Quantitative Cyber-Security

Colorado State University
Yashwant K Malaiya

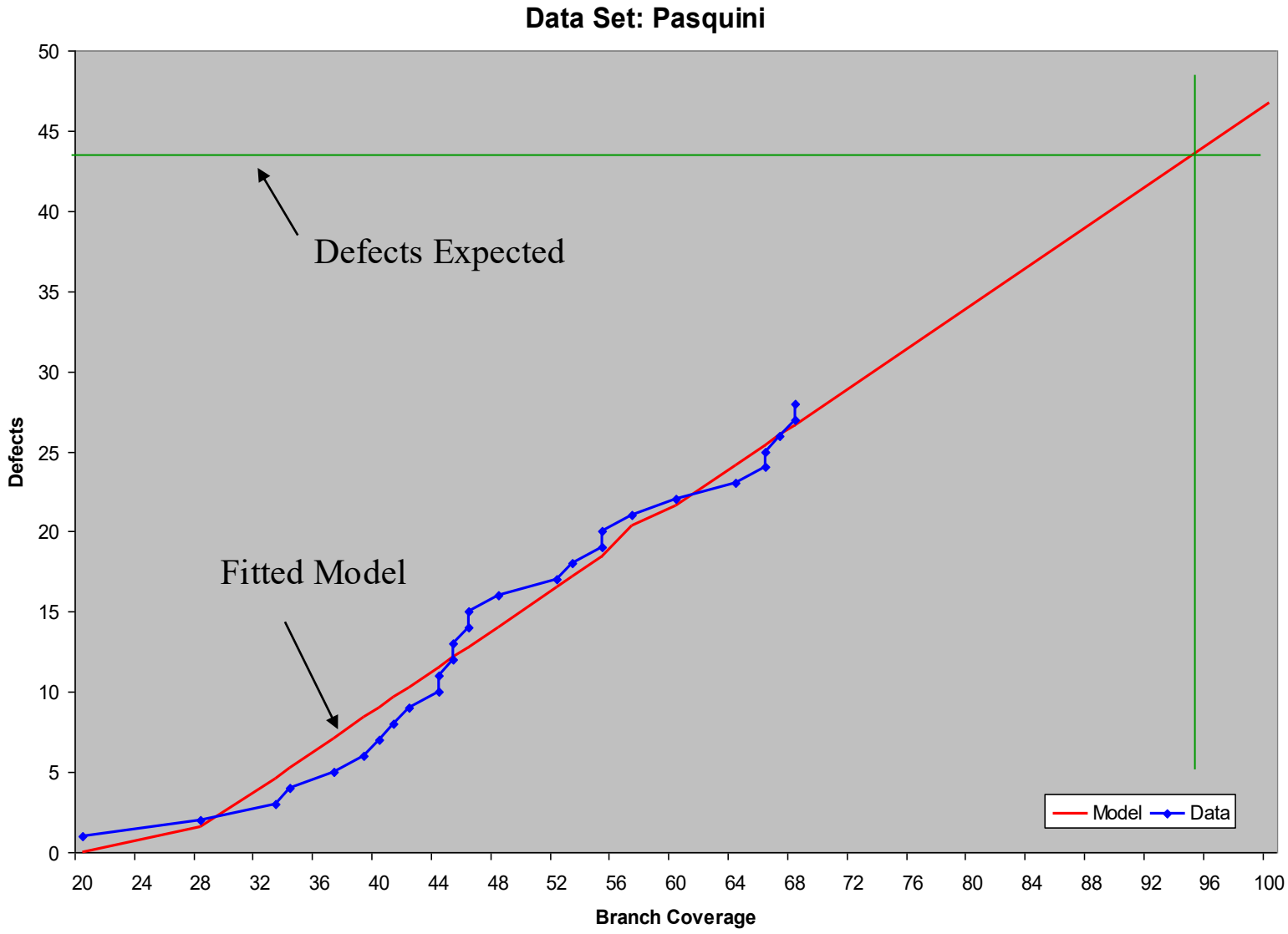
CS559

L18: Coverage, Passwords



CSU Cybersecurity Center
Computer Science Dept

Defects vs. Branch Coverage



Attacking Salted Passwords



hash()

List of possible password hashes

No matches



hashed_ and_salted_ password.txt

cbw	a8
sandi	0X
amislove	hz
bob	K@



hash('0X' + word)

List of possible password hashes w/ salt *0X*

c

sandi YYYY

Hardening Salted Passwords

- Problem: typical hashing algorithms are too fast
  - Enables GPUs to brute-force passwords
- Solution: use hash functions that are designed to be **slow**
  - Examples: bcrypt, scrypt, PBKDF2
  - These algorithms include a work factor that increases the time complexity of the calculation
  - scrypt also requires a large amount of memory to compute, further complicating brute-force attacks

