Security Design Principles: Open Design

CS-3113: PRINCIPLES OF CYBER SECURITY

BENJAMIN R. ANDERSON

Review: The Eleven Design Principles

General/Fundamental Design Principles

1. Simplicity (related to Economy of Mechanism but not exactly the same)
2. ***Open Design***
3. Design for Iteration (not specifically identified by Saltzer and Schroeder)
4. Least Astonishment (related to Psychological Acceptability)

Security Design Principles

5. Minimize Secrets (not specifically identified by Saltzer and Schroeder)
6. Complete Mediation
7. Fail-safe Defaults
8. Least Privilege
9. Economy of Mechanism
10. Minimize Common Mechanism (related to Least Common Mechanism)
11. Isolation, Separation and Encapsulation

Review: Saltzer and Schroeder's Security Design Principles

Many security issues are a result of poor coding techniques which lead to flaws in the program which can result in a security hole that can be exploited

Saltzer and Schroeder came up with a list of 8 security design principles that, if followed, would help programmers reduce the number of errors and design more secure software

1. Economy of mechanism – A simple design is easier to test and validate
2. Fail-safe defaults –In computing systems, the safe default is generally "no access" so that the system must specifically grant access to resources
3. Complete mediation – Access rights are completely validated every time an access occurs
4. ***Open design –secure systems, including cryptographic systems, should have unclassified designs***
5. Separation of privilege – A protection mechanism is more flexible if it requires two separate keys to unlock it, allowing for two-person control and similar techniques to prevent unilateral action by a subverted individual
6. Least privilege – Every program and user should operate while invoking as few privileges as possible
7. Least common mechanism – Users should not share system mechanisms except when absolutely necessary
8. Psychological acceptability – users won't specify protections correctly if the specification style doesn't make sense to them

Comparison with NIST SP 800-160 Vol. 1

The NIST Special Publications (SP) 800 series is made of publications of interest to the computer security community

- https://www.nist.gov/itl/publications-0/nist-special-publication-800-series-general-information

NIST SP 800-160 Vol. 1: *Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems*

- https://csrc.nist.gov/publications/detail/sp/800-160/vol-1/final

This has superseded earlier documents on this topic (SP 800-27 and SP 800-160) and no longer calls out Open Design specifically

- It does have 32 items in its "Taxonomy of Security Related Design Principles"
- This can be found in Table F-1 in the document
- I encourage you to look at the table, but you will not need to be familiar with these specific items and their descriptions

Open Design

According to this principle, the design of software should not be secret

The mechanism should be public, depending on the secrecy of relatively few key items (like a private key)

- This allows for independent confirmation of the design and implies that ***security through obscurity*** doesn't work
- The assertion is that mechanisms should not depend on the ignorance of potential attackers, but rather on the possession of specific, more easily protected, keys or passwords

Designers and implementers of a program must not depend on secrecy of the details of their design and implementation to ensure security

- These can be figured out through technical means
- For example, the AACS encryption keys were found by muslix64 by looking at a memory dump while using a software player for HD-DVDs
- https://en.wikipedia.org/wiki/AACS_encryption_key_controversy

