Practical Network Defense

Master's degree in Cybersecurity 2024-25

Proxies

Angelo Spognardi
spognardi@di.uniroma1.it
Dipartimento di Informatica
Sapienza Università di Roma

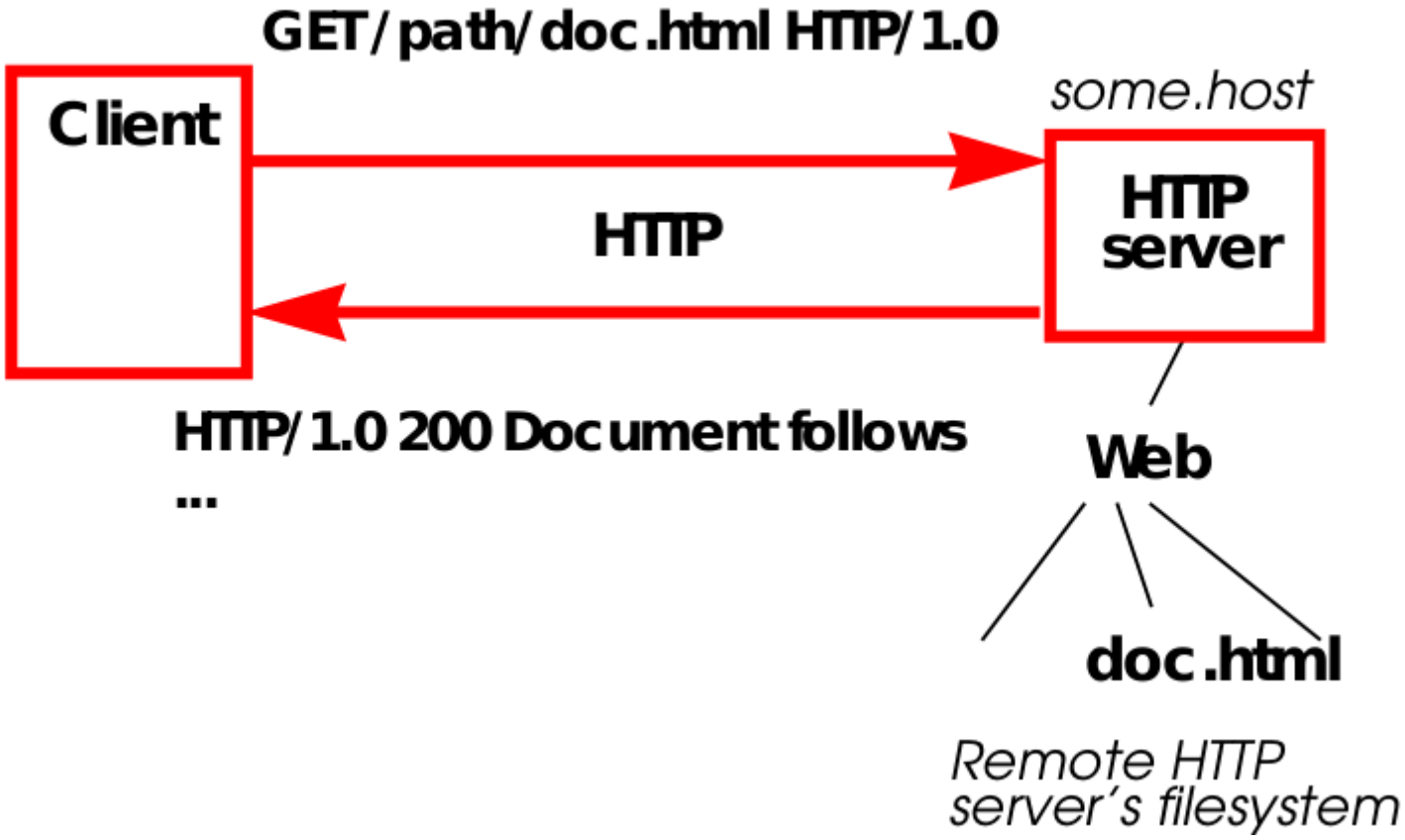
Proxies!

