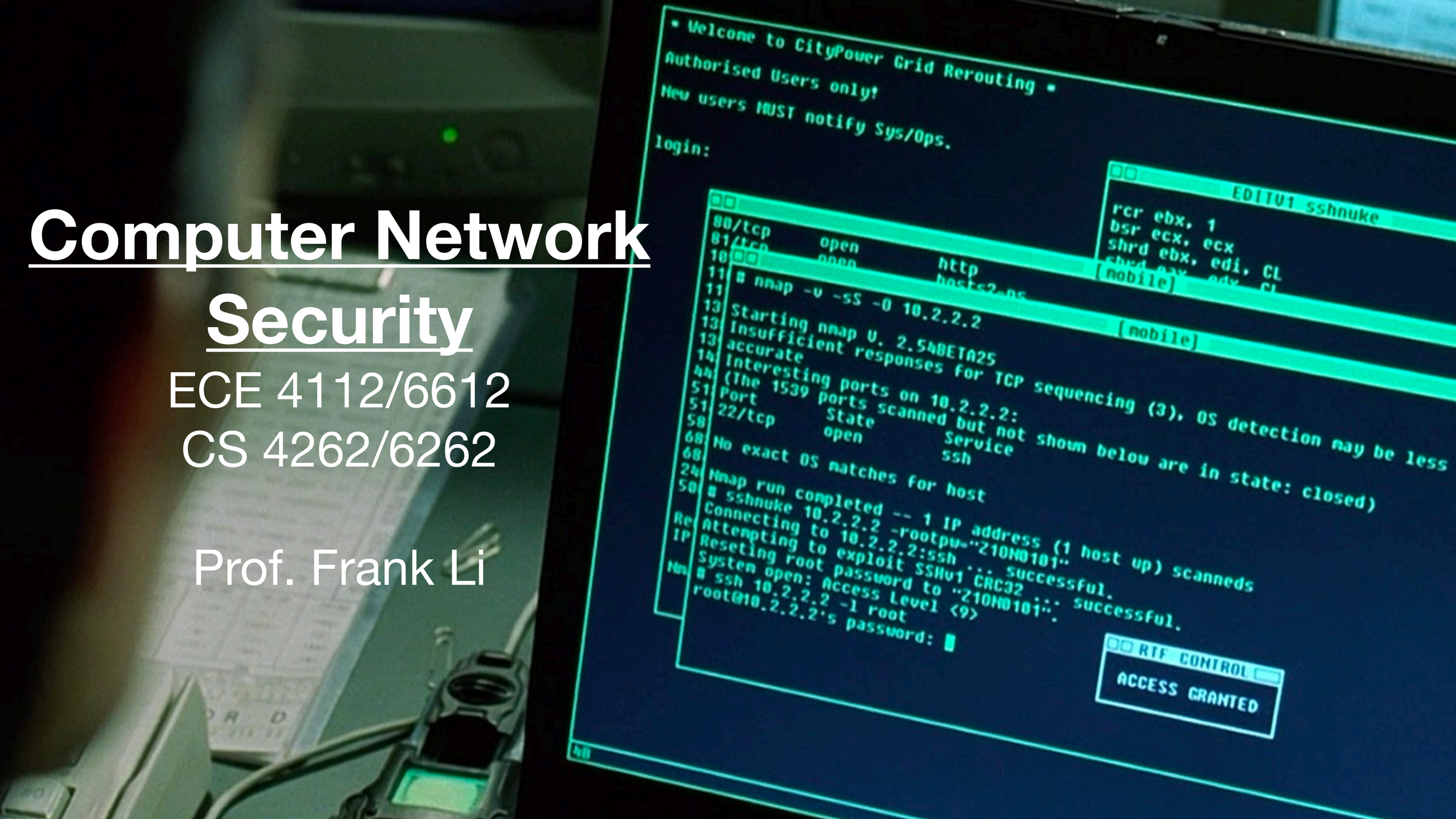
Computer Network Security

ECE 4112/6612
CS 4262/6262

Prof. Frank Li



Logistics

HW3 on web + email security due Monday Nov 13.

Quiz 2 grades released, regrades open until next Thurs, Nov 9.

Work on your projects!

Tue, Oct 10	**No Class (Fall Break)**
Thu, Oct 12	Web security Part 1: Web attacks and defenses
Tue, Oct 17	Web security Part 2: Web attacks and defenses
Thu, Oct 19	Web security Part 3: Web attacks and defenses
Tue, Oct 24*	**Quiz 2**
Thu, Oct 26*	Authentication
Tue, Oct 31	Email Security (Spam, Phishing)
Thu, Nov 2	Network Access Control
Tue, Nov 7	DoS attacks and defenses
Thu, Nov 9	Malware, Botnet
Tue, Nov 14	Last lecture: Censorship and Anonymous Communication
Thu, Nov 16	**Quiz 3**
Tue, Nov 21	**No Class (Early Thanksgiving Break)**
Thu, Nov 23	**No Class (Thanksgiving Break)**
Tue, Nov 28*	Project: Final Project Presentations
Thu, Nov 30*	Project: Final Project Presentations
Tue, Dec 5	Final Class: Final Project Presentations
Thu, Dec 7	**Final Exam: 2:40 - 5:30 PM (Undergraduate Sections Only)**

Wrapping up Email Security

Email Messages

- Emails consist of headers and then the message body (just like HTTP requests/responses)
- A few notable headers:
  - **From**: message sender, **set by the sender client**
  - **To**: message recipient
  - **Received**: each hop in the email delivery path (this header is added at each hop, so may show up multiple times). **Receiving mail server only knows the sending mail server (and must trust that the prior Received headers are correct)**
  - **Subject**: email subject line

Spoofing Defenses

- **Sender Policy Framework (SPF):** Allows a domain to indicate authorized senders (claiming to be sending from that domain)
- **DomainKeys Identified Mail (DKIM):** Allows a domain to publish a public key used to sign emails sent by authorized senders
- **Domain-based Message Authentication, Reporting and Conformance (DMARC):** Allows a domain to indicate what mail servers should do if SPF and/or DKIM checks fail (and who to report to)

