Incomplete mediation

- Inputs to programs are often specified by untrusted users
  - Web-based applications are a common example
  - "Untrusted" to do what?
- Users sometimes mistype data in web forms
  - Phone number: `01280807033`
  - Email: `harrys#kaust.edu.sa`
- The web application needs to ensure that what the user has entered constitutes a meaningful request

- This is called mediation

Format string vulnerabilities

- Class of vulnerabilities discovered only in 2000
- Unfiltered user input is used as format string in `printf()`, `fprintf()`, `sprintf()`, `. . .`
- `printf(buffer)` instead of `printf("%s", buffer)`
  - The first one will parse buffer for %'s and use whatever is currently on the stack to process found format parameters
- `printf("%s%s%s%s")` likely crashes your program
- `printf("%x%x%x%x")` dumps parts of the stack
- `%n` will write to an address found on the stack

Why do we care?

- What's the security issue here?

- What happens if someone fills in:
  - DOB: `9876487423649248364924783648923649 2`
    - Buffer overflow?
  - DOB: `'; DROP DATABASE users; --`
    - SQL injection?

https://xkcd.com/327/

- We need to make sure that any user-supplied input falls within well-specified values, known to be safe

Incomplete mediation

- Incomplete mediation occurs when the application accepts incorrect data from the user

- Sometimes this is hard to avoid
  - Phone number: `012-808-0703`
  - This is a reasonable entry, that happens to be wrong
- We focus on catching entries that are clearly wrong
  - Not well formed
    - DOB: `1980-04-31`
  - Unreasonable values
    - DOB: `1876-10-12`
  - Inconsistent with other entries

Client-side mediation

- You've probably visited web sites with forms that do client-side mediation
  - When you click "submit", Javascript code will first run validation checks on the data you entered
  - If you enter invalid data, a popup will prevent you from submitting it
- Related issue: client-side state
  - Many web sites rely on the client to keep state for them
  - They will put hidden fields in the form which are passed back to the server when the user submits the form

Client-side mediation

- Problem: what if the user
  - Turns off Javascript?
  - Edits the form before submitting it? (Tampermonkey)
  - Writes a script that interacts with the web server instead of using a web browser at all?
  - Connects to the server "manually"?

    (telnet server.com 80)
- Note that the user can send arbitrary (unmediated) values to the server this way
- The user can also modify any client-side state

Example

- At a bookstore website, the user orders a copy of the course text. The server replies with a form asking the address to ship to. This form has hidden fields storing the user's order

```
<input type="hidden" name="isbn"  value="0-13-239077-9">
<input type="hidden" name="quantity"  value="1">
<input type="hidden" name="unitprice"  value="111.00">
```

- What happens if the user changes the "unitprice" value to "50.00" before submitting the form?

Defenses against incomplete mediation

- Client-side mediation is an OK method to use in order to have a friendlier user interface but is useless for security purposes.
- You have to do server-side mediation, whether or not you also do client-side.
- For values entered by the user:
  - Always do very careful checks on the values of all fields
  - These values can potentially contain completely arbitrary 8-bit data (including accented chars, control chars, etc.) and be of any length
- For state stored by the client:
  - Make sure client has not modified the data in any way

Example

- A particular Unix terminal program is setuid (runs with superuser privileges) so that it can allocate terminals to users (a privileged operation)
- It supports a command to write the contents of the terminal to a log file
- It first checks if the user has permissions to write to the requested file; if so, it opens the file for writing
- The attacker makes a symbolic link:

```
logfile -> file_she_owns
```

- Between the "check" and the "open", she changes it:

```
logfile -> /etc/passwd
```


TOCTTOU errors

- TOCTTOU ("TOCK-too") errors
  - Time-Of-Check To Time-Of-Use
  - Also known as "race condition" errors
- These errors may occur when the following happens:
  1. User requests the system to perform an action
  2. The system verifies the user is allowed to perform the action
  3. The system performs the action
- What happens if the state of the system changes between steps 2 and 3?