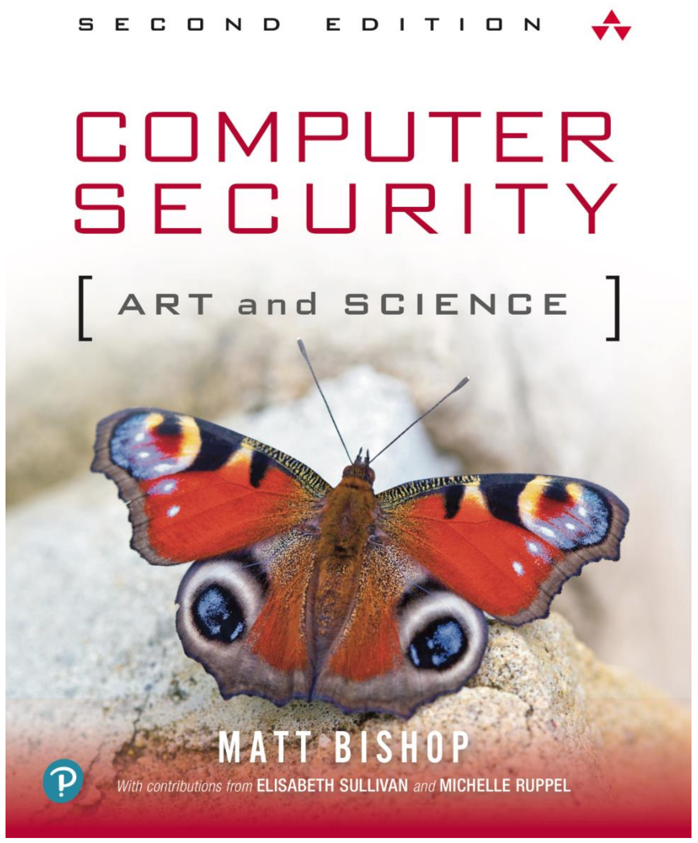
Open Design: Cryptography

Read: Section 14.2.5 *Principle of Open Design* from *Computer Security: Art and Science, 2nd Edition* by Matt Bishop:

- http://nob.cs.ucdavis.edu/book/book-aands2/files/ComputerSecurity2e_CH14.pdf

The argument in cryptography for open design also applies to other security mechanisms

- Open Design allows cryptographic mechanisms (or other security mechanisms) to be examined by many reviewers
- Since the design is open, this review doesn't compromise the safeguard
- In addition, any skeptical users may be allowed to convince themselves that the system they are about to use is adequate for their purpose


SECOND EDITION
COMPUTER SECURITY
[ART and SCIENCE]
MATT BISHOP
With contributions from ELISABETH SULLIVAN and MICHELLE RUPPEL

Open Design: Cryptography

When IBM developed the Digital Encryption Standard (DES) in the 1970s they got help with the design from the NSA

- It was commonly thought that the NSA put in backdoors
- Since DES was an open design, people studied the algorithm (specifically the S-boxes) and discovered that the NSA changes eliminated certain kinds of cryptographic attacks
- https://en.wikipedia.org/wiki/Data_Encryption_Standard#NSA's_involvement_in_the_design

There are some industry concerns about others using the design in their own products

- There are existing laws that deal with the theft of trade secrets
- Non-disclosure agreements (NDAs) can be used if proprietary information is being shared

Note: Remember, the fundamental security design principles:

- Address security – not intellectual property (directly)
- Intellectual property (like a trade secret) would be protected via support for confidentiality

Security Through Obscurity

Wikipedia defines this as:

- *Security through obscurity (or security by obscurity) is the reliance in security engineering on design or implementation secrecy as the main method of providing security to a system or component*
- https://en.wikipedia.org/wiki/Security_through_obscurity

This warning has gone back for over 150 years. From the same Wikipedia page:

- *An early opponent of security through obscurity was the locksmith Alfred Charles Hobbs, who in 1851 demonstrated to the public how state-of-the-art locks could be picked. In response to concerns that exposing security flaws in the design of locks could make them more vulnerable to criminals, he said: "Rogues are very keen in their profession, and know already much more than we can teach them."*

Security Through Obscurity

Port knocking is an example of security through obscurity

- This applies a “secret knock” to gain access to a service
- For example, configuring SSH to only accept connections from an IP address after that IP has tried to connect to port 1234, 5678, and 1337 first
- Video on port knocking by DevOps Directive: https://www.youtube.com/watch?v=IBR3oLqGBj4

You will be responsible for understanding what port knocking is

Drawbacks:

- The pattern can be identified by an attacker monitoring the network
- Replay attacks can be conducted

Single Packet Authorization was designed to address these problems

- Adds 16 bytes of random data (a nonce) to prevent replay attacks
- Uses public key encryption to send a request to open a specific port
  - Can also include a specific protocol like TCP or UDP
- The system only opens the port once it has a valid request

Reference: https://www.linuxjournal.com/article/9565

Obscurity as a Layer

Some also consider obscurity another "layer" in a security model

Read: Daniel Mesier's blog post on the topic

- https://danielmiessler.com/study/security-by-obscurity/#gs.qKEN0lo

Important quote:

- *The key determination for whether obscurity is good or bad reduces to whether it's being used a layer on top of good security, or as a replacement for it. The former is good. The latter is bad.*

Also, review his use of a non-standard port for SSH

- When he changed ports, the connection attempts went from 18,000 to 5!