bcrypt Example

- Python example; install the *bcrypt* package

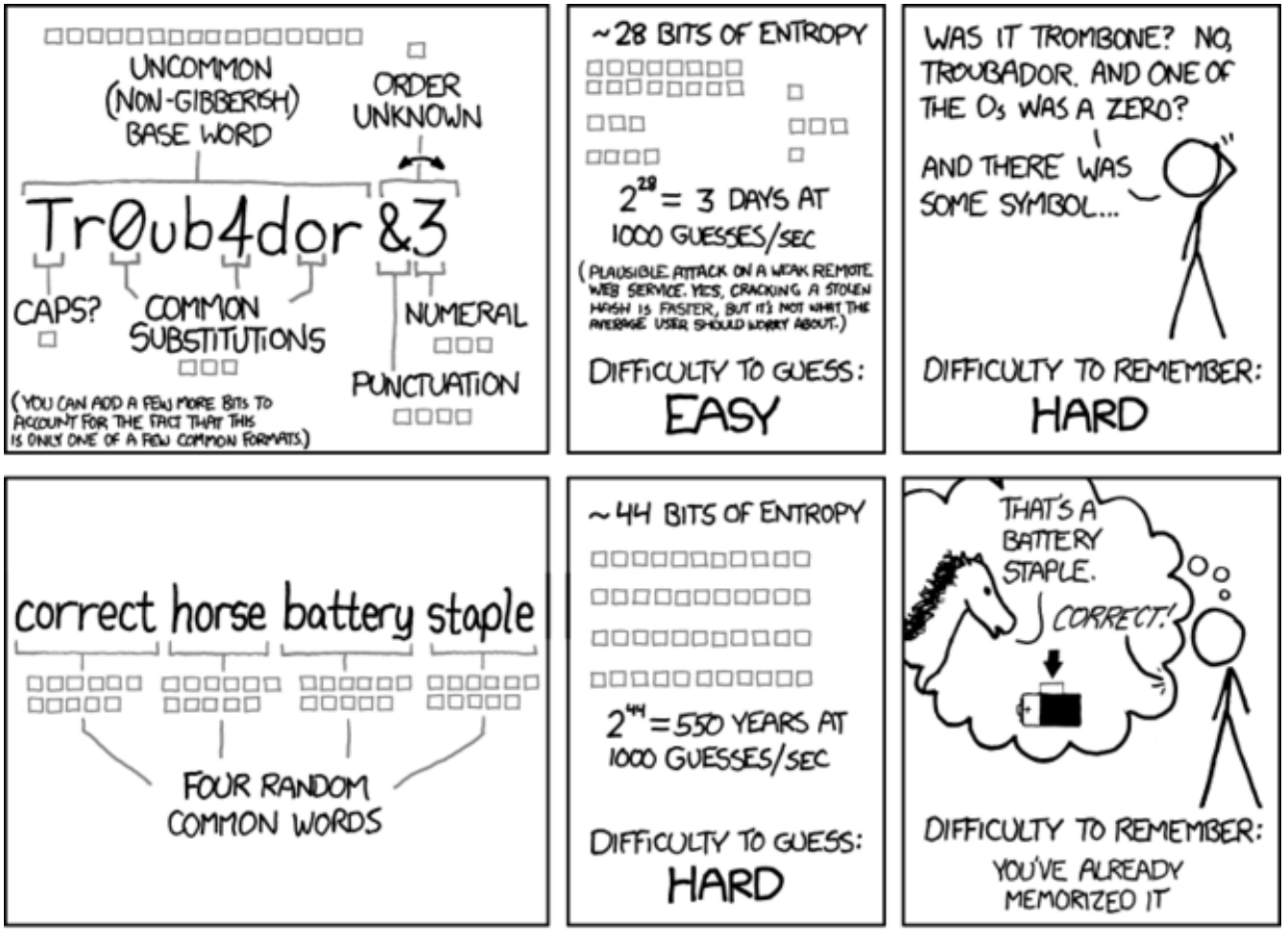
```
[cbw@ativ9 ~] python
>>> import bcrypt
>>> password = "my super secret password"
>>> fast_hashed = bcrypt.hashpw(password, bcrypt.gensalt(0))
>>> slow_hashed = bcrypt.hashpw(password, bcrypt.gensalt(12))
>>> pw_from_user = raw_input("Enter your password:")
>>> if bcrypt.hashpw(pw_from_user, slow_hashed) == slow_hashed:
...         print "It matches! You may enter the system"
...   else:
...         print "No match. You may not proceed"
```

Work factor

Forgotten passwords

- Too many passwords to remember
- “Strong” passwords can be hard to remember
- Traditional approach: user physically requests password reset
  - Using phone numbers or email addresses on record
  - Showing IDs
- Danger: fraudulently obtaining password using social engineering. May represent the weakest link in the password system.

Good passwords are bad



Security Questions

- Security Questions are used to authenticate when
  - Suspicious attempts
  - Forgotten passwords
- 33-39% may be guessed by family members, friends or those with access to personal information
- 20% of users could not remember their own answers.
- Possible solution: Multiple questions with a minimum threshold of right answers

Problem: Password guessing

How easy it is to guess a password?

- If your keyboard has R= 95 unique characters,
- randomly constructing a password from that whole set, 12-character password, then L = 12.
- 95^{12}= 540,360,087,662,636,962,890,625 passwords

Entropy = $\log_2(R^L)$ = 78.9 bits assuming passwords are created randomly

Ascii is 8 bits. Thus about 2^{12x8}

- Non-randomness makes password guessing easier.
- Measures of password strength proposed and used

Password guessing at login? Can be defeated by

- Limited number of tries: 3-5
- Blocking attempts from unknown/suspected IP addresses

Multifactor Authentication

- Smartphone with number xxx-xxx-xxxx: one in 10^{10}
  - About 33 bits of entropy
- Fingerprints might be unique. However, information may be lost when 25-80 minutiae are used for comparison.
  - **False Positive Rate (Erroneous Identification): 0.1%** (two different prints are incorrectly matched as coming from the same source).
  - **False Negative Rate (Erroneous Exclusion): 7.5%** (two prints from the same source are incorrectly judged as not matching)
- Face recognition: 97.25% accuracy
  - **False Positive Rate:** Less than $1x1^{-6}$ in controlled environments
  - **False Negative Rate** : less than 2% in controlled environments
- Iris recognition
  - **False Positive Rate:** $1x1^{-6}$
  - **False Negative Rate:** 0.6-0.7%

Quantitative Cyber-Security

Colorado State University
Yashwant K Malaiya

CS559

L18b: Password Managers



CSU Cybersecurity Center
Computer Science Dept

Password managers

- Can record username, password, form information etc.

for automatic filling.

1. Locally on a device or
2. On the cloud

- Can generate good passwords
- The master password may be kept locally. If you forget it, you may have to extract it yourself.
- Some browsers may include password management capabilities
- Can protect against keyloggers

