Logistics

- Week 8 exercises due today (as always, let us know if you need any extra time - it's rough out here)
- Project 2 due next Thursday
  - Please accept the GitHub invite if you haven't already
  - We're making Milestone 5 optional but recommended
Today

- How do you keep information safe and sound?
- Could be an entire class by itself!
  - Today’s lecture isn’t even a high-level overview… it’s just a slice of the topic, from the perspective of networked systems design
Recall: In a networked service, a server listens for connections from one or more clients

- When a connection is established, the client sends the server some request (usually using a protocol/“language” like HTTP)
- The server interprets the request and sends some response back over the connection

What threats might we need to defend against if our server has sensitive information?
Today:

- Don’t give information to attackers that ask nicely
- Make sure your dependencies don’t give information to attackers that ask nicely
- Don’t give information to attackers that don’t ask nicely
Level 1: Don’t give information to attackers that ask nicely
Level 1: Don’t give information to attackers that ask nicely

- Stupid attack:

  ![Diagram showing a GET request to server, resulting in HTTP/1.1 200 OK, with the message: The secret sauce is MSG.]

- No one would be that silly, right?
GET /foundation-api/users/uramp/7382194 HTTP/1.1

HTTP/1.1 200 OK

{  
  "customerId": 7382194,  
  "username": "redacted@cox.net",  
  "firstName": "redacted",  
  "lastName": "redacted",  
  "loyalty": {  
    "cardNumber": "redacted"  
  },  
  "emails": [  
    {  
      "id": "redacted",  
      "emailAddress": "redacted@cox.net",  
      "emailType": "Personal",  
      "isDefault": true,  
      "isOpt": true,  
      "isVerified": true  
    }  
  ],  
  "phones": [  
    {  
      "id": 18295989,  
      "phoneNumber": "redacted",  
      "phoneType": "Residential",  
      "countryCode": "1",  
      "extension": null,  
      "name": null,  
      "isSmsOpt": false,  
      "isCallOpt": false,  
      "isDefault": true,  
      "isValid": true,  
      "smsPreferences": [  
        {  
          "groupName": "Delivery",  
          "isOpt": false,  
          "isOptPending": false  
        }  
      ],  
      "isSmsGlobalOpt": false,  
      "isEmailGlobalOpt": true,  
      "isMobilePushOpt": false,  
      "birthDate": {  
        "birthDay": "redacted",  
        "birthMonth": "redacted",  
        "birthYear": "redacted"  
      },  
      "userPreferences": {  
        "foodPreferences": [  
          {  
            "code": 3,  
            "displayName": "Low Fat"  
          }  
        ],  
        "gatherPreference": {  
          "code": 7,  
          "displayName": "Meal with family"  
        }  
      },  
      "subscriptions": {  
        "subscriptions": [  
          {  
            "subscriptionCode": 1,  
            "displayName": "Reward Reminders & Expiration Alerts",  
            "isSubscribed": false,  
            "tncVersion": null  
          },  
          {  
            "subscriptionCode": 2,  
            "displayName": "Panera Bread Updates & Special Offers",  
            "isSubscribed": false,  
            "tncVersion": null  
          }  
        ],  
        "suppressors": [  
          {  
            "suppressionCode": 1,  
            "displayName": "Catering",  
            "isSuppressed": false  
          }  
        ]  
      }  
  }
}
Panera Bread mobile ordering app

- Sequential IDs: you could trivially enumerate every ID and download their entire database
- Case study in how *not* to handle a security breach:
  - Blew off security researcher for 8 months
  - Within two hours of researcher going to the press, announces issue is fixed and only 10k users affected
    - Look at the user ID above! 7382194 >> 10000
  - Did *not* actually fix vulnerability! Same mistake was present on dozens of other API “endpoints” as well as other applications
- [https://medium.com/@djhoulihan/no-panera-bread-doesnt-take-security-seriously-bf078027f815](https://medium.com/@djhoulihan/no-panera-bread-doesnt-take-security-seriously-bf078027f815)
- Note: Not trying to pick on Panera. Bad attitudes towards security are endemic throughout industry (part of the motivation for teaching this class!)
How do we avoid this?
Authentication and authorization

- Authentication: who are you?
  - Established by supplying credentials (e.g. username/password, 2FA authentication token, secret key, biometrics, etc.)
- Authorization: are you allowed to do what you’re trying to do?
  - Established by some security policy (e.g. a user may access his/her own emails, but not the emails of other people)
- A secure service *must* establish both
Common setup