The problem

- The state of the system changed between the check for permission and the execution of the operation
- The file whose permissions were checked for writeability by the user (`file_she_owns`) wasn't the same file that was later written to (`/etc/passwd`)
  - Even though they had the same name (`logfile`) at different points in time
- Q: Can the attacker really "win this race"?
- A: Yes.

Defenses against TOCTTOU errors

- When performing a privileged action on behalf of another party, make sure all information relevant to the access control decision is constant between the time of the check and the time of the action ("the race")
  - Keep a private copy of the request itself so that the request can't be altered during the race
  - Where possible, act on the object itself, and not on some level of indirection
    - e.g. Make access control decisions based on filehandles, not filenames
  - If that's not possible, use locks to ensure the object is not changed during the race

Outline

- Flaws, faults, and failures
- Unintentional security flaws
- Malicious code: Malware
- Other malicious code
- Nonmalicious flaws
- Controls against security flaws in programs

Malware

- Various forms of software written with malicious intent
- A common characteristic of all types of malware is that it needs to be executed in order to cause harm
- How might malware get executed?
  - User action
    - Downloading and running malicious software
    - Viewing a web page containing malicious code
    - Opening an executable email attachment
    - Inserting a CD/DVD or USB flash drive
  - Exploiting an existing flaw in a system
    - Buffer overflows in network daemons
    - Buffer overflows in email clients or web browsers

Types of malware

- Virus
  - Malicious code that adds itself to benign programs/files
  - Code for spreading + code for actual attack
  - Usually activated by users
- Worms
  - Malicious code spreading with no or little user involvement

Types of malware (2)

- Trojans
  - Malicious code hidden in seemingly innocent program that you download
- Logic Bombs
  - Malicious code hidden in programs already on your machine

Viruses

- A virus is a particular kind of malware that infects other files
  - Traditionally, a virus could infect only executable programs
  - Nowadays, many data document formats can contain executable code (such as macros)
    - Many different types of files can be infected with viruses now
- Typically, when the file is executed (or sometimes just opened), the virus activates, and tries to infect other files with copies of itself
- In this way, the virus can spread between files, or between computers

Infection

- What does it mean to "infect" a file?
- The virus wants to modify an existing
- (non-malicious) program or document (the host) in such a way that executing or opening it will transfer control to the virus
  - The virus can do its "dirty work" and then transfer control back to the host
- For executable programs:
  - Typically, the virus will modify other programs and copy itself to the beginning of the targets' program code
- For documents with macros:
  - The virus will edit other documents to add itself as a macro which starts automatically when the file is opened

Infection

- In addition to infecting other files, a virus will often try to infect the computer itself
  - This way, every time the computer is booted, the virus is automatically activated
- It might put itself in the boot sector of the hard disk
- It might add itself to the list of programs the OS runs at boot time
- It might infect one or more of the programs the OS runs at boot time
- It might try many of these strategies
  - But it's still trying to evade detection!

Spreading

- How do viruses spread between computers?
- Usually, when the user sends infected files (hopefully not knowing they're infected!) to his friends
  - Or puts them on a p2p network
- A virus usually requires some kind of user action in order to spread to another machine
  - If it can spread on its own (via email, for example), it's more likely to be a worm than a virus

Spotting viruses

- When should we look for viruses?
  - As files are added to our computer
    - Via portable media
    - Via a network
  - From time to time, scan the entire state of the computer
    - To catch anything we might have missed on its way in
    - But of course, any damage the virus might have done may not be reversable
- How do we look for viruses?
  - Signature-based protection
  - Behaviour-based protection

Payload

- In addition to trying to spread, what else might a virus try to do?
- Some viruses try to evade detection by disabling any active virus scanning software
- Most viruses have some sort of payload
- At some point, the payload of an infected machine will activate, and something (usually bad) will happen
  - Erase your hard drive
  - Subtly corrupt some of your spreadsheets
  - Install a keystroke logger to capture your online banking password
  - Start attacking a particular target website

Polymorphism

- To try to evade signature-based virus scanners, some viruses are polymorphic
  - This means that instead of making perfect copies of itself every time it infects a new file or host, it makes a modified copy instead
  - This is often done by having most of the virus code encrypted
    - The virus starts with a decryption routine which decrypts the rest of the virus, which is then executed
    - When the virus spreads, it encrypts the new copy with a newly chosen random key
- How would you scan for polymorphic viruses?