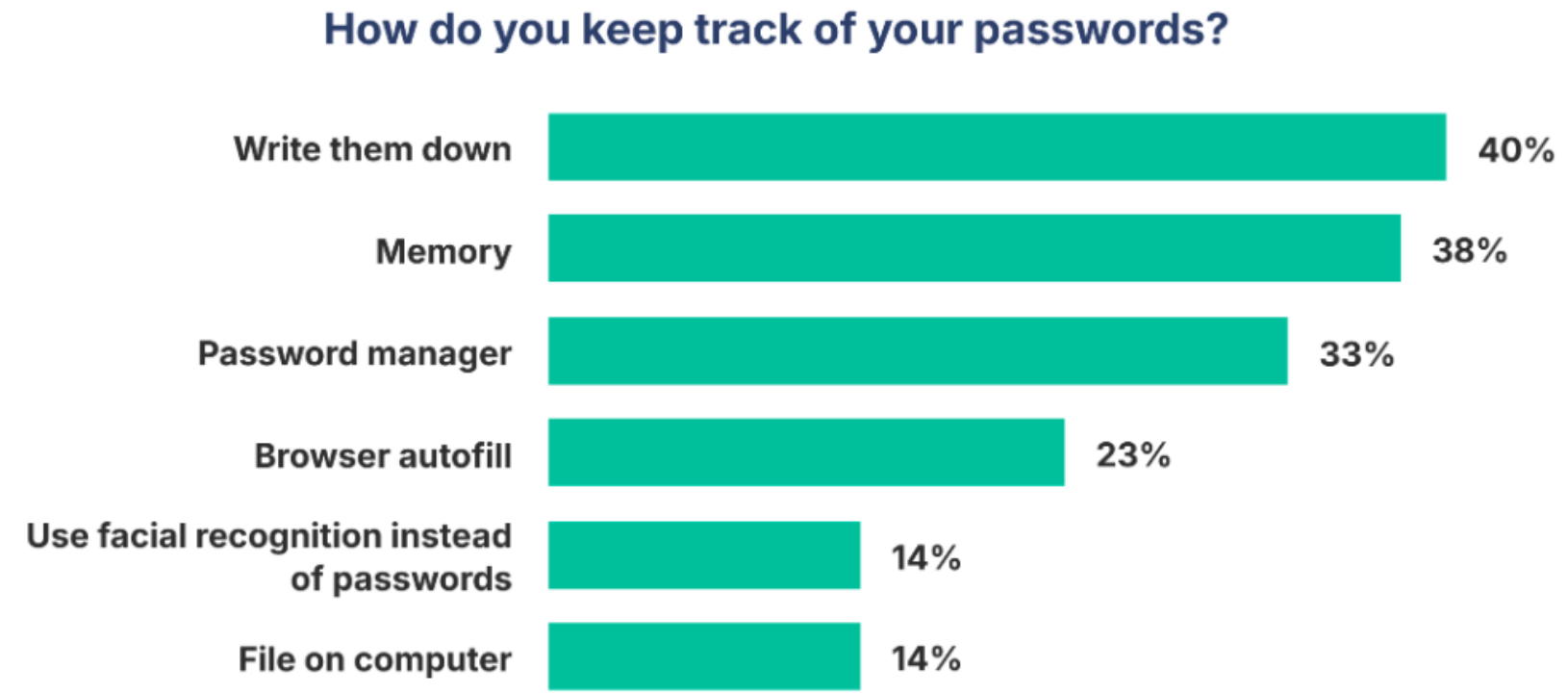
Disadvantages:

- May have vulnerabilities
- May be blocked by some websites

Password guidelines

- **Don’t reuse your passwords.** If you reuse the same password and it is discovered by a hacker, then your other accounts are at risk. Choose a unique password for each account.
- **Store your passwords in a secure password manager.**
- **Don’t share your passwords.** If you need to share your password, change it as soon as possible.
- **Change your password frequently.** For sensitive accounts, change your password at least once a year. For all other accounts, change your password anytime you notice suspicious activity.
- **Enable multi-factor authentication.**
- Set up mechanism to allow to reset the password.
- Use a password manager.