Remember:

- Obscurity doesn't ***replace*** the security mechanism, it just makes it harder to access
- In the SSH example, you still have to authenticate to the SSH Daemon to gain access

Obscurity as a Layer

Dr. Gene Spafford of Purdue on “security through obscurity”

- “Being deceptive can frustrate attackers, but it’s not a perfect defense all by itself”
- “You can get additional security through obscurity or through deception. But it shouldn’t be your primary form of security”
- Interview at RSA 2017: https://blogs.blackberry.com/en/2017/04/gene-spafford-no-security-through-obscurity

Caveat: Being an open design doesn’t guarantee security – someone still has to look at it!

- Kerberos v4 had an implementation problem in its random number generator
- It was released in 1989 and the vulnerability wasn’t discovered until 1997 by Dole, Lodin, and Spafford
- https://www.ndss-symposium.org/wp-content/uploads/2017/09/Misplaced-Trust-Kerberos-4-Session-Keys.pdf

Dr. Gene Spafford from his RSA 2017 interview

Open Design

Primary Arguments For and Against Open Design

For:

- Your system will actually benefit from having everyone examine its design/implementation
- But remember, someone still has to find it!

Against:

- Don't rely on security through obscurity, because your secrets will leak out eventually
- But remember, you will deter a lot of unskilled attackers (and maybe buy yourself some time)

Disclosure

There are many different perspectives when it comes to the concept of disclosure

- Open Source Paradigm
- Public Domain Paradigm
- Information Sharing Paradigm
- Military Paradigm

Reference: Peter Swire, Ohio State, *A Model for When Disclosure Helps Security: What is Different About Computer and Network* Security, The University of Michigan Law School, The Cyberlaw and Economics Workshop, 2005

Open Source Paradigm

Some assumptions that are made:

- Attackers will learn little or nothing from the disclosure
  - **The thought is**: Once a vulnerability is discovered by an attacker, it will be quickly spread (incorporated into attack tools, or sold)
  - Therefore, public disclosure of a vulnerability will not significantly help attackers exploit the vulnerability
- Disclosure will prompt the designers to improve the design of defenses
  - **The thought is**: Software will improve quickly if a wide array of programmers can see the code, find flaws in it, and fix those flaws
  - To quote Randy Bush and Steven Bellovin:
    - "Hiding security vulnerabilities in algorithms, software, and/or hardware decreases the likelihood they will be repaired."
- Disclosure will prompt other defenders to take action
  - **The thought is**: Many people may be affected by a vulnerability other than the software or system designers
  - Therefore, system owners who learn of the vulnerability can install a patch or upgrade once it is available – and then share these mitigations with the larger community
  - In addition, if a patch is not available, or can't be applied, system owners can take other measures to mitigate the vulnerability
    - Changing system configuration
    - Updating firewall rules
    - Taking a system offline

Reference: Peter Swire, Ohio State, *A Model for When Disclosure Helps Security: What is Different About Computer and Network* Security, The University of Michigan Law School, The Cyberlaw and Economics Workshop, 2005

Open Source Paradigm

However, we still must be careful since, to quote Albert Einstein:

- *Assumptions are made and most assumptions are wrong.*

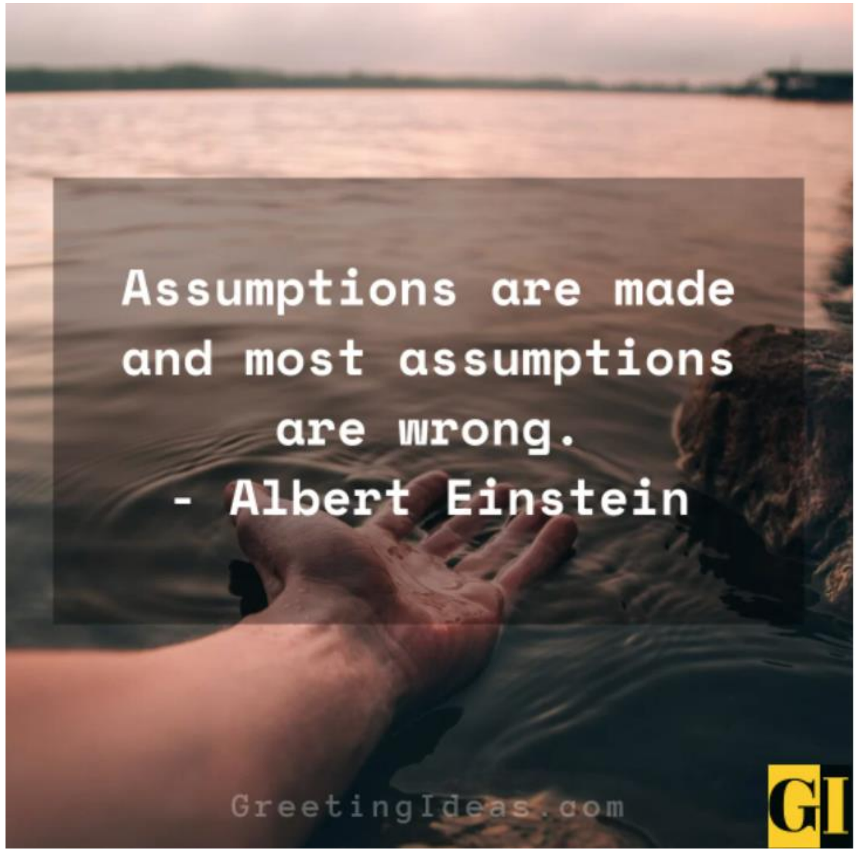


Image from: https://greetingideas.com/dont-make-assumption-quotes-sayings/

Public Domain

Another type of information to consider is public domain information

- Satellite images
- Geographic Information System (GIS) information
- Street map information
- Manuals for hardware or software products

In some cases, this information may be redacted or made unavailable

- Internet services (such as Google Maps) could remove details
- The British removed street signs to confuse attackers during World War II
- However, this may be difficult or costly to achieve

Regarding this information, the assumptions might be:

- Attackers already have this information – so will gain little from it being provided
    - Ex: They have already downloaded Google Earth or software/hardware manuals
- Others may need to reference that information
    - Citizens or system defenders may need that information to conduct their normal operations

Overall, the disclosure needs to outweigh the restriction

Reference: Peter Swire, Ohio State, *A Model for When Disclosure Helps Security: What is Different About Computer and Network* Security, The University of Michigan Law School, The Cyberlaw and Economics Workshop, 2005

Information Sharing Paradigm

Will disclosing information (such as a watch list, or most wanted list) improve or harm security?

Attackers may become alerted they are a "person of interest" if they discover they are on a watch list

- They can change their tactics, be more cautious, or fake their identity
- If it is a larger organization they can have a different member conduct the malicious activities

Wide disclosure of watch lists provides data on what kind of lists are effective

- Not circular logic – think of it like training data for machine learning
- Once the data is collected, optimal rules can be applied to future watch lists

Disclosure of this information can allow defenders to take protective actions

- Allows the detection and identification problem to be distributed
- The TV show *America's Most Wanted* (AMW) allowed millions of viewers to start watching out for fugitives
  - It is alleged that Kevin Mitnick hacked the phone system after seeing himself on AMW so no one could report him

Again, the pros and cons have to be weighed against each other for each situation

Reference: Peter Swire, Ohio State, *A Model for When Disclosure Helps Security: What is Different About Computer and Network* Security, The University of Michigan Law School, The Cyberlaw and Economics Workshop, 2005

Military Paradigm

The default military perspective:

- Secrecy is an essential tool for enhancing security
- Disclosure does not help

For example, in World War 2 there was a huge awareness campaign about not revealing any information

- There was the saying: ***Loose lips sink ships!***
- More posters can be found here: https://www.nh.gov/nhsl/ww2/loose.html