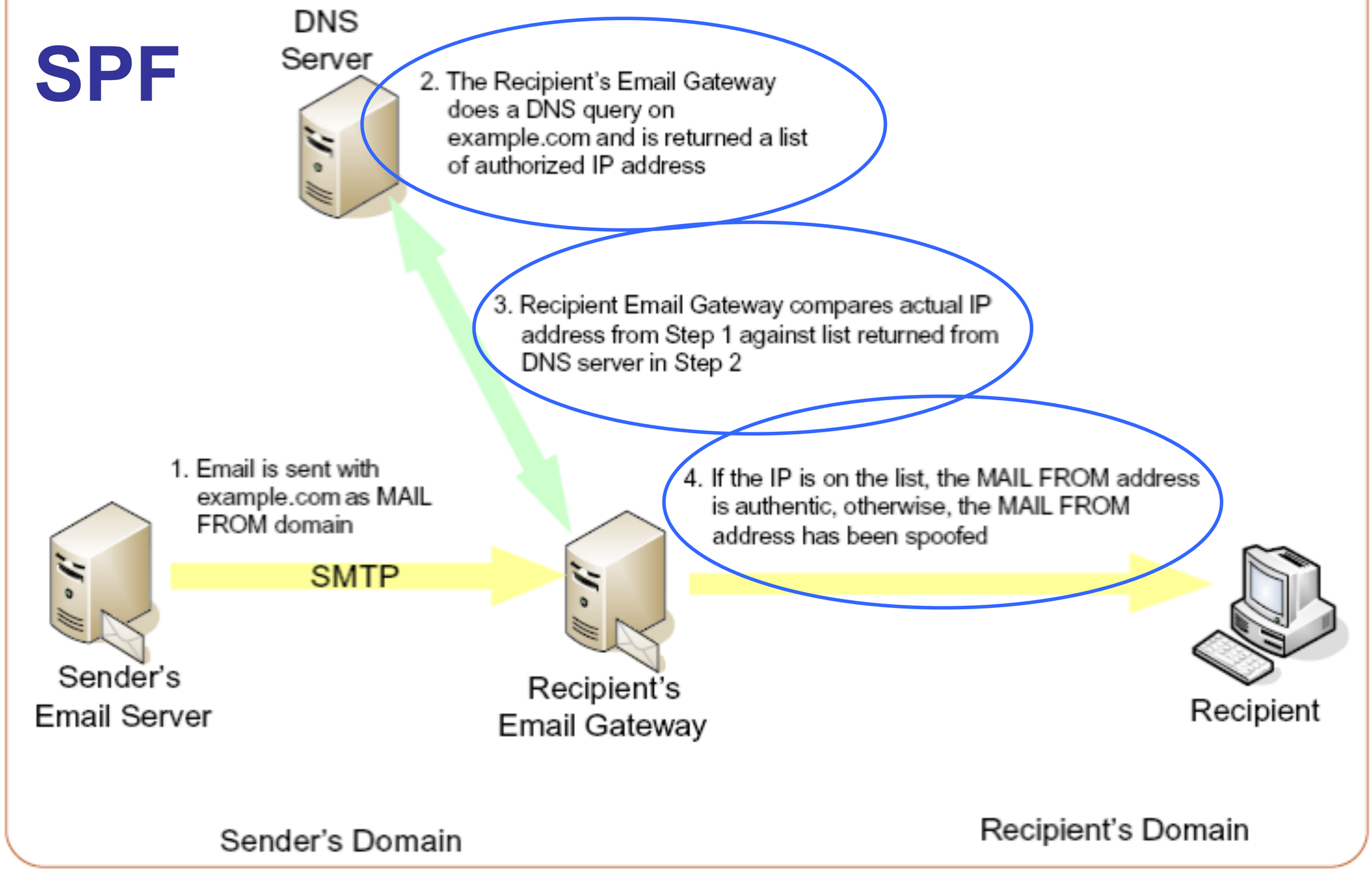
SPF

- A domain can publish a DNS TXT record indicating the mail servers authorized to send emails from that domain
  - Can list the authorized mail servers' IP addresses or domain names
  - Example: frank.com can list which IP addresses are allowed to send @frank.com emails.

- When receiving an email, a mail server can lookup the SPF DNS record for the sender's domain, and see if the email was received from an authorized sender. If not, could be spoofing.

- SPF records can also indicate a policy of what to do for unauthorized senders (e.g., allow, drop)

SPF


DNS Server
2. The Recipient's Email Gateway does a DNS query on example.com and is returned a list of authorized IP address
3. Recipient Email Gateway compares actual IP address from Step 1 against list returned from DNS server in Step 2
1. Email is sent with example.com as MAIL FROM domain
SMTP
4. If the IP is on the list, the MAIL FROM address is authentic, otherwise, the MAIL FROM address has been spoofed
Sender's Email Server
Recipient's Email Gateway
Recipient
Sender's Domain
Recipient's Domain