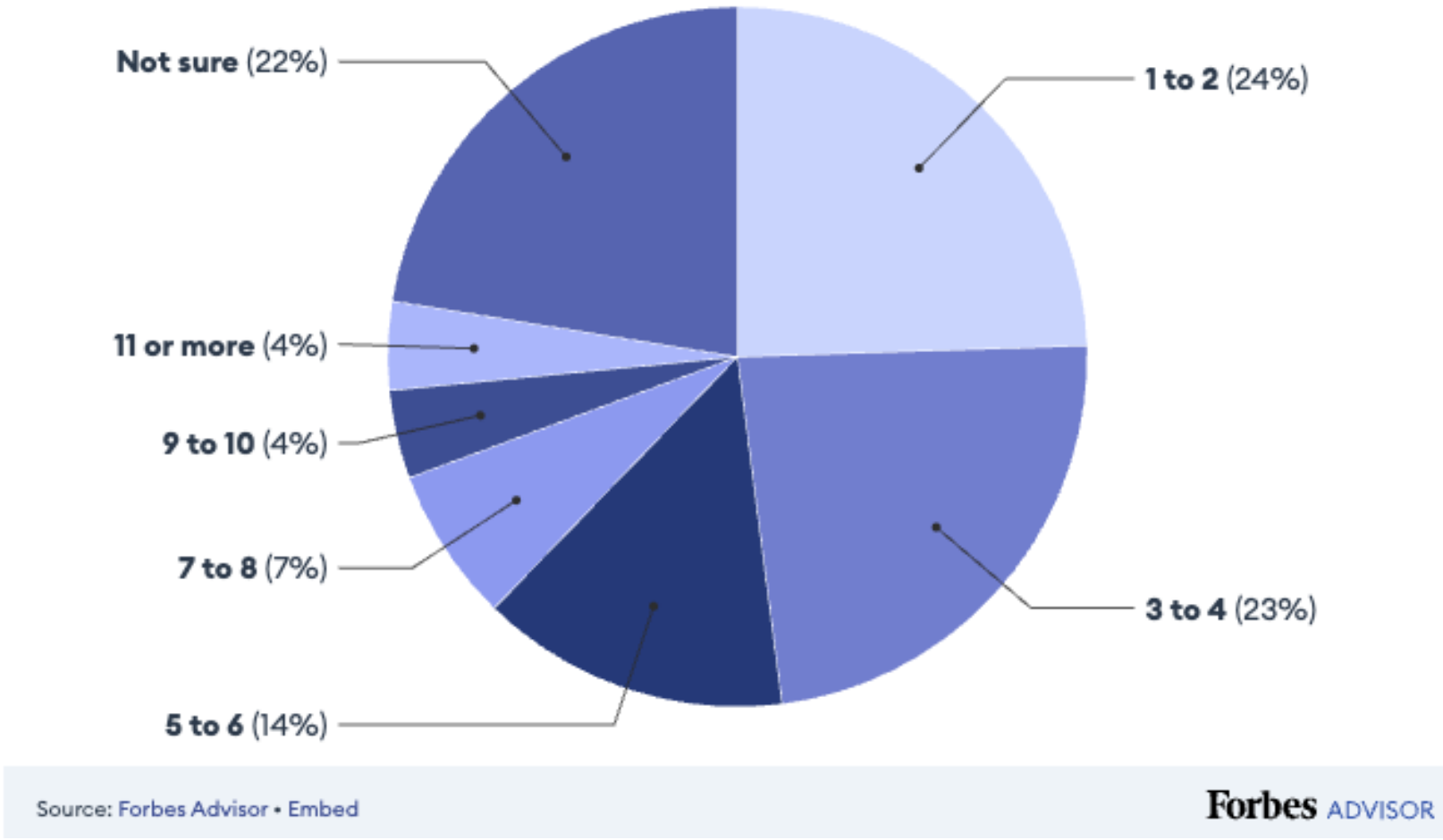
Where people keep their passwords



https://allaboutcookies.org/password-users-behavior-survey

Colorado State University

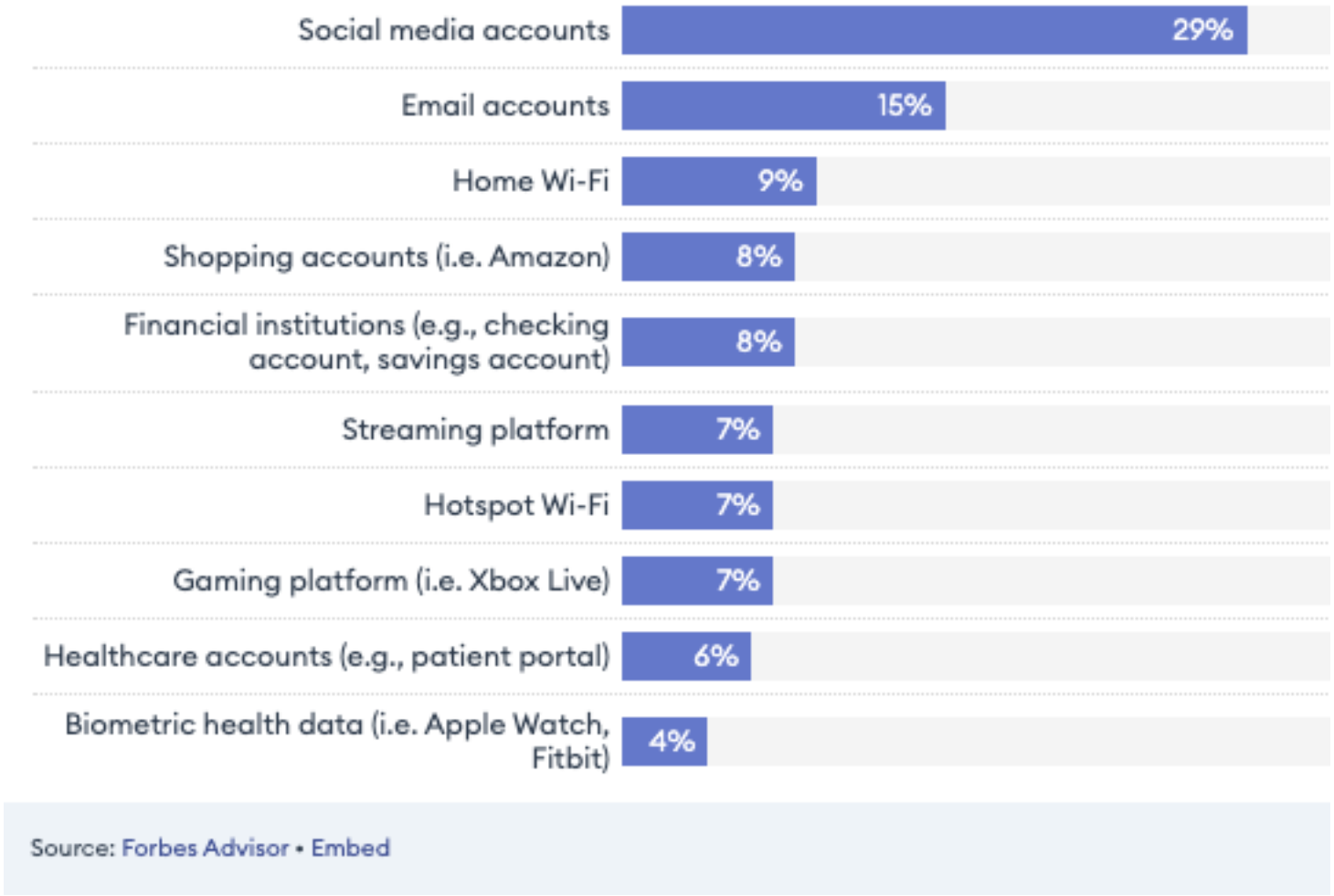
The Number of Accounts Where People Reuse the Same Password



https://www.forbes.com/advisor/business/software/american-password-habits/

Colorado State University

Passwords Are Most Likely to be Stolen



Colorado State University

Need for Password Managers

- Average person needs about 168 passwords. NordPass
- Passwords can be easy to guess/crack. Common strings, reused passwords (52-78% of people) , large password collections
- Passwords are easily forgotten. 78% of the people need a reset within 3 month. Recovery can be time consuming, or prone to social engineering-based discovery.
- Password managers can save time. Password Manager Savings Calculator.

Common Password Managers

Name	Introduced	Million users	Free/Paid	Cost/year Ind/business
LastPass	2008	13.12 23.3%	both	36/p, 72/u b
1Password	2006	6.19. 4.5%	14-d trial	36/p, 96/u b
Roboform	1999	4.87 3.3%	paid	24/p, 35/u b
Keeper		3.4%		
Bitwarden	2016	4.59	both	10/p, 60/u b
Dashlane	2012	3.03 4%	both	78/p, 96/u b
Chrome	2015			
Mac				

What password manager do you primarily use?

What password manager do you primarily use?	2021	2024
Google Password Manager	8%	32%
Apple (iCloud Keychain or Passwords app)	7%	23%
LastPass	21%	11%
Bitwarden	8%	10%
1Password	7%	5%
Norton	3%	3%
Keeper	10%	3%
Dashlane	7%	2%
NordPass	3%	1%

https://www.security.org/digital-safety/password-manager-annual-report/

Cloud vs Local Password Storage

Password vaults is stored either

- Locally on the device (or on premise), may be synched with other devices
  - Will be lost with the device
  - May be hacked by malware
  - More secure?
- On the cloud, log in from any device
  - Recoverable if device lost, can be changed
  - May be stolen with a cloud breach
  - May be encrypted/decrypted at the device level (zero-knowledge architecture)

Master Password

Needed to login into the Password Vault

- Passwords in the vault may be lost if forgotten
- Might be possible to reset the Master Password if the recovery code is available.
- Guidelines:
  - Is unique to your password manager
  - Has not been used in other places online
  - Something you can remember, or write down to remember and put in a secure place
  - Strong: long, random, hard to guess
  - Enable two factor authentication
- Not needed to change frequently, unless evidence of a compromise

Password Generation

- Password Managers generate strong passwords with desirable attributes
  - Random combination of characters with symbols. It will be stored and applied thus no need to store
  - 16 characters better than 12
  - `kdH/3yBqXa*4n7r}`
  - `randomescapeemountoptimal`
  - Can also generate answers to questions

    MOTHER'S MAIDEN NAME? `Smith1980`

Password manager security

Lastpass breach August 2022:

- limited to the service's development environment that isn't connected to customers' data and encrypted vaults.
- LastPass has no access to users' master passwords, which are needed to decrypt their vaults.