A careless word...
...A NEEDLESS SINKING
SOMEONE TALKED!
LOOK WHO'S Listening
If you tell where he's going... He may never get there!
BITS OF CARELESS TALK ARE PIECED TOGETHER BY THE ENEMY
England
Convoy sails for tonight
DON'T BE A SUCKER!
KEEP YOUR MOUTH SHUT
BUTTON YOUR LIP!
LOOSE TALK CAN COST LIVES

Reference: Peter Swire, Ohio State, *A Model for When Disclosure Helps Security: What is Different About Computer and Network* Security, The University of Michigan Law School, The Cyberlaw and Economics Workshop, 2005

Military Paradigm

This is the opposite of the open source paradigm

Assumptions in the military paradigm:

- Attackers will learn a lot from disclosure of a vulnerability
- Disclosure will teach the designers little or nothing about how to improve the defenses
- Disclosure will prompt little or no improvement in defense by other defenders

We can discuss these in relation to a physical defense like a mine field

- A disclosure regarding the location of the mines would reveal any safe path through it
- The minefield was likely placed in an optimal configuration already
- There will be few, if any, changes by the defenders
  - After all, that would require walking into a mine field!

The same idea holds true of other military technologies

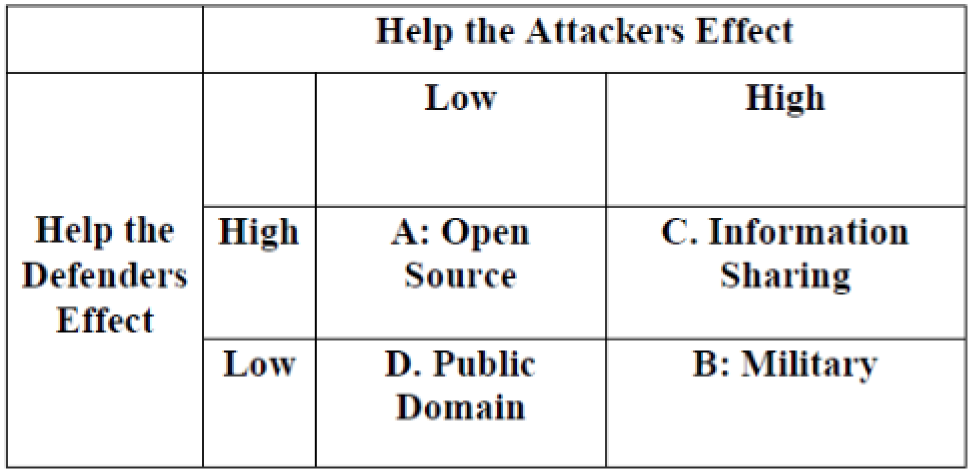
- If an enemy knows what kind of physical or electronic countermeasures a system has (plane, ship, submarine, even a satellite) they can design systems to defeat it
- This can include specific technical information such as frequencies used – which is why this is often classified!

Reference: Peter Swire, Ohio State, *A Model for When Disclosure Helps Security: What is Different About Computer and Network* Security, The University of Michigan Law School, The Cyberlaw and Economics Workshop, 2005

Net Effect of Disclosure

As mentioned previously, we have to weigh Open Design vs. Disclosure

The diagram to the right shows a simple breakdown of the four paradigms and their assumptions on how it helps the attacker or defender



		Help the Attackers Effect	
Help the Defenders Effect		Low	High
	High	A: Open Source	C. Information Sharing
	Low	D. Public Domain	B: Military

Reference: Peter Swire, Ohio State, *A Model for When Disclosure Helps Security: What is Different About Computer and Network* Security, The University of Michigan Law School, The Cyberlaw and Economics Workshop, 2005

Hiddenness and the First-Time Attack

A hidden security feature is likely to be effective against the first attack, but will lose effectiveness once used

- A firewall may block port scans but, once an attacker discovers their scans are being blocked (or the results spoofed) they can use other techniques
- This is one reason adaptive deception systems have been developed
  - An example is the High-fidelity Adaptive Deception & Emulation System (HADES) from Sandia National Laboratories
  - More information on HADES: https://newsreleases.sandia.gov/misleading-hackers/