Signature-based protection

- Keep a list of all known viruses
- For each virus in the list, store some characteristic feature (the signature)
  - Most signature-based systems use features of the virus code itself
    - The infection code
    - The payload code
  - Can also try to identify other patterns characteristic of a particular virus
    - Where on the system it tries to hide itself
    - How it propagates from one place to another

False negatives and positives

- Any kind of test or scanner can have two types of errors:
  - False negatives: fail to identify a threat that is present
  - False positives: claim a threat is present when it is not
- Which is worse?
- How do you think signature-based and behaviour-based systems compare?

Behaviour-based protection

- Signature-based protection systems have a major limitation
  - You can only scan for viruses that are in the list!
  - But there are several brand-new viruses identified every day
    - One anti-virus program recognizes over *36 million* virus signatures
  - What can we do?
- Behaviour-based systems look for suspicious patterns of behaviour, rather than for specific code fragments
  - Some systems run suspicious code in a sandbox first

Worms

- A worm is a self-contained piece of code that can replicate with little or no user involvement
- Worms often use security flaws in widely deployed software as a path to infection
- Typically:
  - A worm exploits a security flaw in some software on your computer, infecting it
  - The worm immediately starts searching for other computers (on your local network, or on the Internet generally) to infect
  - There may or may not be a payload that activates at a certain time, or by another trigger

Base rate fallacy

- Suppose a breathalyzer reports false drunkness in 5% of cases, but never fails to detect true drunkness.
- Suppose that 1 in every 1000 drivers is drunk (the base rate).
- If a breathalyzer test of a random driver indicates that he or she is drunk, what is the probability that he or she really is drunk?
- Applied to a virus scanner, these numbers imply that there will be many more false positives than true positives, potentially causing the true positives to be overlooked or the scanner disabled.

The Code Red worm

- Launched in 2001
- Exploited a buffer overflow in Microsoft's IIS web server (for which a patch had been available for a month)
- An infected machine would:
  - Deface its home page
  - Launch attacks on other web servers (IIS or not)
  - Launch a denial-of-service attack on a handful of web sites, including https://www.whitehouse.gov/
  - Installed a back door to deter disinfection
- Infected 250,000 systems in nine hours

The Morris worm

- The first Internet worm, launched by a graduate student at Cornell in 1988
- Once infected, a machine would try to infect other machines in three ways:
  - Exploit a buffer overflow in the "finger" daemon
  - Use a back door left in the "sendmail" mail daemon
  - Try a "dictionary attack" against local users' passwords. If successful, log in as them, and spread to other machines they can access without requiring a password
- All three of these attacks were well known!
- First example of buffer overflow exploit in the wild
- Thousands of systems were offline for several days

Conficker Worm

- First detected in November 2008
- Multiple variants
- Propagated a command-and-control style botnet
- Security experts had to generate and sinkhole C&C domains
- Number of infected hosts in 2009: 9-15 million, 2011: 1.7 million, 2015: 400,000

The Slammer worm

- Launched in 2003, performed denial-of-service attack
- First example of a "Warhol worm"
  - A worm which can infect nearly all vulnerable machines in just 15 minutes
- Exploited a buffer overflow in Microsoft's SQL Server (also having a patch available)
- A vulnerable machine could be infected with a single UDP packet!
  - This enabled the worm to spread extremely quickly
  - Exponential growth, doubling every 8.5 seconds
  - 90% of vulnerable hosts infected in 10 minutes

Stuxnet

- Very promiscuous: Used 4(!) different zero-day attacks to spread. Has to be installed manually (USB drive) for air-gapped systems.
- Very stealthy: Intercepts commands to SCADA system and hides its presence
- Very targeted: Detects if variable-frequency drives are installed, operating between 807-1210 Hz, and then subtly changes the frequencies so that distortion and vibrations occur resulting in broken centrifuges.

