Charlie Miller's 5-Line Fuzzer

- Randomly changed selected bytes to random values in files
- Produce ~3 million test cases from 1,500 files
- Use standard common tools to determine if crash represents an exploit
  - Acrobat: 100 unique crashes, 4 actual exploits
  - Preview: 250 unique crashes, 60 exploits (tools may over-estimate)
- What kind of fuzzing is it?

Marketplace for Exploits

- What can you do with the exploits you find?
- Yes, you can make a living by doing that if you are good (oftentimes, requires more creative effort)

Marketplace for Exploits

- Option 1: bug bounty programs (white market)
  - Google Vulnerability Reward Program: up to $31,337
  - Microsoft Bounty Program: up to $100K
  - Apple Bug Bounty Program: up to $200K
  - Pwn2Own (hacking contest) competition: $15K

Microsoft Bounty Programs

July 01, 2019 to June 30, 2020

$13.7M

in bounty rewards

15

Bounty programs

1,226

Eligible vulnerability reports

327

Researchers awarded

$200K

Biggest reward

Marketplace for Exploits

- Option 1: bug bounty programs (white market)
  - Google Vulnerability Reward Program: up to $31,337
  - Microsoft Bounty Program: up to $100K
  - Apple Bug Bounty Program: up to $200K
  - Pwn2Own (hacking contest) competition: $15K
- Option 2: grey market
  - Zerodium: Acquiring ***zero-day exploits*** selling it back to customers which mostly include government agencies
    - up to $2M for iOS, $2.5M for Android
    - Not available for the public
- Option 3: black market
  - Do not do this, illegal

Zero Day Exploits