We can create an equation to determine the effectiveness of ***hiddenness***

Reference: Peter Swire, Ohio State, *A Model for When Disclosure Helps Security: What is Different About Computer and Network* Security, The University of Michigan Law School, The Cyberlaw and Economics Workshop, 2005

Hiddenness Effectiveness Equation

Five variables:

- E – Effectiveness: The effectiveness of the defensive feature at stopping the first attack
- N – Number of attacks
- L – Learning: The extent to which an attacker learns from previous attacks
- C – Communication: The extent to which the attacker communicates this learning to other attackers
- A – Alteration: The extent to which the defenders can effectively alter the defensive feature before the next attack

The effectiveness of hiddenness will vary directly with:

- Greater initial effectiveness (E)
- Greater ability by the designer to alter the defense (A-D) [A-D: Alteration by Designer]

It will vary inversely with:

- The number of attacks (N)
- The degree of learning by attackers (L)
- The ability of attackers to communicate (C)
- The ability of third-parties to alter the defense (A-T) [A-T: Alteration by Third-parties]

When the effects of N, L, and C grow very large, there will be no usefulness of hiding the defensive feature. All attackers will then know everything about the "hidden" feature.

Reference: Peter Swire, Ohio State, *A Model for When Disclosure Helps Security: What is Different About Computer and Network* Security, The University of Michigan Law School, The Cyberlaw and Economics Workshop, 2005

Hiddenness Effectiveness Equation

Things to remember:

- Under the assumptions for Open Source software:
  - Hiddenness will be much less effective
  - The initial effectiveness of a defensive trick, E, will be substantial
  - The number of attacks, N, will most likely be high
- Malicious hackers can probe for weaknesses in a software product repeatedly
  - The attackers learn (L) from the attacks, such as by seeing whether they can gain control over the software
  - Attackers can communicate (C) about flaws, either directly or by incorporating them into attack tools
- Under these assumptions, each attack on a software program is very cheap
  - Attackers can probe the program repeatedly in their own homes or computer labs
  - They can discover flaws and vulnerabilities and share that information (Ex: On the Dark Web)

Reference: Peter Swire, Ohio State, *A Model for When Disclosure Helps Security: What is Different About Computer and Network* Security, The University of Michigan Law School, The Cyberlaw and Economics Workshop, 2005

Another Concept: Uniqueness of Defense

Uniqueness is a function of the Hiddenness variables

- U = f (E, N, L, C, A)
  - E – Effectiveness
  - N - Number of attacks
  - L – Learning
  - C – Communication
  - A - Alternation of Defense

"High Uniqueness" refers to situations where hiddenness is effective

- This is due to:
  - A combination of high values of initial effectiveness (E) and ability to alter the defense (A)
  - And low values for the number of attacks (N), learning from previous attacks (L), and communication among attackers (C)

"Low uniqueness" refers to situations where the values are reversed

Reference: Peter Swire, Ohio State, *A Model for When Disclosure Helps Security: What is Different About Computer and Network* Security, The University of Michigan Law School, The Cyberlaw and Economics Workshop, 2005

Firewalls vs. Physical Walls

Both are a barrier to allow friends in, but attackers out

Both can be configured to meet current needs
- Firewalls can allow different types of protocols, ports, or locations (Geo IP) to have access
- A city's walls can have the gates open, guards stationed, etc.

Both can also be part of a defense-in-depth strategy
- Multiple firewalls (with different rules) guarding different parts of the network
- Inner and outer walls

The image shows the different walls and levels of Minas Tirith from the Lord of the Rings movies

Image from: https://lotr.fandom.com/wiki/Minas_Tirith

Firewalls vs. Physical Walls

Compared to physical walls, a firewall is:

- Subject to a large number of attacks (N)
- Considerable learning by attackers (L)
- And effective communications among attackers (C)

The reasoning for this is:

- Using the Internet, attackers can probe a firewall from anywhere on the planet (or buy the same kind of firewall from the vendor)
- They can attack again and again at low cost, trying various combinations of attacks until they find one that works
- They can then tell other attackers about the vulnerability by posting it on a forum
- If included in a tool, even unskilled attackers can then use it to defeat that firewall, or similarly configured firewalls