- **Authentication**: clients must demonstrate their identities
- **Authorization**: server must check permission before carrying out request
- Tokens aren’t strictly necessary here, but provide a mechanism for expiring credentials after some time
  - Cookies = tokens

Client

My username is **cactus** and my password is **prickly**

Great! Use this token next time you talk to me: **abc123**

Show me emails for user cactus. My token is **abc123**

Here are emails for user cactus: …

Server

**Validate username/password**

**Validate abc123**

Check that **cactus** has necessary permissions

Authorization
Life without authentication: SaltStack

- Last week, we alluded to clusters of hundreds or thousands of machines used to provide scale and availability.
- You can’t manage that many machines by SSHing in individually.
Life without authentication: SaltStack

- Last week, we alluded to clusters of hundreds or thousands of machines used to provide scale and availability
- You can’t manage that many machines by SSHing in individually
Life without authentication: SaltStack

- SaltStack accidentally exposed a function to network requests that enqueues messages
- Was never intended to be called directly in network requests
Life without authentication: SaltStack

Exactly three weeks ago, companies’ entire clusters started becoming unreachable
- Many of them targeted with bitcoin mining + backdoor
- DigiCert, Algolia, Ghost, Xen Orchestra, LineageOS, others
- Nightmare to fix! Once you manage to get back in, how do you verify attackers aren’t still hiding?
  - [https://duo.com/decipher/saltstack-flaw-used-in-numerous-attacks](https://duo.com/decipher/saltstack-flaw-used-in-numerous-attacks)
  - [https://blog.sonatype.com/saltstack-20-breaches-within-four-days](https://blog.sonatype.com/saltstack-20-breaches-within-four-days)
Life without authorization: LocationSmart

- LocationSmart is a location tracking service that partners with every major US cell carrier and sells location data (e.g. to law enforcement, marketing agencies, companies wanting to track corporate devices)
  - Location data is collected via cell phone tower triangulation. Impossible to opt-out
Life without authorization: LocationSmart

- The company offered a demo website that shows your own location on a map
Life without authorization: LocationSmart

💡 Subscription request:

POST /try/api HTTP/1.1
requestdata={"deviceType":"Wireless","deviceID":"8005551212","devicedetails":"true","carrierReq":"true"}&requesttype=statusreq.json

HTTP/1.1 200 OK
{"uid":"REDACTED","requestTime":"2018-05-16T21:25:50.689+00:00","statusCode":0,"statusMsg":"Success","deviceId":"8005551212","token":"TOKEN","locatable":"True","network":{"carrier":"T-Mobile","locatable":"True","callType":"wireless","locAccuracySupport":"Precise Possible","nationalNumber":"8005551212","countryCode":"1","regionCode":"US","regionCountry":"UNITED STATES"},"subscriptionGroup":[{"name":"LOCA-D01-LOCNOPIN","locatable":"False","smsAvailable":"False"},{"name":"LOCA-D02-WELCOME","locatable":"False","smsAvailable":"False"}],"smsAvailable":"True","privacyConsentRequired":"True","clientLocatable":"false","clientSMSAvailable":"Not supported","whiteListed":false}
Subscription request:
POST /try/api HTTP/1.1
requestdata=
{"deviceType":"Wireless","deviceId":"8005551212","devicedetails":"true","carrierReq":"true"}

POST /try/api HTTP/1.1
requestdata=
{"subscriptionAction":"status","tn":"8005551212","carrierReq":"true"}

Status check:
POST /try/api HTTP/1.1
requestdata=
{"subscriptionAction":"status","tn":"8005551212","carrierReq":"true"}

Life without authorization: LocationSmart

💡 Subscription request:
💡 Status check:
Life without authorization: LocationSmart

💡 Subscription request:
POST /try/api HTTP/1.1
requestdata={"deviceType":"Wireless","deviceId":"8005551212","devicedetails":"true","carrierReq":"true"}&requesttype=statusreq.json

HTTP/1.1 200 OK
{"uid":"REDACTED", "requestTime":"2018-05-16T21:25:50.689+00:00", "statusCode":0,"statusMsg":"Success", "deviceId":"8005551212", "token":"TOKEN", "locatable":"True", …}

💡 Status check:
POST /try/api HTTP/1.1
requestdata={"subscriptionAction":"status","tn":"8005551212","carrierReq":"true"}
&requesttype=subscriptionreq