Lastpass breach on June 14, 2015

- user email addresses, authentication hashes, password reminders and server per user salts were compromised
- no evidence that user vault data (a user's stored passwords) was taken or that accounts were accessed

https://www.engadget.com/lastpass-hacked-no-user-data-was-compromised-064640557.html

Multifactor Authentication

- Rely on multiple factors of authentication to decrease false positive rate
- Dual authentication by A1 and A2:
  - Pr{false authentication by A1}: p1
  - Pr{false authentication by A2}: p2

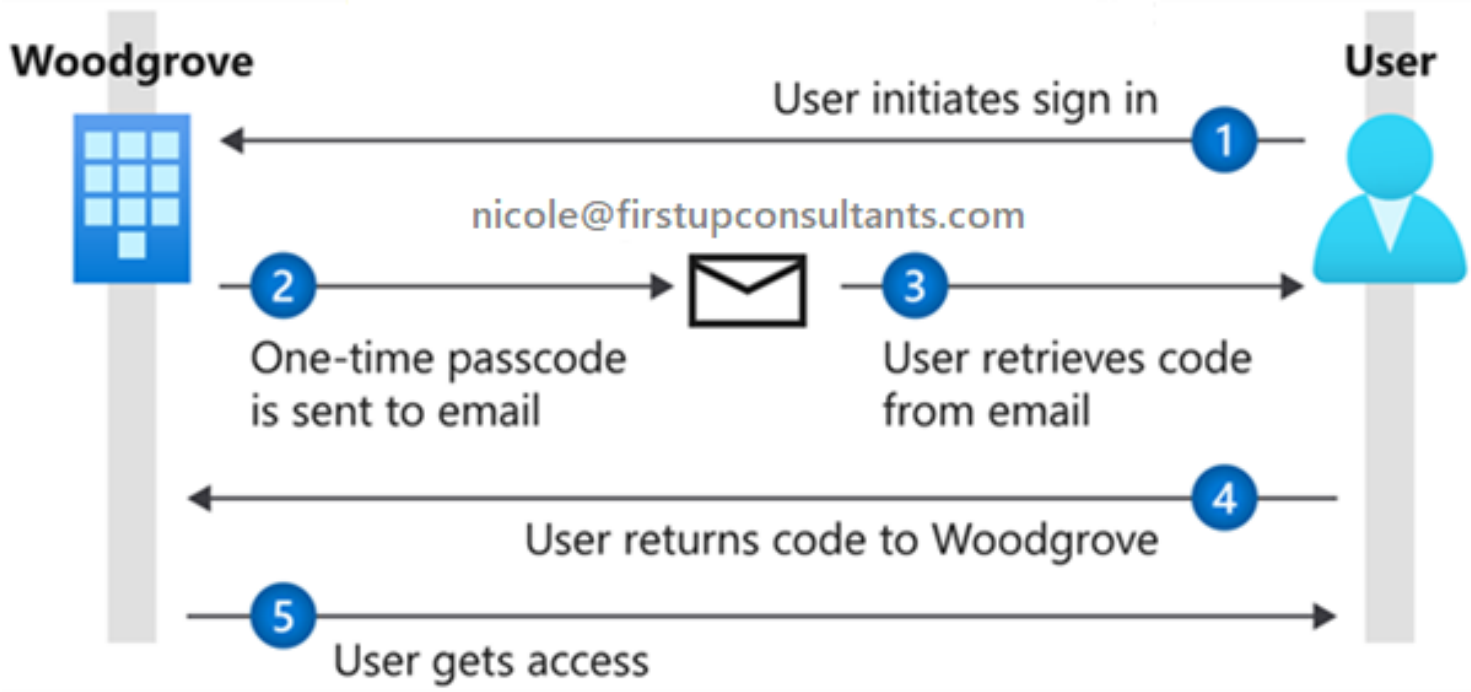
  Pr{Improper Access} = p1.p2

Assuming statistical independence

- Ex: p1 = 2×10^{-3} (two in 1000), p2 = 4×10^{-2}

  Pr{Improper Access} = p1.p2 = 8×10^{-5} orders of magnitude harder
- Second factor authentication may be adaptive
  - Use of a new device
  - First time login
  - Multiple password failures

Smartphone OTP Authentication

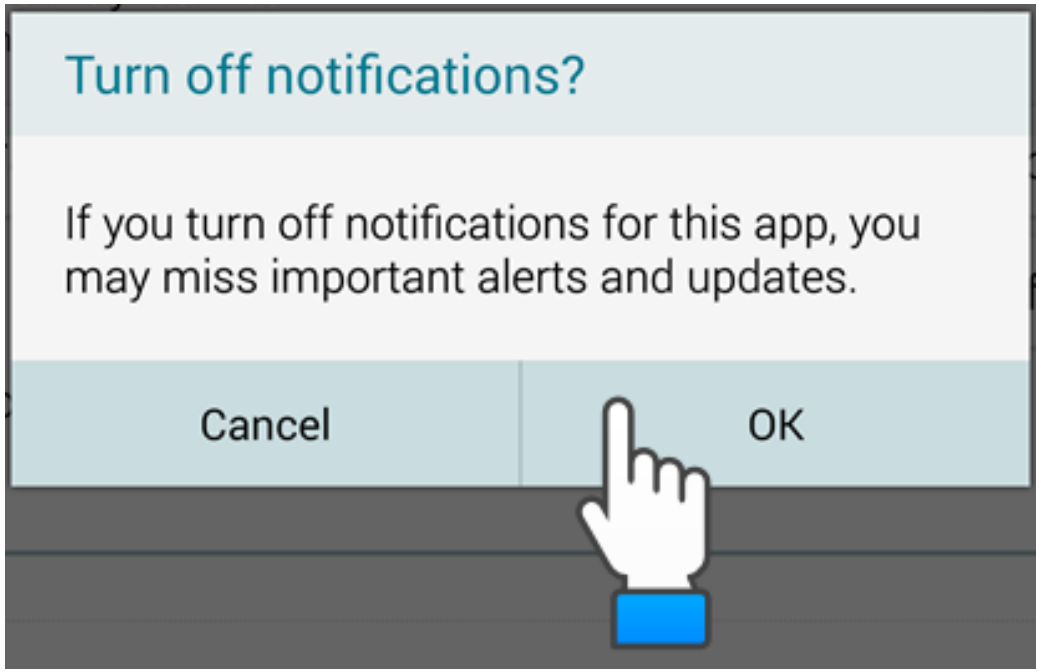


Bank sends OTP (one-time passcode authentication) texted or email to unique phone number/email address.