Metric	Firewall	Physical Wall
Effectiveness		
Number of Attacks	High	Low
Learning	High	Low
Communication	High	Low
Alteration of Defense		

Firewalls vs. Physical Walls

For a physical wall:

- For attacks against a physical wall to have high N, L, and C it must be poorly defended
- In medieval warfare, an attack against a walled city was a major event in which many people might die
- Any hidden trick by the defenders might cost attackers' lives or save defenders' lives before the attackers learned how to counter the trick
- In general, the number of attacks was low since attackers might not survive to tell about weak spots
- And communication back to the attacking generals was rudimentary.
- Similarly, any hidden weaknesses might not be revealed in time to help the attack.
- In short, N, L, and C would all be low.

This means that medieval city walls had ***high uniqueness*** so secrecy was likely to be a useful tool

Firewalls using standard software likely have ***low uniqueness*** due to the high levels of N, L, and C

Commercial Software

Commercial software – including games and business software – need to be protected from attackers

Some products are deployed in the millions

- Minecraft: 738 million players
- League of Legends: 111 million players (Wikipedia)
- Elden Ring has already sold 13.4 million units (Google)
- Office 365 is used by over a million companies

There are some defenses – like requiring the software to verify licensing on start-up

However, many allow for offline use

- This allows an attacker to install it in a virtual machine (VM) and try multiple attacks before reverting their VM and trying other attacks

References:
https://en.wikipedia.org/wiki/List_of_most-played_video_games_by_player_count
https://www.polygon.com/23070948/elden-ring-sales-chart-npd-call-of-duty-best-sellers
https://www.statista.com/statistics/983321/worldwide-office-365-user-numbers-by-country/

Commercial Software

An attacker that keep trying until something works would have a high N and L

This can also be done by other attackers – and they can communicate with each other – giving a high value for C

For video games, defeating the game can be done with walkthroughs and other information

- Basic strategies for boss battles
- Locations of secret items or warp worlds
- Glitches that can be exploited (Elden Ring teleport glitch)
- Cheat codes (Up, Up, Down, Down, Left, Right, Left, Right, B, A, Start)
- Looking through the game data files

This translates into:

- A high number of attacks – N
- High learning and communication – high L and C
- A hidden measure by a game designer will not stay hidden for long

Encryption

We touched on encryption earlier

A basic rule of cryptography is to use published, public, algorithms and protocols

- This principle was first stated in 1883 by Auguste Kerckhoffs
  - *In a well-designed cryptographic system, only the key needs to be secret; there should be no secrecy in the algorithm*

Bruce Schneier has this quote:

- *Any system that tries to keep its algorithms secret for security reasons is quickly dismissed by the community, and referred to as "snake oil" or even worse.*

If a system is proprietary, when it is actually used – and comes under attack – there is little proof that it will be secure

In addition, it is unlikely that a secret algorithm will remain secret once it is known to a large number of people (or a few determined ones)

- See this example of how the encryption scheme for TV signals was cracked: https://www.theguardian.com/technology/2002/mar/13/media.citynews

Assumptions

Remember what Einstein said about assumptions

The same holds true for software – and transmitted data

If there is a client-side web form, the developer might have set the Zip Code field to only allow the user to enter 5 digits

- This assumes the attacker doesn’t use Burp Suite to intercept the transmission of the web form
- An attacker could then set the field to be 20 digits, 100 digits, or even be letters or symbols
- The developer ***thinks*** they have created a secure form – but they made a bad assumption

Also, the assumption is open source software allows for more people to find bugs

- Have you done a code audit of every open source package you install?
- Are you assuming that the developer won’t provide a malicious update?
  - https://www.bleepingcomputer.com/news/security/dev-corrupts-npm-libs-colors-and-faker-breaking-thousands-of-apps/

Closing

Remember these points

This principle is concerned with security – and not intellectual property

Open Design is one of the most contentious of the fundamental security design principles

- And has even been removed from more recent NIST standards

There are different perspectives on open design between the military, open source, information sharing, and public domain paradigms

- Each scenario makes different assumptions
- Each scenario has different risks

It takes knowledge and experience to make good trade-offs between Open Design and Secrecy