- Forward proxy
- Reverse proxy
- Application proxy
- Transparent proxy

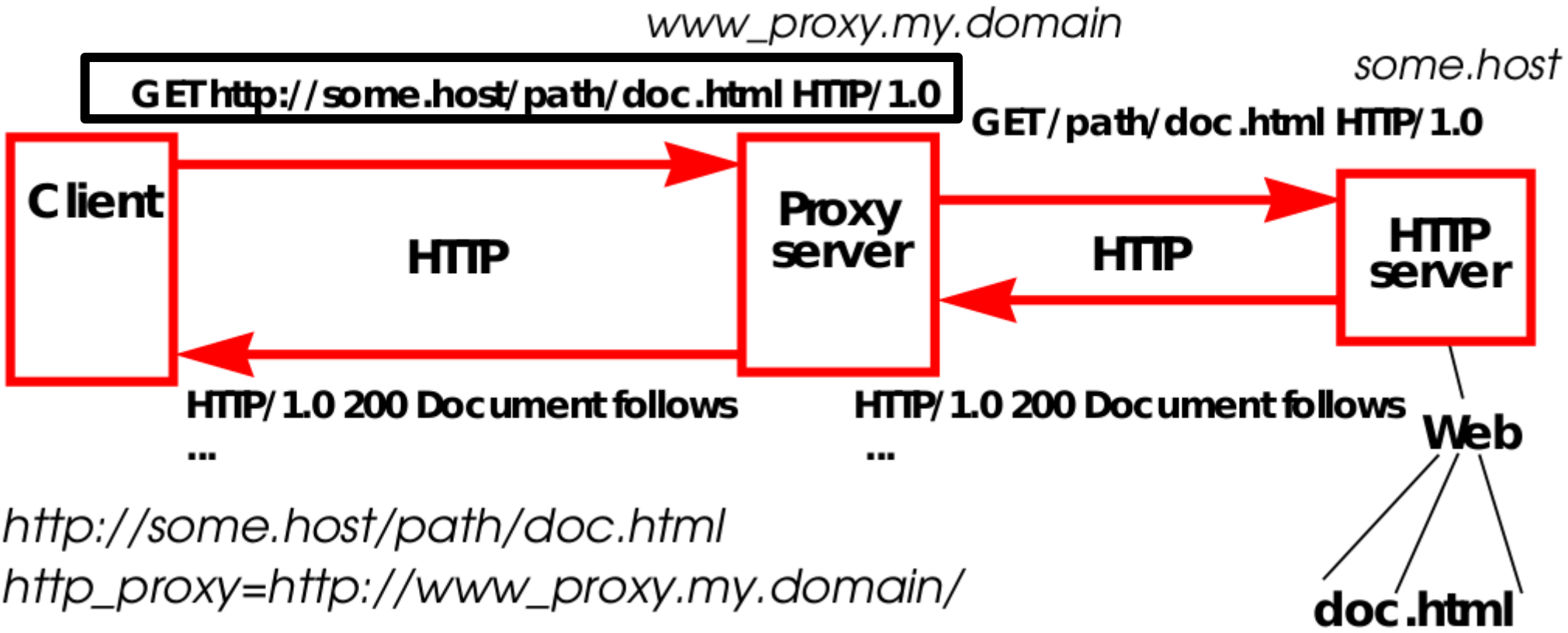
Proxy history: forward proxy

- A WWW proxy server provides access to the Web for people on closed subnets who can only access the Internet through a firewall machine
  - Ari Luotonen, CERN Kevin Altis, Intel, April 1994
- Original idea: *An application-level proxy makes a firewall safely permeable for users in an organization, without creating a potential security hole through which "bad guys" can get into the organizations' net.*
  - Namely: one single host handling requests from several users.