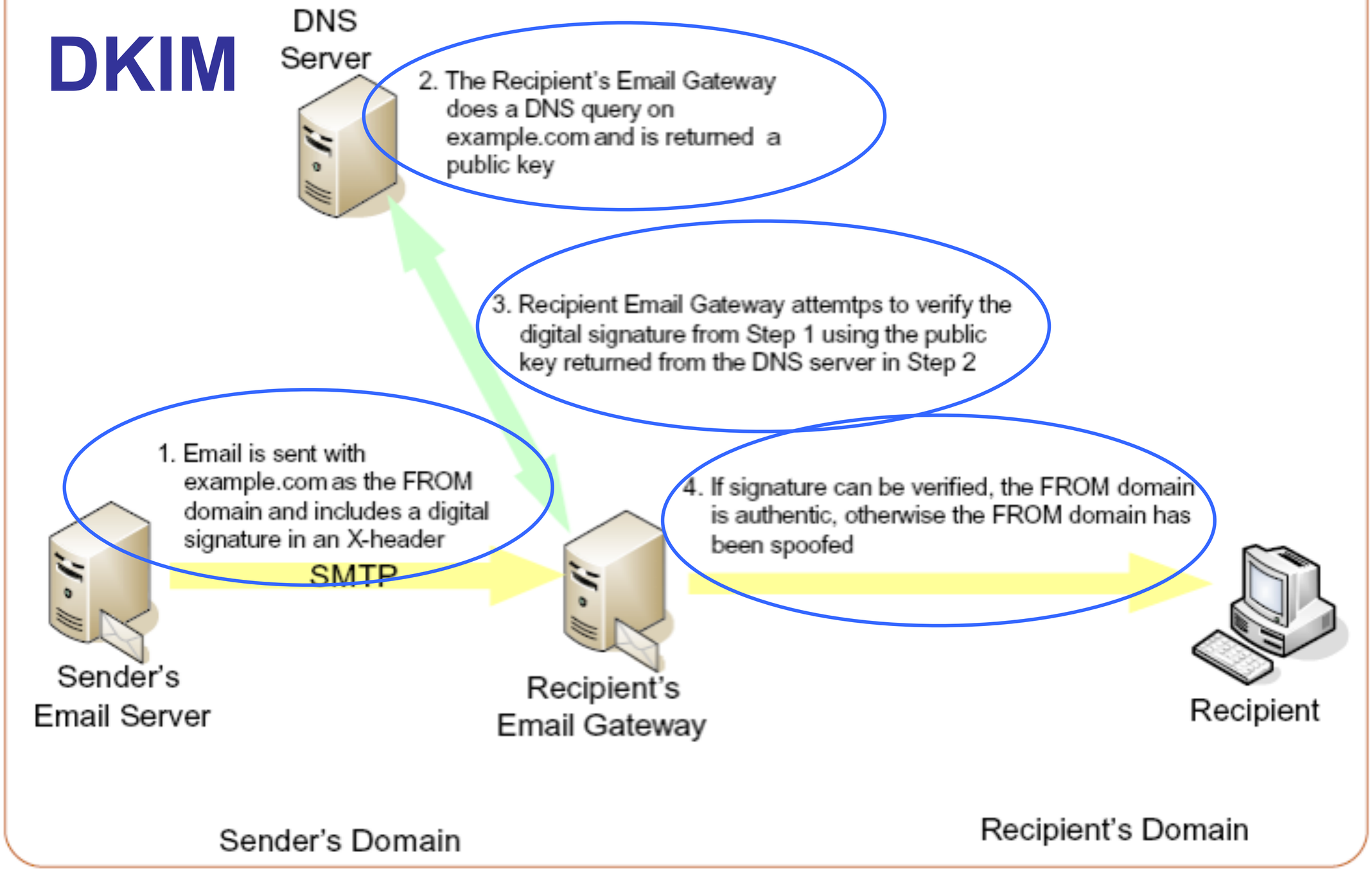
DKIM

- A domain can publish a DNS TXT record with a public key used to sign legitimate email from that domain.

- All legitimate email from that domain must include a signature created using the published key, put in the DKIM-Signature email header.
  - Authorized mail servers can generate this signature for every sent email.

- When receiving an email, a mail server can lookup the DKIM DNS record for the sender's domain, and see if the public key there can validate the email's DKIM signature. If not, email could be spoofed.

DKIM


DNS Server
2. The Recipient's Email Gateway does a DNS query on example.com and is returned a public key
3. Recipient Email Gateway attemtps to verify the digital signature from Step 1 using the public key returned from the DNS server in Step 2
1. Email is sent with example.com as the FROM domain and includes a digital signature in an X-header
SMTP
4. If signature can be verified, the FROM domain is authentic, otherwise the FROM domain has been spoofed
Sender's Email Server
Recipient's Email Gateway
Recipient
Sender's Domain
Recipient's Domain

DKIM-Signature Header (HW3)