HTTP/1.1 200 OK
<?xml version="1.0" encoding="UTF-8"?>
<SubscriptionResp>
<uid>REDACTED</uid>
<requestTime>2018-05-17T00:43:44.631+00:00</requestTime>
<statusCode>0</statusCode>
<statusMsg>Success</statusMsg>
<tn>8005551212</tn>
<subscriptionGroup>LOCA-D01-LOCNOPIN</subscriptionGroup>
<subscriptionOptInState>requested</subscriptionOptInState>
<contact>sms</contact>
</SubscriptionResp>
Life without authorization: LocationSmart

💡 Subscription request:
POST /try/api HTTP/1.1
requestdata={"deviceType":"Wireless","deviceID":"8005551212","devicedetails":"true","carrierReq":"true"}&requesttype=statusreq.json

HTTP/1.1 200 OK
{"uid":"REDACTED","requestTime":"2018-05-16T21:25:50.689+00:00","statusCode":0,"statusMsg":"Success","deviceId":"8005551212","token":"TOKEN","locatable":"True", ...}

💡 Status check:
POST /try/api HTTP/1.1
requestdata={"subscriptionAction":"status","tn":"8005551212","carrierReq":"true"}&requesttype=subscriptionreq

HTTP/1.1 200 OK
...

💡 Location request:
POST /try/api HTTP/1.1
requestdata={"civicAddressReq":"True","geoAddressReq":"True","extAddressReq":"True","nearbyPoiReq":"True","privacyConsent":"True","token":"TOKEN","locationtype":"network","accuracyReq":"Coarse","tnDetailReq":"False","carrierReq":"true"}&requesttype=locreq

HTTP/1.1 200 OK
Location data in XML format
Life without authorization: LocationSmart

💡 Subscription request:
POST /try/api HTTP/1.1
requestdata=
{"deviceType":"Wireless","deviceID":"8005551212","devicedetails":"true","carrierReq":"true"} & requesttype=statusreq.json

HTTP/1.1 200 OK
{"uid":"REDACTED","requestTime":"2018-05-16T21:25:50.689+00:00","statusCode":0,"statusMsg":"Success","deviceId":"8005551212","token":"TOKEN","locatable":"True", ...}

🚫 Status check:
POST /try/api HTTP/1.1
requestdata=
{"subscriptionAction":"status","tn":"8005551212","carrierReq":"true"} & requesttype=subscriptionreq

HTTP/1.1 200 OK
...

💡 Location request:
POST /try/api HTTP/1.1
requestdata=
{"civicAddressReq":"True","geoAddressReq":"True","extAddressReq":"True","nearbyPoiReq":"True","privacyConsent":"True","token":"TOKEN","locationtype":"network","accuracyReq":"Coarse","tnDetailReq":"False","carrierReq":"true"} & requesttype=locreq

Error if user has not consented (or location info if they have)
Life without authorization: LocationSmart

💡 Subscription request:
POST /try/api HTTP/1.1
requestdata={"deviceType":"Wireless","deviceID":"8005551212","devicedetails":"true","carrierReq":"true"}&requesttype=statusreq.json

HTTP/1.1 200 OK
{"uid":"REDACTED","requestTime":"2018-05-16T21:25:50.689+00:00","statusCode":0,"statusMsg":"Success","deviceId":"8005551212","token":"TOKEN","locatable":"True"…

⚠️ Status check:
POST /try/api HTTP/1.1
requestdata={"subscriptionAction":"status","tn":"8005551212","carrierReq":"true"}&requesttype=subscriptionreq

HTTP/1.1 200 OK
…

👉 Location request:
POST /try/api HTTP/1.1
requestdata={"civicAddressReq":"True","geoAddressReq":"True","extAddressReq":"True","nearbyPoiReq":"True","privacyConsent":"True","token":"TOKEN","locationtype":"network","accuracyReq":"Coarse","tnDetailReq":"False","carrierReq":"true"}&requesttype=locreq.json

Location info (regardless of whether user consented)
Life without authorization: LocationSmart

- Almost certainly a bad case of copy/paste
- Trivial to exploit
- Overview and context: https://krebsonsecurity.com/2018/05/tracking-firm-locationsmart-leaked-location-data-for-customers-of-all-major-u-s-mobile-carriers-in-real-time-via-its-web-site/
- Technical writeup: https://www.robertxiao.ca/hacking/locationsmart/
How can we prevent this?