Colorado State University

Alert fatigue

- Alert messages may be accepted without understanding them, if they are very frequent.
- Some hackers rely on Alert fatigues to make user accept a security risk.



Research questions

- A password manages solves some key password related issues.
- What risks arise because of the use of password manager?
- What are the potential security holes in software, systems and human procedures?
- Any cases of unauthorized extraction of passwords from the password vault?

Peer Interaction

- Your interaction with other student's research is a part of the class.
- You will need to review semi-final report of two fellow students
  - Identify main contributions
  - Strength/weaknesses
  - Suggestions for improvements
  - Suggested additional references
- Presentations: reviews and comments
- Your interaction will be evaluated

Quantitative Cyber-Security

Colorado State University

Yashwant K Malaiya

CS559

Projects



CSU Cybersecurity Center
Computer Science Dept

Term Project Objectives

- Benefit to the fellow students and the instructor
  - Learn something new about an important topic of current interest
  - Contribution may be publishable
- Benefit to the student researcher
  - Learn how to dig out information
  - Learn how to interconnect ideas
  - Identify credibility of sources
  - Make a novel contribution
- Reliance on AI defeats these objectives

Research

- Understand the techniques and results in a chosen field
  - Examine articles from diverse sources
  - Study selectively
- Identify current status, trends, unexplored issues
- Search for information
  - Multiple types of sources
  - Multiple key words
- Search “around” an article
  - Backward search: citations
  - Forward search: cited by (Google scholars)
  - Horizontal search: related publications
- Construct a perspective
  - Identify unsolved problems
  - Novel contribution

Source types

- News (such as Google News)
- Conferences: held once a year, proceedings published
  - Conference, Symposium, …
- Industry publications
  - Magazines, blogs, white papers, product website
- Journals: published several times a year
  - Rigorously reviewed, long publication delay
  - Journal, Transactions, …
- Research groups
  - Industry, academic, consultants: web site
- Books: often well-known stuff
  - Research updates: monographs

Soures of Information Examples

Sources you should keep checking time to time as a researcher.

- Journals: Example:
  - Journal of Cybersecurity
- Conferences:
  - Usenix Security Symposium
- Organizations/Groups:
  - Ponemon Institute
- Industry Publications
  - 2021 Verizon Data Breach Investigations Report
- News
  - Wired (Google News is a useful source)

You should not rely on a few selected keywords on *Google/Google Scholar* to locate information you need.

How to Read Papers: THE THREE-PASS APPROACH

- The first pass: Read (a large number)
  - the title, abstract, and introduction
  - section and sub-section headings, but ignore everything else
  - the conclusions
- The second pass: Read (an intermediate number)
  - figures, diagrams and other illustrations
  - mark relevant unread references for further reading
  - Do you need to read it in detail?
- The third pass: Read critically (closely related)
  - identify and challenge assumption and views
  - Loop up references needed

Keshav, S., How to Read a Paper, ACM SIGCOMM, http://ccr.sigcomm.org/online/files/p83-keshavA.pdf

Evaluation of Research

Similar to paper review for conferences/journals

- Significance and originality
  - Timeliness
  - Originality
- Thoroughness of research
  - Familiar with the field?
  - Has seen significant/recent papers?
- Depth of understanding displayed
- Presentation

Why Citations?

Purpose of citations:

- Acknowledge contributions
- Provide authority behind a claim: Who did the work? Where was it published? When was this work done? Give Page numbers or DOI.
- Inform a reader where the source of a claim can be further examined.

- You must acknowledge: Sources of data, definitions, plots/equations/figures, concepts
- Clearly identify your own ideas and work

Examples of bad citations:

- "IEEE Explore"
- "Google scholar"
- "http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.50.550"

(A reader would think - I'm busy. Any compelling reason I should click?)

Gen AI Guidelines

Overarching Principles in the Use of AI by Graduate Students

- Students remain fully responsible for all scholarly work produced with AI assistance.
- AI may assist but does not assume authorship or accountability
- Transparency is expected whenever AI makes a substantive contribution.
- Human judgement is indispensable for scientific reasoning, interpretation, and ethical decision making. AI should not replace human judgment.
- Compliance with legal, ethical, contractual, and sponsor requirements is mandatory.

Using Generative AI to Support Writing Papers and Theses

- The use of generative AI tools and technologies to create content is permitted but must be **fully disclosed**.
- See department policies.

Do not use AI at the beginning of literature search. Use only after the the problem and the preliminary approach is identified.

AI cannot be used for peer reviews.

Use of AI is detectable (even after ..)

- By an experienced reader
- Using automated tools

Quantitative Security

Colorado State University
Yashwant K Malaiya

CS 559
Fuzzing



CSU Cybersecurity Center
Computer Science Dept

Fuzzing vs. Testing

User Testing for ordinary bugs

- Run program on many **normal** inputs, look for bad things to happen
- **Goal:** Prevent **normal users** from encountering errors

Fuzzing to find vulnerabilities

- Run program on many **abnormal** inputs, look for bad things to happen
- **Goal:** Prevent **attackers** from encountering exploitable errors

Probing input space:

- Vulnerabilities are very sparce
- Only 1-5% of defects are vulnerabilities
- Lot of testing to find vulnerabilities. Automation needed.

Ack: Stanford, Columbia

Types of Fuzzing

- **Mutation-based (Dumb) fuzzing**
  - Add anomalies to existing good inputs (e.g., test suite)
- **Generative (Smart) fuzzing**
  - Generate inputs from specification of format, protocol, etc
- **Evolutionary (Responsive) fuzzing**
  - Leverage program instrumentation, code analysis
  - Use response of program to build input set

Mutation-Based Fuzzing

Basic Idea

- Take known good input and add anomalies
- Anomalies may be completely random or follow some heuristics
  - Large integers or strings
  - Randomly flip bits