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;
d=coe.gatech.edu; s=gt1; h=from:to:date:message-id:content-id:
content-transfer-encoding:mime-version:subject:reply-to:
sender:list-id:list-archive:list-help:list-owner: list-
post:list-subscribe:list-unsubscribe; bh=GuDd4TCd/
gkGXXG5B7j3aTtSIbh41BhM7KFbHa8Oi/A=; b=NFL8jSdKfYSkzdZBbEVZG+K/
qQH32v7pVb4tzRv7MbjKEAHbNNktAsof hzH/
MJOKvzISunXwxs855ChR+UCFLKQ2xb4B/aZXJvTr7FKGsmX78hwC5
JtmdnBHyKXOSzk7CYWHLpV4QpLdGh4L2piERO/xNyyEzIBr9VXFjHiP39
NLgrcPFSW+towhsJNpX8EcGX8LwO+pzUL+qQ2Ez/xGYn0DRTQOQFxMxzQ
5KTyQAz1dzGIL7r8QU3xZt5hmfFurO0ZetaHaNKpT5+5eBtUwKke8agmV
vhBqZXpHLH3zkUkvkMlXoWyIJnkjQAugAcajNOU8L6+0+W8fim9YZvO37 g==;
```

SPF vs DKIM

- Need to check both in practice, b/c some domains only deploy one and not the other (SPF is older, DKIM is newer).

- Some tradeoffs:
  - SPF requires being able to enumerate authorized senders.
  - DKIM requires modifying mail servers to generate DKIM signatures.
  - If email routing legitimately requires many hops, this can screw up SPF (b/c later hops don't directly communicate with/see the address of authorized senders). DKIM isn't affected.
  - SPF provides defense-in-depth though (more information on authorized senders). For example, DKIM doesn't prevent message replay attacks (although the threat seems(?) rather limited).

SPF and DKIM in practice

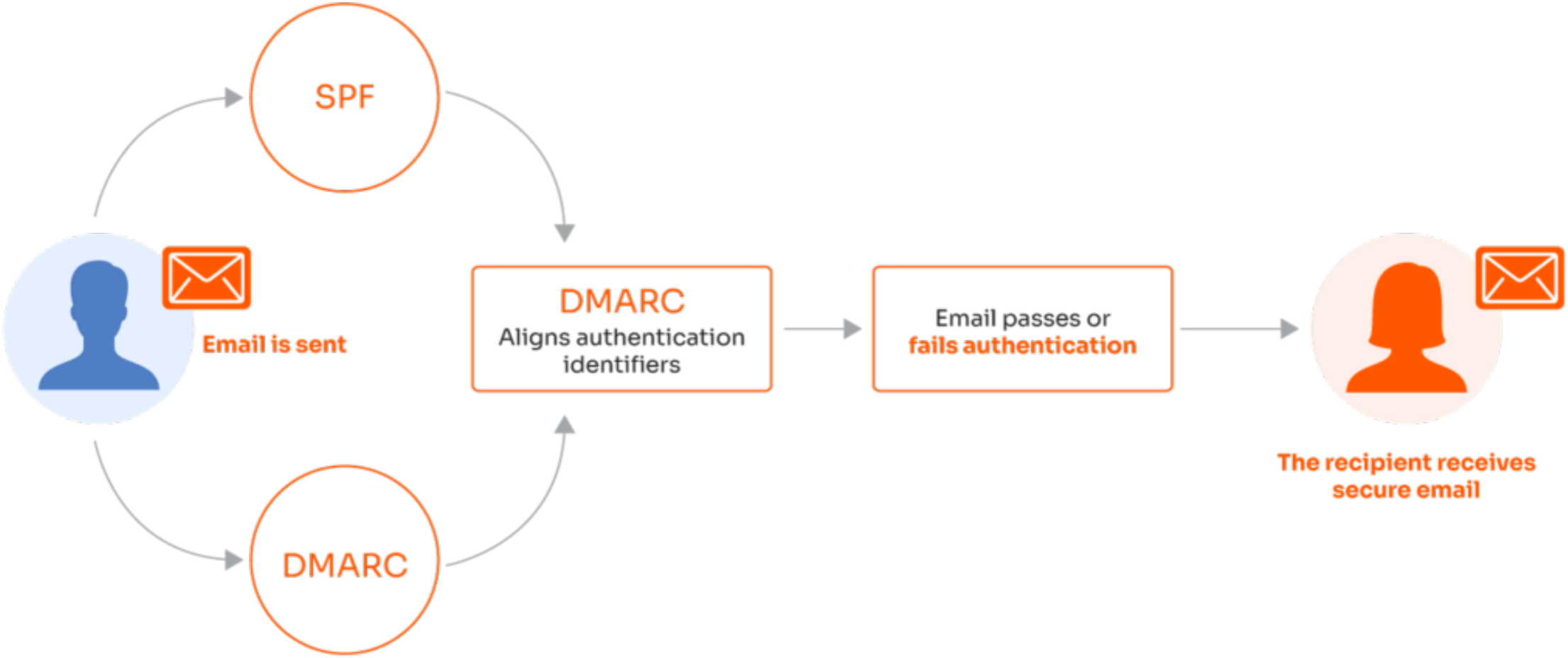
Authentication Method	Nov. 2013	Apr. 2015	Change
DKIM & SPF	74.66%	81.01%	+6.31%
DKIM only	2.25%	1.98%	−0.27%
SPF only	14.44%	11.41%	−2.99%
No authentication	8.65%	5.60%	−3.00%

Table 18: **Gmail Incoming Mail Authentication**—During April 2015, 94.40% of incoming Gmail messages were authenticated with DKIM, SPF, or both.

DMARC

- What should a mail server do if either SPF and/or DKIM fail for an email?
  - DMARC can specify a policy combining both.
  - A domain can publish a DNS TXT record indicating a policy for what to do with SPF/DKIM errors for emails from that domain.
    - Action: Reject, quarantine, accept the email
    - Reporting: Whether the error should be reported. If so, who to contact and what information to provide (automated process).
  - A receiving mail server that observes SPF/DKIM errors can consult the DMARC policy to determine how to respond, and who to report observed errors to.
    - Reporting helps the sending domain identify SPF/DKIM misconfigurations (or potential attacks spoofing that domain).

DMARC


SPF
Email is sent
DMARC
Aligns authentication identifiers
Email passes or
fails authentication
The recipient receives secure email
DMARC

DMARC

Published Policy	Gmail Messages	Top Million Domains
Quarantine	1.34%	709 (0.09%)
Empty	11.66%	6,461 (0.82%)
Reject	13.08%	1,720 (0.22%)
Not published	73.92%	783,851 (98.9%)

Table 22: **DMARC Policies**—We categorize DMARC policies for incoming Gmail messages from April 2015 and for Top Million domains with MX records on April 26, 2015.

- DMARC hasn't been as widely deployed yet, perhaps as DMARC came after SPF/DKIM and sites are hesitant to publish a meaningful policy that drops emails.
- Affects DKIM more than SPF (as SPF has a policy component)
- SPF/DKIM can still be used as signals/features even w/o DMARC

Other Email Security: Spam Filtering

- Many email providers deploy spam filters
  - Filter emails from blacklisted sources
  - Rate limit emails from a source
  - Use machine learning to detect likely spam emails
  - Cluster users/emails sending high volume of identical/similar emails

- Attackers can still often circumvent
  - Use botnets to avoid blacklists/rate limits
  - Change spam strategy to avoid ML detection/clustering

Other Email Security: Legal

- CAN-SPAM Act passed in 2003
  - Bans email harvesting, misleading header information, deceptive subject lines, use of proxies/spoofing
  - Requires opt-out and identification of ads
  - Imposes penalties

- FTC has brought a number of cases against violators.
  - Doesn't affect non-US spam though
  - Botnets and open relays/proxies make attribution hard

Other Email Security: Legal

FTC Announces First Can-Spam Act Cases

Two Operations Generated Nearly One Million Complaints to Agency

April 29, 2004

The FTC has cracked down on two spam operations that have clogged the Internet with millions of deceptive messages and violated federal laws. A complaint targeting Detroit-based Phoenix Avatar was developed in a joint investigation with the U.S. Attorney's Office in Detroit and the U.S. Postal Inspection Service. At the request of the FTC, a U.S. District Court judge has barred the illegal spamming and frozen the defendants' assets. Federal

FTC lawsuit reminds businesses: CAN-SPAM means CAN'T spam

By: Seena Gressin, Attorney, Division of Consumer and Business Education, FTC | August 14, 2023

Recent News

GMAIL

New Gmail protections for a safer, less spammy inbox

Oct 03, 2023
2 min read

Starting in 2024, we'll require bulk senders to authenticate their emails, allow for easy unsub under a reported spam threshold.

Neil Kumaran
Group Product Manager, Gmail Security & Trust

New requirements for bulk senders

By February 2024, Gmail will start to require that bulk senders:

1. **Authenticate their email:** You shouldn't need to worry about the intricacies of email security standards, but you should be able to confidently rely on an email's source. So we're requiring those who send significant volumes to strongly authenticate their emails following well-established best practices. Ultimately, this will close loopholes exploited by attackers that threaten everyone who uses email.