Stuxnet

- Discovered in 2010
- Allegedly created by the US and Israeli intelligence agencies
- Allegedly targeted Iranian uranium enrichment program
- Targets Siemens SCADA systems installed on Windows. One application is the operation of centrifuges
- It tries to be very specific and uses many criteria to select which systems to attack after infection

Trojan horses

http://www.sampsonuk.net/B3TA/TrojanHorse.jpg

WannaCry

- Launched in May 2017, ransomware
- Infected 230,000 computers, including many of the British National Health Service
- Exploits a Windows SMB vulnerability originally discovered by the NSA
- NSA kept it secret (and exploited it?)
- The “Shadow Brokers” leaked it (and others) in April 2017
- Microsoft had released a patch for it a month ago (after being alerted by NSA? Somebody else?) but many systems remained unpatched
- Emergency patch for Windows XP and 8 in May 2017

Trojan horses

- Gain control by getting the user to run code of the attacker's choice, usually by also providing some code the user wants to run
  - "PUP" (potentially unwanted programs) are an example
  - For scareware, the user might even pay the attacker to run the code
- The payload can be anything; sometimes the payload of a Trojan horse is itself a virus, for example

- Trojan horses usually do not themselves spread between computers; they rely on multiple users executing the "trojaned" software
  - Better: users share the trojaned software on p2p

Trojan horses

- Trojan horses are programs which claim to do something innocuous (and usually do), but which also hide malicious behaviour

You're surfing the Web and you see a button on the Web site saying, "Click here to see the dancing pigs." And you click on the Web site and then this window comes up saying, "Warning: this is an untrusted Java applet. It might damage your system. Do you want to continue? Yes/No."

Well, the average computer user is going to pick dancing pigs over security any day.

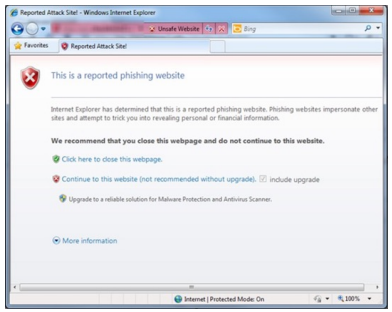
And we can't expect them not to. — Bruce Schneier

Ransomware

https://en.wikipedia.org/wiki/WannaCry_ransomware_attack#/media/File:Wana_Decrypt0r_screensh

Scareware

http://static.arstechnica.com/malware_warning_2010.png



Ransomware

- Demands ransom to return some hostage resource to the victim
- CryptoLocker in 2013:
  - Spread with spoofed e-mail attachments from a botnet
  - Encrypted victim's hard drive
  - Demanded ransom for private key
  - Botnet taken down in 2014; estimated ransom collected between $3 million to $30 million
- Could also be scareware

Logic bombs

- A logic bomb is malicious code hiding in the software already on your computer, waiting for a certain trigger to "go off" (execute its payload)
- Logic bombs are usually written by "insiders", and are meant to be triggered sometime in the future
  - After the insider leaves the company
- The payload of a logic bomb is usually pretty dire
  - Erase your data
  - Corrupt your data
  - Encrypt your data, and ask you to send money to some offshore bank account in order to get the decryption key!