HTTP Fuzzing Example

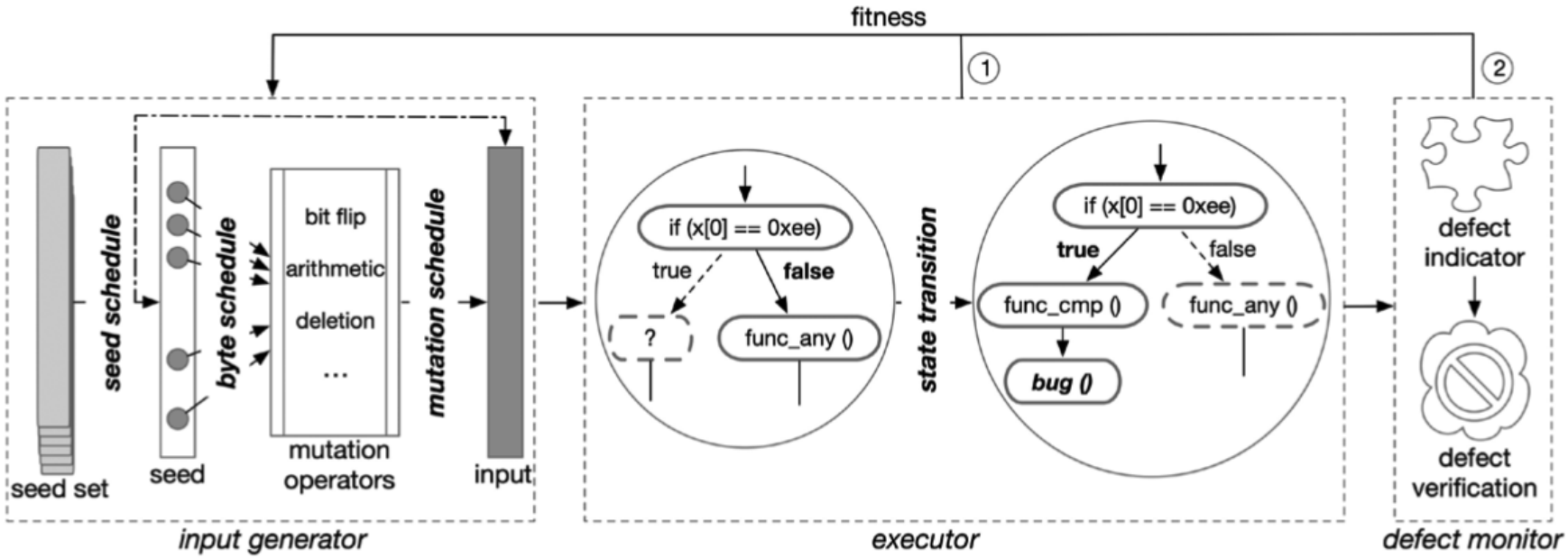
Standard HTTP GET Request

GET /index.html HTTP/1.1

Anomalous Requests

GEEEE…EET /index.html HTTP/1.1
GET ///////index.html HTTP/1.1
GET %n%n%n%n%n%n.html HTTP/1.1
GET /AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA.html HTTP/1.1
GET /index.html HTTTTTTTTTTTTP/1.1
GET /index.html HTTP/1.1.1.1.1.1.1.1
df%w3rasd8#r78jskdflasdjf
4isg8swksdfskdflsdgmsf$gkjs

General workflow of fuzzing



General workflow of fuzzing. Essentially, fuzzing consists of three components: input generator, executor, and defect monitor.

An input is retained as a seed when the input achieves better fitness (e.g., new coverage). Later, fuzzing campaigns will select seeds in the seed set to perform mutations.

Xiaogang Zhu, Sheng Wen, Seyit Camtepe, and Yang Xiang. 2022. Fuzzing: A Survey for Roadmap. ACM Comput. Surv. 54, 11s, Article 230 (January 2022), 36 pages. https://doi.org/10.1145/3512345

Bugs that are detected by fuzzing

- **Memory-violation Bugs** can be classified into two categories, i.e., spatial violations and temporal violations. A spatial violation gets access to memory that is out of bounds, while a temporal violation accesses an invalid referent.
- **Concurrency Bugs** which occurs when concurrent programs run without proper synchronization or order.
- **Algorithm complexity (AC) vulnerabilities** occur when the worst-case complexity of an algorithm significantly reduces the performance, which could result in Denial-of-Service (DoS) attacks.
- **A Spectre-type bug** is a microarchitectural attack that exploits mispredicted branch speculations to control memory accesses.
- **Side-channel bugs** leak secret information via observing non-functional behaviors of a system (e.g., execution time)
- **Integer overflow/underflow** bugs occur when the value of an arithmetic expression is out of the range that is determined by a machine type. Also integer conversion bugs.

Fuzzing PDF Reader

- Download 100s of random PDF files
- Mutate content in the PDF file:
  - flip bits
  - increase size of integers or strings
  - remove data
- Limited by the functionality that the existing files happened to use — unlikely to hit less commonly tested code paths





Mutation-Based Fuzzing

Basic Idea

- Take known good input and add anomalies
- Anomalies may be completely random or follow some heuristics

Advantages

- Little or no knowledge of the structure of the inputs is assumed
- Requires little to no set up time

Disadvantages

- Dependent on the inputs being modified
- May fail for protocols with checksums, challenge-response, etc.