2. **Enable easy unsubscription:** You shouldn't have to jump through hoops to stop receiving unwanted messages from a particular email sender. It should take one click. So we're requiring that large senders give Gmail recipients the ability to unsubscribe from commercial email in one click, and that they process unsubscription requests within two days. We've built these requirements on open standards so that once senders implement them, everyone who uses email benefits.

3. **Ensure they're sending wanted email:** Nobody likes spam, and Gmail already includes many tools that keep unwanted messages out of your inbox. To add yet another protection, moving forward, we'll enforce a clear spam rate threshold that senders must stay under to ensure Gmail recipients aren't bombarded with unwanted messages. This is an industry first, and as a result, you should see even less spam in your inbox.

Email Security Summary

- Key email protocols: SMTP, IMAP, POP3
  - Hop-to-hop secure communication using TLS
  - End-to-end secure communication using PGP or S/MIME

- Email spoofing is a major problem
  - Leads to spam/phishing
  - Can use SPF, DKIM, and DMARC to defend

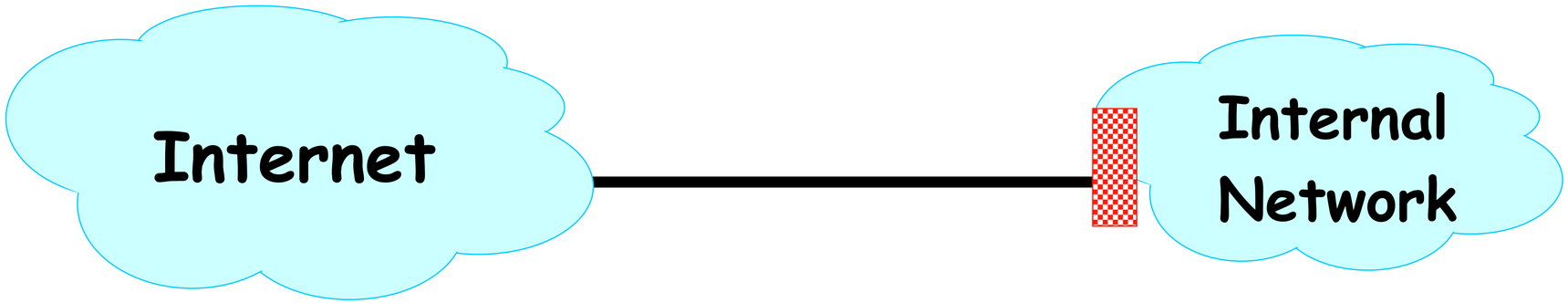
Controlling Networks Using *Firewalls*

Controlling Networks … On The Cheap

- Motivation: How do you harden a set of systems against external attack?
  - *Key Observation:*
    - *The more network services your machines run, the greater the risk*
  - Due to larger attack surface
- One approach: on each system, turn off unnecessary network services
  - But you have to know ***all*** the services that are running
  - And sometimes some trusted remote users still require access
- Plus key question of scaling
  - What happens when you have to secure 100s/1000s of systems?
  - Which may have different OSs, hardware & users …
  - Which may in fact not all even be identified …

Taming Management Complexity

- Possibly more scalable defense: Reduce risk by blocking *in the network* outsiders from having unwanted access your network services
  - Interpose a **firewall** that traffic to/from the outside must traverse
  - Chokepoint can cover 1000s of hosts



Selecting a Security Policy

- What policy should be implemented:
  - *Who is allowed to talk to whom, accessing what service?*
- Distinguish between inbound & outbound connections
  - Inbound: attempts by external users to connect to services on internal machines
  - Outbound: internal users to external services
  - Why? Because fits with a common ***threat model***

Selecting a Security Policy

- What policy should be implemented:
  - *Who is allowed to talk to whom, accessing what service?*
- Distinguish between inbound & outbound connections
  - Inbound: attempts by external users to connect to services on internal machines
  - Outbound: internal users to external services
  - Why? Because fits with a common ***threat model***
- Conceptually simple *access control policy*:
  - Permit inside users to connect to any service (trusted insider)
  - External users restricted:
    - Permit connections to services meant to be externally visible
    - Deny connections to services not meant for external access

How To Treat Traffic Not Mentioned in Policy?

- **Default Allow**: start off permitting external access to services
  - Shut them off as problems recognized

How To Treat Traffic Not Mentioned in Policy?

- **Default Allow**: start off permitting external access to services
  - Shut them off as problems recognized
- **Default Deny**: start off permitting just a few known, well-secured services
  - Add more when users complain (and mgt. approves)

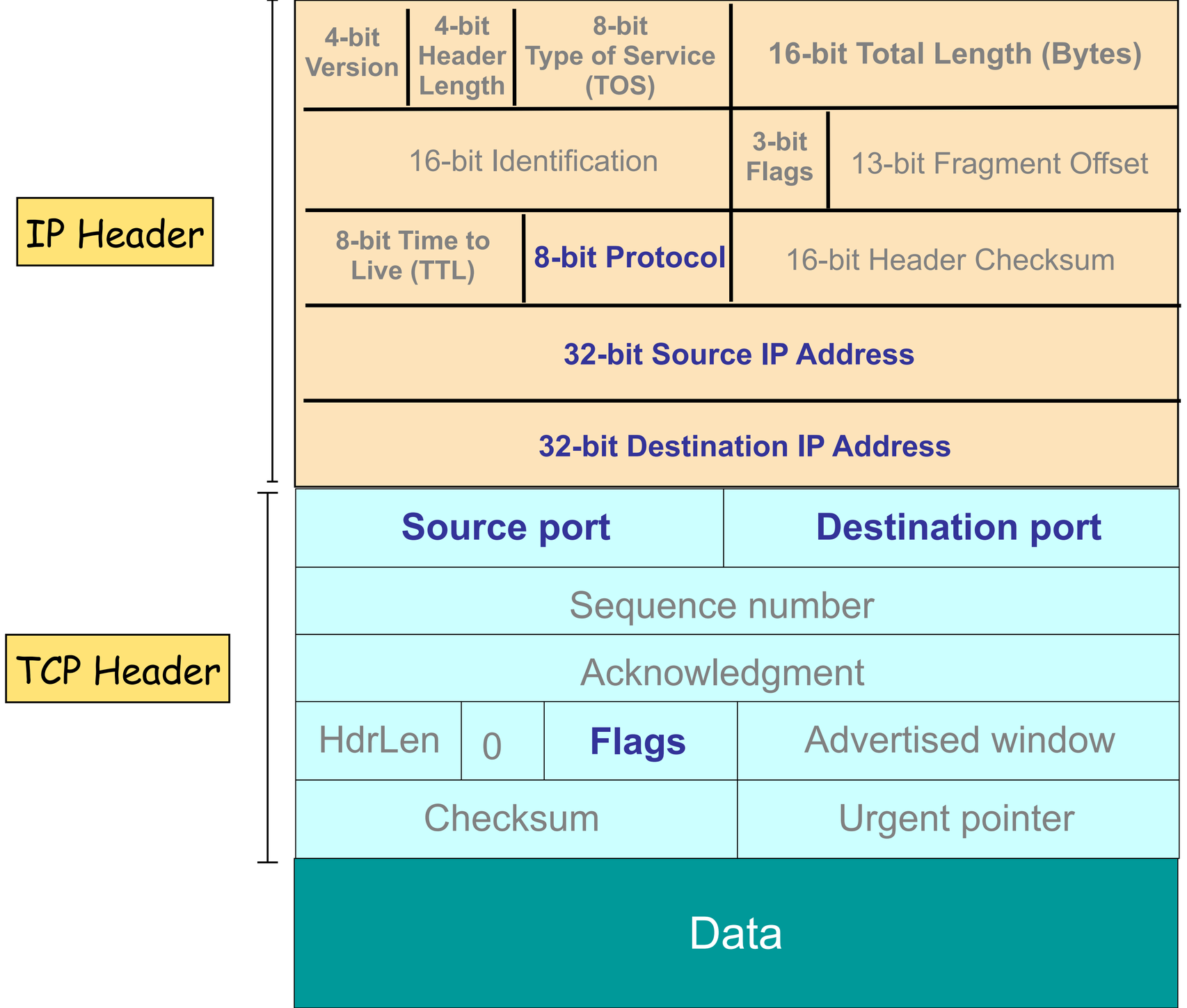
How To Treat Traffic Not Mentioned in Policy?

- **Default Allow**: start off permitting external access to services
  - Shut them off as problems recognized
- **Default Deny**: ✓ off permitting just a few known, well-secured services
  - Add more when users complain (and mgt. approves)
- Pros & Cons?
  - Flexibility vs. conservative design
  - Flaws in Default Deny get noticed more quickly / less painfully

In general, use Default Deny

Packet Filters

- Most basic kind of firewall is a *packet filter*
  - Router with list of *access control rules*
  - Router checks each received packet against security rules to decide to forward or drop it
  - Each rule specifies which packets it applies to based on a packet's header fields (stateless)
    - Specify source and destination IP addresses, port numbers, and protocol names, or wild cards


IP Header
4-bit Version
4-bit Header Length
8-bit Type of Service (TOS)
16-bit Total Length (Bytes)
16-bit Identification
3-bit Flags
13-bit Fragment Offset
8-bit Time to Live (TTL)
8-bit Protocol
16-bit Header Checksum
32-bit Source IP Address
32-bit Destination IP Address
TCP Header
Source port
Destination port
Sequence number
Acknowledgment
HdrLen
0
Flags
Advertised window
Checksum
Urgent pointer
Data

Packet Filters

- Most basic kind of firewall is a *packet filter*
  - Router with list of *access control rules*
  - Router checks each received packet against security rules to decide to forward or drop it
  - Each rule specifies which packets it applies to based on a packet's header fields (stateless)
    - Specify source and destination IP addresses, port numbers, and protocol names, or wild cards
    - Each rule specifies the *action* for matching packets: **ALLOW** or **DROP** (aka DENY)

    `<ACTION> <PROTO> <SRC:PORT> -> <DST:PORT>`
  - First listed rule has *precedence*

Examples of Packet Filter Rules