- Standard approach: Use a framework that handles every request, checks authentication/authorization, then calls your application code
- Even stronger: Use a type system to make it *impossible* to get information without having the proper privileges
  - The Rocket web framework has request guards, which are types that require some validation to be done in order to be passed to a function
  - \( \text{fn } \text{health}_\text{records}(\text{user: } &\text{SuperUser}) \rightarrow \text{Records} \{ /* \ldots */ \} \)
  - A SuperUser can only be created from a request that has proper permissions
  - Therefore, it is *impossible* to get these records without an authorized superuser asking for them
  - If SaltStack required a SysAdmin to add to the job queue, that vulnerability could not have happened
Level 2: Make sure your dependencies don’t give information to attackers that ask nicely
Level 2: Make sure your dependencies don’t give information to attackers that ask nicely.

These servers have IP addresses too!
Elasticsearch

- “Elasticsearch is a distributed, open source search and analytics engine for all types of data, including textual, numerical, geospatial, structured, and unstructured” (Elastic website)
  - Used for application search, website search, logging and log analytics, infrastructure metrics, geospatial data analysis and visualization, etc.
- Extremely handy! You can throw up an Elasticsearch cluster, throw data in there as it comes in, and quickly run queries on that data
Elasticsearch default settings

- By default, only responds to local connections (i.e. connections coming from the machine Elasticsearch is installed on)
  - This is a problem if you want to use Elasticsearch in the context of a cluster of machines
- No problem! Just change the configuration to accept external connections
Elasticsearch default settings

- By default, only responds to local connections (i.e. connections coming from the machine Elasticsearch is installed on)
  - This is a problem if you want to use Elasticsearch in the context of a cluster of machines
- No problem! Just change the configuration to accept external connections
Misconfigured Elasticsearch Instance Exposes More Than 5 Billion Records. The collections contained information collected by a UK research firm on data breaches from the years 2012 to 2019. An open Elasticsearch instance has exposed more than 5 billion records in an incident discovered on March 16.

www.darkreading.com > attacks-breaches > d-id
Misconfigured Elasticsearch Instance Exposes More ...

Unprotected Elasticsearch Server Leaks 5 Billion Records
Mar 20, 2020 - Around 5,088,635,374 records (more than five billion) were exposed after a U.K.-based security firm inadvertently exposed its "Data breach Database", which stored huge information related security incidents from 2012 to 2019, without password protection. Security researcher Bob Diachenko discovered the leaky database.

www.cisomag.com > News
Unprotected Elasticsearch Server Leaks 5 Billion Records

Elasticsearch Server Exposed 1.2 Billion People Data
Nov 25, 2019 - Recently, almost everyone in Ecuador became a victim of a massive data breach that exposed the personal information of over 20 million ...
Unprotected Elasticsearch Server Leaks 5 Billion Records
Mar 20, 2020 - Around 5,088,635,374 records (more than five billion) were exposed after a U.K.-based security firm inadvertently exposed its "Data breach Database", which stored huge information related security incidents from 2012 to 2019, without password protection. Security researcher Bob Diachenko discovered the leaky database.

Elasticsearch Server Exposed 1.2 Billion People Data
Nov 25, 2019 - Recently, almost everyone in Ecuador became a victim of a massive data breach that exposed the personal information of over 20 million...

Over 1 billion people's data leaked in an unsecured server
Dec 5, 2019 - Elasticsearch servers and personal data. This is not the only data breach of the last few months involving an Elasticsearch server. At the...

ElasticSearch server exposed the personal data of over 57 ...
Nov 28, 2018 - In the meantime, the security researcher has provided a copy of the leaked data to data breach index service Have I Been Pwned (HIBP), and...

A hacker has wiped, defaced more than 15,000 Elasticsearch ...
Apr 4, 2020 - Due to the highly volatile nature of data stored inside Elasticsearch servers, it is...
Data leaks: The most common sources SEE FULL GALLERY.

How to prevent an Elasticsearch server breach - Elastic.co
Feb 24, 2020 - Read about how data breaches come about and how users can best protect against them in the context of Elasticsearch. Learn how to secure...
How to prevent an Elasticsearch server breach - Elastic.co
Feb 24, 2020 - Read about how data breaches come about and how users can best protect against them in the context of Elasticsearch. Learn how to secure ...