Generation Based Fuzzing

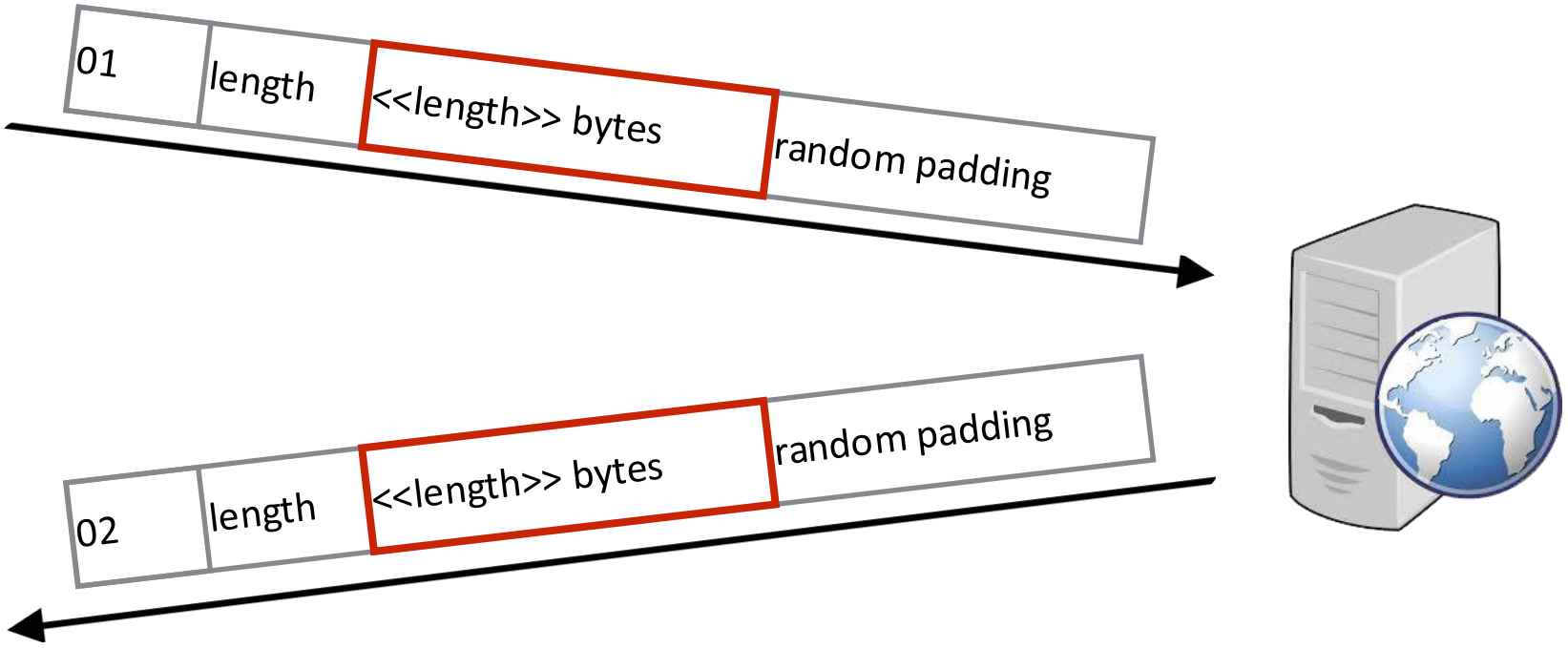
Basic Idea

- Test cases are generated from protocol description: RFC, spec, etc.
- Anomalies are added to each possible spot in the inputs
- Knowledge of protocol should give better results than random fuzzing

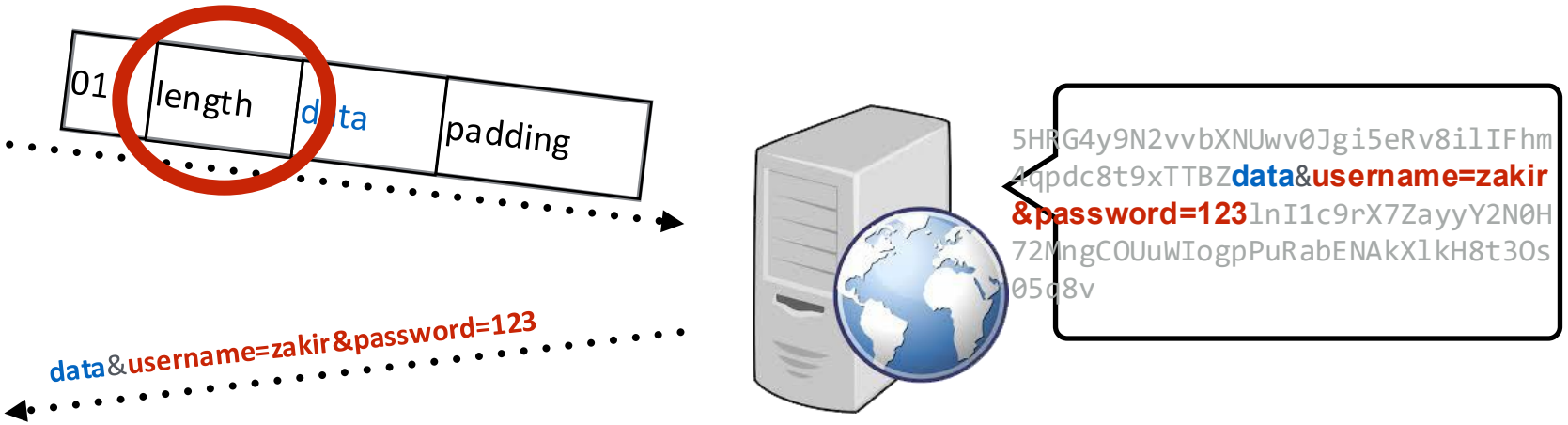
```
<!--  A.  Local file header -->
  <Block name="LocalFileHeader">
    <String name="lfh_Signature" valueType="hex" value="504b0304" token="true" mut
    <Number name="lfh_Ver" size="16" endian="little" signed="false"/>
    ...
    [truncated for space]
    ...
    <Number name="lfh_CompSize" size="32" endian="little" signed="false">
      <Relation type="size" of="lfh_CompData"/>
    </Number>
    <Number name="lfh_DecompSize" size="32" endian="little" signed="false"/>
    <Number name="lfh_FileNameLen" size="16" endian="little" signed="false">
      <Relation type="size" of="lfh_FileName"/>
    </Number>
    <Number name="lfh_ExtraFldLen" size="16" endian="little" signed="false">
      <Relation type="size" of="lfh_FldName"/>
    </Number>
    <String name="lfh_FileName"/>
    <String name="lfh_FldName"/>
    <!--  B.  File data -->
    <Blob name="lfh_CompData"/>
  </Block>
```

Generation Example: TLS Heartbeat

Heartbeat Extension for the Transport Layer Security:
to test and keep alive secure communication links
without the need to renegotiate the connection each time



Generation Example: TLS Heartbeat



Heartbleed Vulnerability: server trusts user provided length field and echoes back memory contents following request data

Mutation-based vs. Generation-based

- Mutation-based fuzzer
  - Pros: Easy to set up and automate, little to no knowledge of input format required
  - Cons: Limited by initial corpus, may fall for protocols with checksums and other hard checks
- Generation-based fuzzers
  - Pros: Completeness, can deal with complex dependncies (e.g, checksum)
  - Cons: writing generators is hard, performance depends on the quality of the spec

How much fuzzing is enough?

- Mutation-based-fuzzers may generate an infinite number of test cases. When has the fuzzer run long enough?
- Generation-based fuzzers may generate a finite number of test cases. What happens when they're all run and no bugs are found?
- Sometimes every anomalous test case triggers the same (boring) bug?

Charlie Miller's 5 Lines

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 Lines

Collect a large number of pdf files

- Aim to exercise all features of pdf readers
- Found 80,000 PDFs on Internet

Reduce to smaller set with apparently equivalent code coverage

- Used Adobe Reader + Valgrind in Linux to measure code coverage
- Reduced to 1,515 files of 'equivalent' code coverage (Test compaction)
- Same effect as fuzzing all 80k in 2% of the time

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 a vulnerability

- Acrobat: 100 unique crashes, 4 actual vulnerability 4%
- Preview: 250 unique crashes, 60 exploits (tools may over-estimate) 24%

Code Coverage

What if we tried to build tests that try to reach code in the program?

Code coverage is a metric which can be used to determine how much code has been executed.

- **Function coverage:** Has each function in the program been called?
- **Edge coverage:** Has every edge in the Control flow graph been executed?
- **Branch coverage:** Has each branch of each control structure been executed?
- **Predicate coverage:** Has each Boolean expression been evaluated to true and false?