```
allow tcp 4.5.5.4:1025 -> 3.1.1.2:80
```

- States that the firewall should **permit** any TCP packet that's:
  - from Internet address `4.5.5.4` **and**
  - using a source port of `1025` **and**
  - destined to port `80` of Internet address `3.1.1.2`

```
deny tcp 4.5.5.4:* -> 3.1.1.2:80
```

- States that the firewall should **drop** any TCP packet like the above, regardless of source port

Examples of Packet Filter Rules

```
deny tcp 4.5.5.4:* -> 3.1.1.2:80
allow tcp 4.5.5.4:1025 -> 3.1.1.2:80
```

- *In this order*, the rules won't allow *any* TCP packets from `4.5.5.4` to port `80` of `3.1.1.2`

```
allow tcp 4.5.5.4:1025 -> 3.1.1.2:80
deny tcp 4.5.5.4:* -> 3.1.1.2:80
```

- *In this order*, the rules allow TCP packets from `4.5.5.4` to port `80` of `3.1.1.2` ***only*** if they come from source port `1025`

Firewall Considerations

- Firewalls can have 1000s of filtering rules like these
  - Easy to introduce subtle errors 😕
- Provide not only security but also *policy enforcement*
  - E.g. do not allow company systems to access file-sharing sites

- Modern firewalls operate in a stateful fashion
  - Make Yes/No decisions upon establishment of a connection/ flow
    - For Yes decisions, add 4-tuple to a *connection table* consulted for future traffic
    - Drop arriving non-establishment packet if not in table
- An important example of a *reference monitor*

Security Principle: *Reference Monitors*

- A *reference monitor* examines every request to access a controlled resource (an *object*) and determines whether to allow request

Reference Monitor Security Properties

- *Always invoked*
  - *Complete mediation* property: all security-relevant operations must be mediated by RM
  - RM should be invoked on every operation controlled by access control policy
- *Tamper-resistant*
  - Maintain RM integrity (no code/state tampering)
- *Verifiable*
  - Can verify RM operation (correctly enforces desired access control policy)
    - Requires extremely **simple** RM
    - We find we can't verify correctness for systems with any appreciable degree of complexity

Considering Firewalls as Reference Monitors

- Always invoked?
  - Place firewall as an *in-path* element on all chokepoint links for all internal-external communications
    - Set of links known as the ***security perimeter***
  - Packets only forwarded across link if firewall explicitly decides to do so after inspection

Potential Problems?

- What if user brings an infected device onto the premises?
  - Attacker has foothold within the internal network
  - Will firewall help?
    - Not with attacker targeting other internal systems. But might be able to prevent attacker from communicating with external world (e.g., exfiltrating data).

RM Property: *Tamper-Resistant*

- Will this hold?

- Do not allow management access to firewall other than from specific hosts
  - I.e., firewall itself needs firewalling
- Protect firewall's physical security
- Must also secure storage & propagation of configuration data