Logic bombs

- What is the trigger?
- Usually something the insider can affect once he is no longer an insider
  - Trigger when this particular account gets three deposits of equal value in one day
  - Trigger when a special sequence of numbers is entered on the keypad of an ATM
  - Just trigger at a certain time in the future (called a "time bomb")

Spotting Trojan horses and logic bombs

- Spotting Trojan horses and logic bombs is extremely tricky. Why?
- The user is intentionally running the code!
  - Trojan horses: the user clicked "yes, I want to see the dancing pigs"
  - Logic bombs: the code is just (a hidden) part of the software already installed on the computer
- Don't run code from untrusted sources?
- Better: prevent the payload from doing bad things
  - More on this later

Other malicious code

- Web bugs (beacon)
- Back doors
- Salami attacks
- Privilege escalation
- Rootkits
- Keystroke logging
- Interface illusions

Outline

- Flaws, faults, and failures
- Unintentional security flaws
- Malicious code: Malware
- Other malicious code
- Nonmalicious flaws
- Controls against security flaws in programs

Web bug example

- On the quicken.com home page:
  - <IMG WIDTH="1" HEIGHT="1" src="http://app.insightgrit.com/1/nat?id=79152388778&ref=http://www.eff.org/ Privacy/Marketing/webbug.html&z=668951 &purl=http://quicken.intuit.com/">
- What information can you see being sent to insightgrit.com?

Web bugs

- A web bug is an object (usually a 1x1 pixel transparent image) embedded in a web page, which is fetched from a different server from the one that served the web page itself.
- Information about you can be sent to third parties (often advertisers) without your knowledge or consent
  - IP address
  - Contents of cookies (to link cookies across web sites)
  - Any personal info the site has about you

Leakage of your identity

- With the help of cookies, an advertiser can learn what websites a person is interested in
- But the advertiser cannot learn person's identity
- ... unless the advertiser can place ads on a social networking site
- Content of HTTP request for Facebook ad:

```
GET [pathname of ad]  Host:ad.doubleclick.nt
Referer:https://www.facebook.com/profile.php?id=
123456789&ref=name
Cookie:  id=2015bdfb9ec...
```


"Malicious code"?

- Why do we consider web bugs "malicious code"?
- This is an issue of privacy more than of security
- The web bug instructs your browser to behave in a way contrary to the principle of informational self-determination
  - Much in the same way that a buffer overflow attack would instruct your browser to behave in a way contrary to the security policy

Examples of back doors

- Real examples:
  - Debugging back door left in sendmail
  - Back door planted by Code Red worm
  - Port knocking
    - The system listens for connection attempts to a certain pattern of (closed) ports. All those connection attempts will fail, but if the right pattern is there, the system will open, for example, a port with a root shell attached to it.
  - Attempted hack to Linux kernel source code

```
if ((options == (__WCLONE|__WALL)) &&
(current->uid = 0))
retval = -EINVAL;
```


Back doors

- A back door (also called a trapdoor) is a set of instructions designed to bypass the normal authentication mechanism and allow access to the system to anyone who knows the back door exists
  - Sometimes these are useful for debugging the system, but don't forget to take them out before you ship!
- Fanciful examples:
  - "Reflections on Trusting Trust" (mandatory reading)
  - "WarGames"

Sources of back doors

- Forget to remove them
- Intentionally leave them in for testing purposes
- Intentionally leave them in for maintenance purposes
  - Field service technicians
- Intentionally leave them in for legal reasons
  - "Lawful Access"
- Intentionally leave them in for malicious purposes
  - Note that malicious users can use back doors left in for non-malicious purposes, too!