250 Million Microsoft Records Exposed in Another ...
Jan 24, 2020 - Let's take a look at this latest breach and why Elasticsearch software appears so often in online data exposure incidences. On December 28 ...

Microsoft exposes 250M customer service records via ...
Jan 22, 2020 - today disclosed a data breach that exposed 250 million customer records via a misconfigured Elasticsearch database. As is somewhat typical ...

People also ask
- What is Elasticsearch server breach?
- What is People Data Labs PDL and OxyData io?
- What is Elasticsearch server?
- What is PDL customer data breach?

An unsecured Elasticsearch server exposed 1.2 billion user ...
Nov 28, 2019 - The Elasticsearch server did not have any kind of authentication The ... US Customs and Border Protection reveal data breach that exposed ...
Five billion records exposed in open 'data breach database ...
Mar 19, 2020 - More than five billion records were exposed after an Elasticsearch "data breach database" managed by a U.K.-based security firm and housing ...

24 million credit and mortgage records exposed on ...
Jan 24, 2019 - An open Elasticsearch database has again been found this time exposing 24.3 million mortgage and credit reports.

Why Do Elasticsearch Databases Keep Getting Hacked ...
Feb 15, 2019 - Don't blame us if your Elasticsearch database shows up online, says Elastic; just learn how to configure the software properly.

Microsoft Error Exposed 250 Million Elasticsearch Records
Jan 23, 2020 - Again, any data leak is bad, but on a scale of 1 to 10 with 10 being the worse, this is easily below 5. Maybe even 3 or lower." Millions of Records.

Data-Enriched Profiles on 1.2B People Exposed in Gigantic ...
Nov 22, 2019 - An open Elasticsearch server has exposed the rich profiles of more than "Data breaches that expose information such as phone numbers to ...

Data about 57 million people exposed by Elasticsearch servers
Jun 22, 2019 - A technical vulnerability in Elasticsearch led to the exposure of personal information about 57 million people.
Data-Enriched Profiles on 1.2B People Exposed in Gigantic ...
Nov 22, 2019 - An open Elasticsearch server has exposed the rich profiles of more than ...
"Data breaches that expose information such as phone numbers to ...

Data about 57 million people exposed by Elasticsearch servers
Nov 28, 2018 - A data breach involving Elasticsearch search-engine technology exposed the personal information of nearly 57 million people for at least two ...

Decathlon Leaks 123 Million Records via Insecure ...
Feb 25, 2020 - French sports giant Decathlon has leaked over 123 million records via an improperly secured Elasticsearch server, according to security ...

Another Big Elasticsearch Data Leak Rocks China, Raises ...
Jul 10, 2019 - The latest leak is in China. Elasticsearch data leaks continue to put millions of people and businesses at risk. The most recent server breach was ...

Elasticsearch database, unprotected and available, exposes ...
Apr 3, 2020 - In a rather odd case of data leakage, researcher Bob Diachenko has uncovered an Elasticsearch database with a large cluster of previously ...

Data Breach- Elasticsearch Server | Ingram Micro Advisor META
There was a breach due to the security misconfiguration in the Elasticsearch database where the permissions were set to "public". How can data breaches affect ...
HIBP “db8151dd” breach

- **Have I Been Pwned** is a free service that will notify you if your information has been found in an online data dump.
- Last year, I was notified my data was compromised in a company’s data breach involving 103M records.
  - Big twist: No one has any idea *which* company!
  - Found on an Elasticsearch instance on the Internet. No one knows who it belongs to.
- Records include social media profiles, contact information, addresses, employment information, and random stuff like “Recommended by Andie [redacted last name]. Arranged for carpenter apprentice Devon [redacted last name] to replace bathroom vanity top at [redacted street address], Vancouver, on 02 October 2007.”
- Excellent read: [https://www.troyhunt.com/the-unattributable-db8151dd-data-breach/](https://www.troyhunt.com/the-unattributable-db8151dd-data-breach/)
Elasticsearch: It’s not our fault

- According to ES, breaches are caused by “a poor understanding of Elasticsearch security and how the software works: ‘Reports usually involve instances where individuals or organizations have actively configured their installations to allow unauthorized and authenticated users to access their data over the internet.’” (source)

- I’m picking on Elasticsearch, but if you Google “S3 data breach” or “MongoDB data breach,” you’ll find just as many severe cases (some are even worse)
Why does this happen?