RM Property: *Verifiable*

- Will this hold?
- Current practice:
  - Packet filter software too complex for feasible systematic verification …
  - … and rulesets with 1,000s (!) of rules
- Result:
  - *Bugs* that allowed attackers to defeat intended security policy by sending unexpected packets that packet filter doesn't handle as desired
- In addition: challenging to ensure network topology does not allow internal access by untrusted devices

Why Have Firewalls Been Successful?

- *Central control – easy administration and update*
  - Single point of control: update one config to change security policies
  - Potentially allows rapid response
- *Easy to deploy – transparent to end users*
  - Easy incremental/total deployment to protect 1,000's
- *Addresses an important problem*
  - Security vulnerabilities in network services are rampant
  - Easier to use firewall than to directly secure code …

Firewall Disadvantages?

- *Functionality loss* – *less connectivity, less risk*
  - May reduce network's usefulness
  - Some applications don't work with firewalls
    - Two peer-to-peer users behind different firewalls
- *The malicious insider problem*
  - Deployment assumes insiders are trusted
    - Malicious insider (or *anyone gaining control of internal machine*) can wreak havoc
- Firewalls establish a *security perimeter*
  - Threat from travelers with laptops, cell phones, …

Getting Around Firewalls

Subverting Firewalls

- Along with possible bugs, packet filters have a fundamentally limited semantic model
  - May only handle certain network layers or applications

- How can a local user who wants to get around their site's firewall exploit this? (**Note**: we're not talking about how an external attacker can escape a firewall's restrictions)

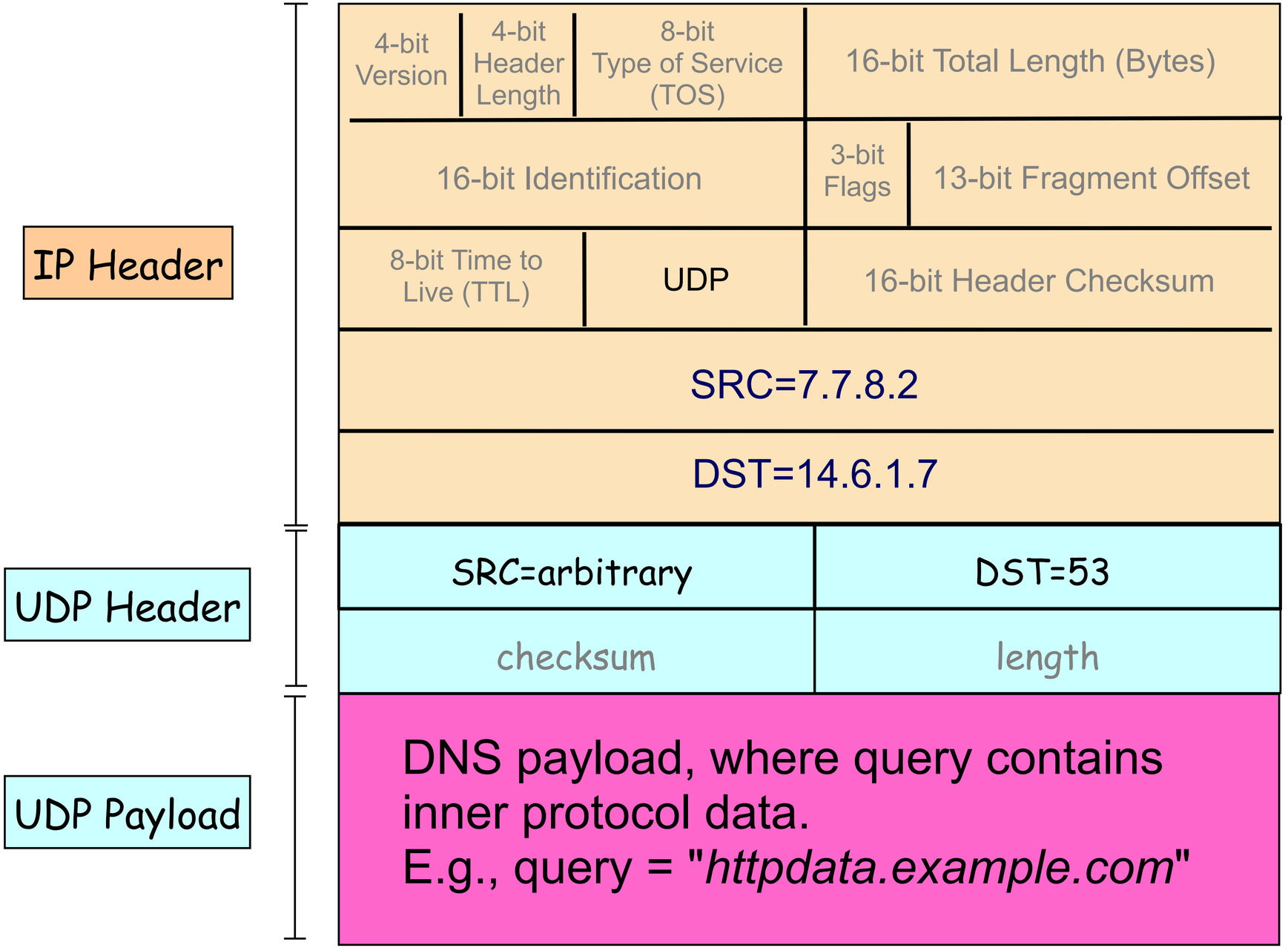
Hiding on Other Ports

- Method #1: use port allocated to another service
  - Who says that e.g. port 53/udp = DNS?
    - Why couldn't it be say Skype or BitTorrent? Just requires that client & server agree on application protocol

- Method #2: **tunneling**
  - Encapsulate one protocol inside another
  - Receiver of "outer" protocol *decapsulates* interior tunneled protocol to recover it
  - Pretty much **any** protocol can be tunneled over another (with enough effort)

- Detecting these methods?
  - **Deep Packet Inspection (DPI):** process packet data to check for correct protocols and apply policies.

Tunnel HTTP GET Request through DNS

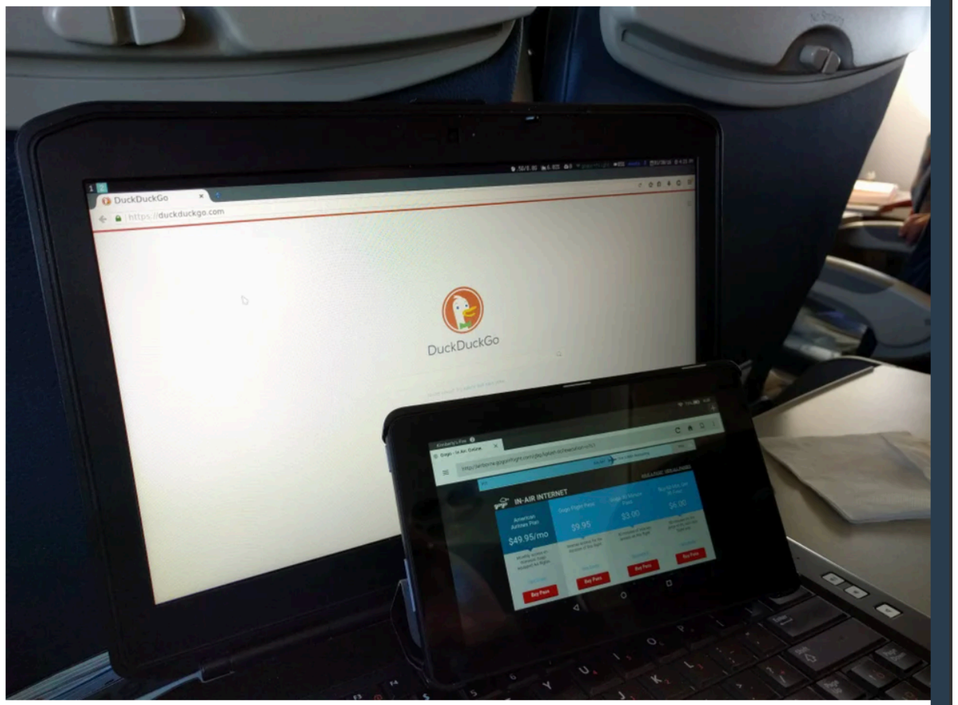


HTTP over DNS tunnel in practice!

- Iodine: Tunnel IPv4 traffic through DNS

All These Other Plebs Are Paying for in-flight Wi-Fi: Linux with Iodine DNS Tunneling

i.imgur.com/3vYC6V...



74 Comments Share Save Hide Report 98% Upvoted

iodine

The latest code is on github

Latest relase (from 2014-06-16): **0.7.0**
Download source / binaries: win32/64, android
Older downloads available below.

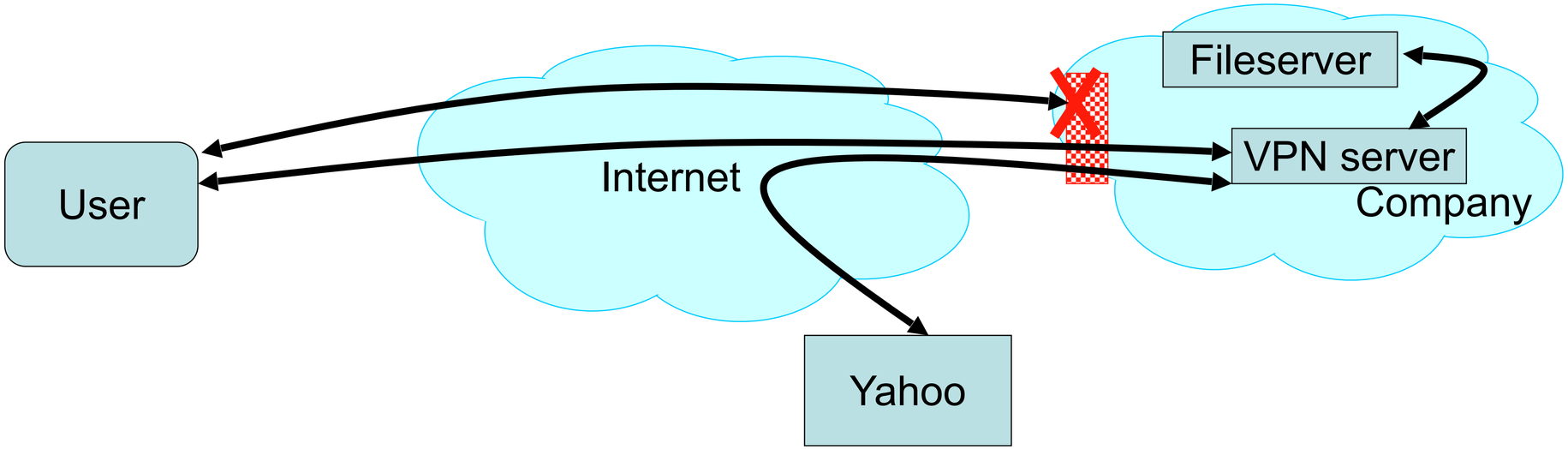