Salami attacks

- A salami attack is an attack that is made up of many smaller, often considered inconsequential, attacks
- Classic example: send the fractions of cents of round-off error from many accounts to a single account owned by the attacker
- More commonly:
  - Credit card thieves make very small charges to very many cards
  - Clerks slightly overcharge customers for merchandise
  - Gas pumps misreport the amount of gas dispensed

Privilege escalation

- Most systems have the concept of differing levels of privilege for different users
  - Web sites: everyone can read, only a few can edit
  - Unix: you can write to files in your home directory, but not in /usr/bin
  - Mailing list software: only the list owner can perform certain tasks
- A privilege escalation is an attack which raises the privilege level of the attacker (beyond that to which he would ordinarily be entitled)

Sources of privilege escalation

- A privilege escalation flaw often occurs when a part of the system that legitimately runs with higher privilege can be tricked into executing commands (with that higher privilege) on behalf of the attacker
  - Buffer overflows in setuid programs or network daemons
  - Component substitution (See attack on search path in textbook)
- Also: the attacker might trick the system into thinking he is in fact a legitimate (higher-privileged) user
  - Problems with authentication systems
    - `"-froot"` attack
  - Obtain session id/cookie from another user to access their bank account

Rootkits

- A rootkit is a tool often used by "script kiddies"
- It has two main parts:
  - A method for gaining unauthorized root / administrator privileges on a machine (either starting with a local unprivileged account, or possibly remotely)
    - This method usually exploits some known flaw in the system that the owner has failed to correct
    - It often leaves behind a back door so that the attacker can get back in later, even if the flaw is corrected
  - A way to hide its own existence
    - "Stealth" capabilities
    - Sometimes just this stealth part is called the rootkit

Stealth capabilities

- How do rootkits hide their existence?
  - Clean up any log messages that might have been created by the exploit
  - Modify commands like `ls` and `ps` so that they don't report files and processes belonging to the rootkit
  - Alternately, modify the kernel so that no user program will ever learn about those files and processes!

Example: Sony XCP

- Mark Russinovich was developing a rootkit scanner for Windows
- When he was testing it, he discovered his machine already had a rootkit on it!
- The source of the rootkit turned out to be Sony audio CDs equipped with XCP "copy protection"
- When you insert such an audio CD into your computer, it contains an `autorun.exe` file which automatically executes
- `autorun.exe` installs the rootkit

Example: Sony XCP

- The "primary" purpose of the rootkit was to modify the CD driver in Windows so that any process that tried to read the contents of an XCP-protected CD into memory would get garbled output
- The "secondary" purpose was to make itself hard to find and uninstall
  - Hid all files and processes whose names started with sys
- After people complained, Sony eventually released an uninstaller
  - But running the uninstaller left a back door on your system!

Who installs keyboard loggers?

- Some keyboard loggers are installed by malware
  - Capture passwords, especially banking passwords
  - Send the information to the remote attacker
- Others are installed by one family member to spy on another
  - Spying on children
  - Spying on spouses
  - Spying on boy/girlfriends

Keystroke logging

- Almost all of the information flow from you (the user) to your computer (or beyond, to the Internet) is via the keyboard
  - A little bit from the mouse, a bit from devices like USB keys
- An attacker might install a keyboard logger on your computer to keep a record of:
  - All email / IM you send
  - All passwords you type
- This data can then be accessed locally, or it might be sent to a remote machine over the Internet

Interface illusions

- You use user interfaces to control your computer all the time
- For example, you drag on a scroll bar to see offscreen portions of a document
- But what if that scrollbar isn't really a scrollbar?
- What if dragging on that "scrollbar" really dragged a program (from a malicious website) into your "Startup" folder (in addition to scrolling the document)?
  - This really happened

Kinds of keyboard loggers

- Application-specific loggers:
  - Record only those keystrokes associated with a particular application, such as an IM client
- System keyboard loggers:
  - Record all keystrokes that are pressed (maybe only for one particular target user)
- Hardware keyboard loggers:
  - A small piece of hardware that sits between the keyboard and the computer
    - Works with any OS
    - Completely undetectable in software

Interface illusions

- We expect our computer to behave in certain ways when we interact with "standard" user interface elements.
- But often, malicious code can make "nonstandard" user interface elements in order to trick us!
- We think we're doing one thing, but we're really doing another
- How might you defend against this?