- **Bad default settings**
  - Databases commonly have a default username and password
  - MongoDB used to accept all network connections by default
  - We’re slowly getting better at this

- **Negligent/inexperienced engineers and system administrators**
  - “I need to access my database from a different server, so let’s open it up on the network!”
  - Systemic problem: Security is often a poorly-understood afterthought in organizations
  - I’m not really sure if we’ve been improving very much

- **We’ve designed systems where the path of least resistance = bad security**
  - It needs to be harder to do things wrong than it is to do things right
  - In many places, only beginning to think about this
As engineers, what should we do?

- We need to think about how to design libraries, frameworks, and systems that are secure by default
  - UI/UX people spend a lot of time thinking about how to design interfaces that are easy to use and hard to misuse
  - As systems people, we really need to do the same!! *Humans* are going to use your stuff, and you need to design for them.
  - As a language, Rust is an excellent example of this
- There’s also a lot of work to be done in figuring out how to improve security for systems we don’t have direct control over
  - E.g. Github has started scanning repositories for passwords, API keys, known vulnerabilities
  - Other systems being developed to proactively monitor the internet for obvious mistakes
Level 3: Don’t give information to attackers that don’t ask nicely
Level 3: Don’t give information to attackers that don’t ask nicely

- Imagine you’re trying to hack into a system. How would you go about it?
- Try the easy things first (e.g. finding obvious weaknesses, or social engineering)
- What if that doesn’t work?
- Next best thing: known vulnerabilities
  - Most of the time, you don’t even need to find new vulnerabilities yourself! People are generally bad at updating software
  - If your target is using outdated software (e.g. HTTP server, graphics library, Linux, you name it) with known bugs, you can simply exploit those bugs
● Ransomware: Encrypts all of the files on your computer and demands Bitcoin payment before you can get them back
● Estimated 200,000 machines infected across 150 countries, up to $4B in economic damage
● Crippled National Health Service in UK: infected computers, MRI scanners, blood storage refrigerators, and more
WannaCry

**Timeline**
- At some point, the NSA discovered an exploitable buffer overflow in the Windows SMB (file sharing) stack. Did not share it with Microsoft (used it for offensive exploits)
- March 14, 2017: Microsoft independently discovers bug, releases patch and security advisory
- April 14, 2017: The Shadow Brokers announce they hacked the NSA, and they release NSA’s EternalBlue exploit
- May 12, 2017: WannaCry begins to spread across the internet
Equifax breach

- Scope: 143 million affected (basically every adult with a credit history in the US)
- March 7, 2017: Apache releases a patch and a security advisory for a critical vulnerability in Apache Struts (web application framework)
- Mid-May 2017: attackers use this vulnerability to get RCE in Equifax systems
- July 29, 2017: Equifax finally discovers the breach
- September 7(!!!), 2017: Equifax finally announces they’ve been hacked
- [https://www.csoonline.com/article/3444488/equifax-data-breach-faq-what-happened-who-was-affected-what-was-the-impact.html](https://www.csoonline.com/article/3444488/equifax-data-breach-faq-what-happened-who-was-affected-what-was-the-impact.html)
- [https://krebsonsecurity.com/2017/09/equifax-breach-response-turns-dumpster-fire/](https://krebsonsecurity.com/2017/09/equifax-breach-response-turns-dumpster-fire/)
Takeaways for engineers

- Take the low-hanging fruit: Updating may be annoying, but being compromised is much worse
- Much of the last decade has been spent trying to figure out how to get people to update
  - Chrome updates in the background
  - Android has tried to move more functionality into apps that can be updated via Google Play, since carriers are bad at updating the OS
  - Windows has forced updates now
  - Still more room for creativity!
- Reduce your attack surface: Don’t expose anything to the Internet that doesn’t need to be exposed to the Internet
Last last resort: Zero day vulnerabilities

- The last resort for an attacker is to find a brand new flaw in your system
- If you want to stop the attackers, you have to find and fix the flaws before they do
- This is really hard! Need to pay people to do this
  - Larger tech companies have dedicated security “red teams” that try to find ways to attack their systems
  - Also a good idea to crowdsource: bug bounty programs pay out to people that find exploitable vulnerabilities
- If you need high security, you should also be looking for bugs in dependencies
  - Heartbleed (2014): Realized everyone uses OpenSSL, but no one pays for it
  - Google operates an incredible team called Project Zero that hunts for bugs in any commonly-used software