iodine lets you tunnel IPv4 data through a DNS server. This can be usable in different situations where internet access is firewalled, but DNS queries are allowed.

It runs on Linux, Mac OS X, FreeBSD, NetBSD, OpenBSD and Windows and needs a TUN/TAP device. The bandwidth is asymmetrical with limited upstream and up to 1 Mbit/s downstream.

Network Control & Tunneling

- *Tunneling* = embedding one protocol inside another
  - Sender and receiver at each side of the tunnel both cooperate (so it's not useful for initial attacks)
- Traffic takes on properties of outer protocol
  - Including for firewall inspection, which generally can't analyze inner protocol (due to complexity, unless implementing DPI)
- Tunneling has legitimate uses
  - E.g., Virtual Private Networks (VPNs)
    - Remote client establishes a tunnel to a VPN server, which relays/ forwards the client's packets
    - Makes remote client look like it's local to the VPN server's network
    - Tunnel encrypts traffic for privacy & to prevent meddling

Secure External Access to Inside Machines

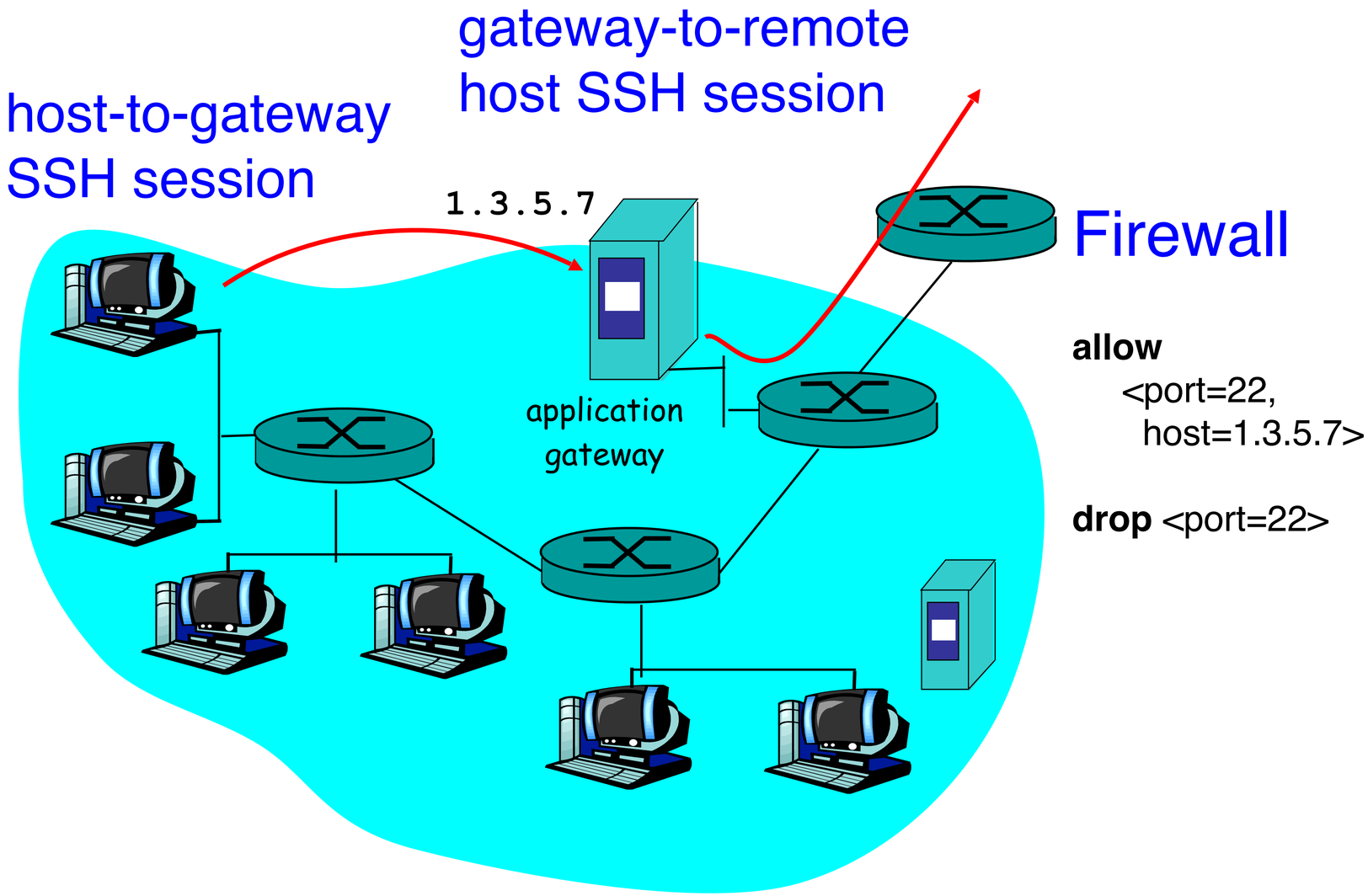


- Often need to provide secure remote access to a network protected by a firewall
  - Remote access, telecommuting, branch offices, …
- Create secure channel (*Virtual Private Network*, or **VPN**) to tunnel traffic from outside host/network to inside network
  - Provides Authentication, Confidentiality, Integrity
    - Requires some form of key management to set up
  - However, also raises *perimeter issues*

Application Proxies

- Can more directly control applications by requiring them to go through a proxy for external access
  - Proxy doesn't simply forward, but acts as an application-level middleman
- Example: SSH gateway
  - Require all SSH in/out of site to go through gateway
  - Gateway logs authentication, inspects decrypted text
  - Site's firewall configured to *prohibit any other* SSH access

SSH Gateway Example



Application Proxies

- Can more directly control applications by requiring them to go through a proxy for external access
  - Proxy doesn't simply forward, but acts as an application-level middleman
- Example: SSH gateway
  - Require all SSH in/out of site to go through gateway
  - Gateway logs authentication, inspects decrypted text
  - Site's firewall configured to prohibit any other SSH access
- Provides a powerful degree of monitoring/control
- Costs?
  - Need to run extra server(s) per app (possible *bottleneck*)
  - Each server requires careful hardening

Other Forms of Access Control

End hosts/applications also implement access control, separate from network-level access control.

- User authentication: passwords, hardware tokens, biometrics
- Access control policies:
  - User-based: List which actions each user can do
  - Role-based: Users have roles, and policies are defined per role
  - Capabilities: Whichever user possesses a special token (called a capability) can perform the action
  - Access control lists (ACL): For each data object, list who has access (and what type of access)