Interface Illusion by Conficker worm

Phishing Detection

- Unusual email/URL
  - Especially if similar to known URL/email
  - Email that elicits a strong emotional response and requests fast action on your part
- Attachments with uncommon names
- Typos, unusual wording
- No https (not a guarantee)

Phishing

- Phishing is an example of an interface illusion
- It looks like you're visiting Paypal's website, but you're really not.
  - If you type in your password, you've just given it to an attacker
- Advanced phishers can make websites that look every bit like the real thing
  - Even if you carefully check the address bar, or even the SSL certificate!

Man-in-the-middle attacks

- But not only is the man-in-the-middle able to see (and record) everything you're doing, and can capture passwords, but once you've authenticated to your bank (for example), the man-in-the-middle can hijack your session to insert malicious commands
  - Make a $700 payment to attacker@evil.com
- You won't even see it happen on your screen, and if the man-in-the-middle is clever enough, he can edit the results (bank balances, etc.) being displayed to you so that there's no visible record (to you) that the transaction occurred
  - Stealthy, like a rootkit

Man-in-the-middle attacks

- Keyboard logging, interface illusions, and phishing are examples of man-in-the-middle attacks
- The website/program/system you're communicating with isn't the one you think you're communicating with
- A man-in-the-middle intercepts the communication from the user, and then passes it on to the intended other party
  - That way, the user thinks nothing is wrong, because his password works, he sees his account balances, etc.

Covert channels

- An attacker creates a capability to transfer sensitive/unauthorized information through a channel that is not supposed to transmit that information.
- What information can/cannot be transmitted through a channel may be determined by a policy/guidelines/physical limitations, etc.

Outline

- Flaws, faults, and failures
- Unintentional security flaws
- Malicious code: Malware
- Other malicious code
- Nonmalicious flaws
- Controls against security flaws in programs

Side channels

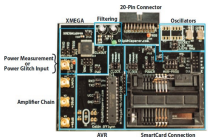
- What if Eve can't get Trojaned software on Alice's computer in the first place?
- It turns out there are some very powerful attacks called side channel attacks
  - Eve watches how Alice's computer behaves when processing the sensitive data
  - Eve usually has to be somewhere in the physical vicinity of Alice's computer to pull this off
    - But not always!

Covert channels

- Assume that Eve can arrange for malicious code to be running on Alice's machine
  - But Alice closely watches all Internet traffic from her computer
  - Better, she doesn't connect her computer to the Internet at all!
- Suppose Alice publishes a weekly report summarizing some (nonsensitive) statistics
- Eve can "hide" the sensitive data in that report!
  - Modifications to spacing, wording, or the statistics itself
  - This is called a covert channel

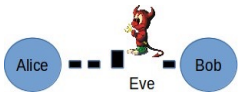
Potential Attack Vectors

- Timing computations
- Power consumption
- Electromagnetic emission
- Sound emissions
- Cache access
- Differential power analysis
- Differential fault analysis

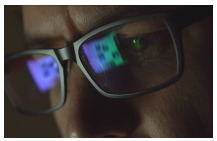


Potential Attack Vectors

- Bandwidth consumption



- 'Shoulder-surfing'
- Reflections

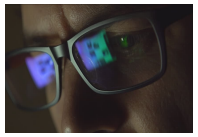
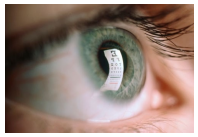


Reflections

- Eve uses a camera and a telescope
- Off-the-shelf: less than $2k
- Photograph reflection of screen through telescope
- Reconstruct original image
- Distance: 10–30 m
- Depends on equipment and type of reflecting surface

Reflections: Scenario

- Alice types her password on a device in a public place
- Alice hides her screen
- But there is a reflecting surface close

The picture so far

- We've looked at a large number of ways an attacker can compromise program security
  - Exploit unintentional flaws
  - Introduce malicious code, including malware
  - Exploit intentional, but nonmalicious, behaviour of the system
- The picture looks pretty bleak
- Our job is to control these threats
  - It's a tough job

Reflections: Defense