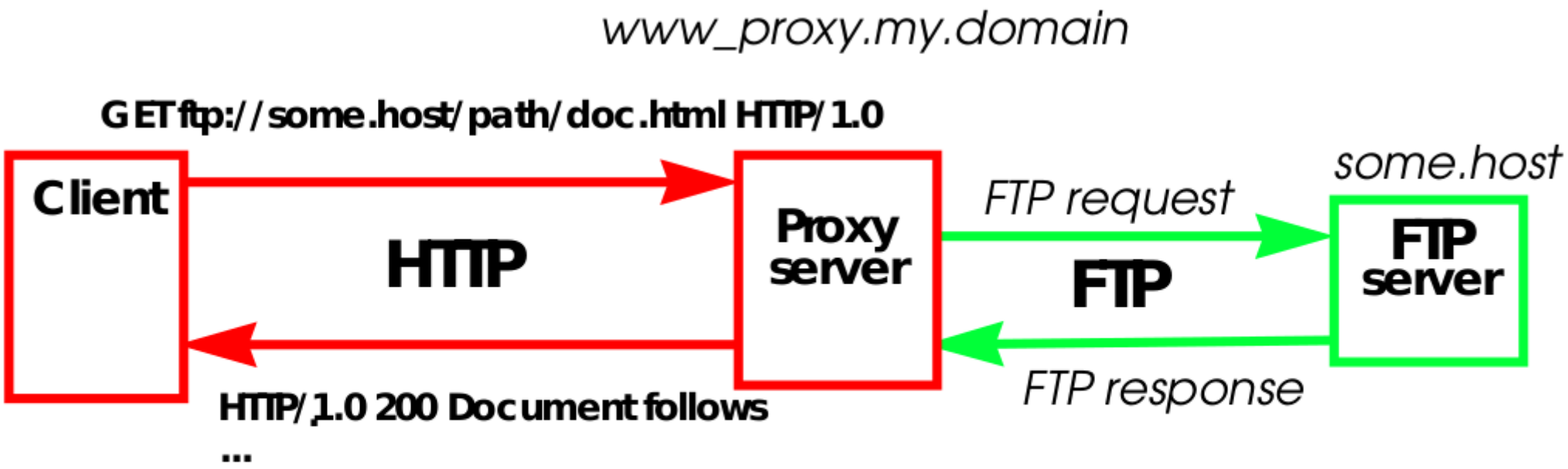
Normal HTTP transaction



Proxied HTTP transaction



Proxied FTP transaction



ftp://some.host/path/doc.html
ftp_proxy=http://www_proxy.my.domain/

Other benefits of forward proxy

- Authentication, Authorization, Auditing, whitelisting, blacklisting…
- Caching
  - store the retrieved document into a local file for further use so it won't be necessary to connect to the remote server the next time that document is requested
  - Problems:
    - How long is it possible to keep a document in the cache and still be sure that it is up-to-date?
    - How to decide which documents are worth caching and for how long?
  - Solutions:
    - HEAD http request (very inefficient)
    - `If-Modified-Since` request header

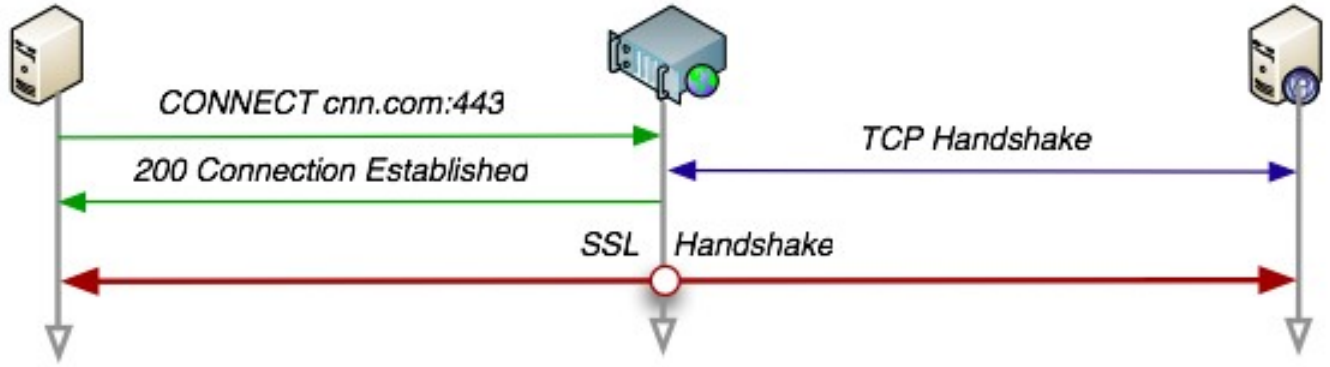
Forward proxy

- HTTP requests
  - Standard request in absolute-form to the proxy
  - The proxy will be the middle-point, forwarding the request towards the final termination
- Other (non-HTTP requests)
  - HTTP tunneling
  - HTTP CONNECT request with absolute-form to the proxy
  - The proxy establishes the TCP connection and becomes the middle-point

HTTP tunneling: HTTP CONNECT

- Allow the use of any protocol that uses TCP
  - https://tools.ietf.org/html/draft-luotonen-web-proxy-tunneling-01
- Idea: the proxy simply receives the destination host the client wants to connect to and establishes the connection on its behalf
- Then, when the connection is established, the proxy server continues to proxy the TCP stream unmodified to and from the client
- Clearly, the proxy can perform authentication, whitelisting, and so on before accepting to forward the stream of data

HTTP CONNECT method



- https://tools.ietf.org/html/rfc7231#section-4.3.6
- Anything that uses a two-way TCP connection can be passed through a CONNECT tunnel
  - Example: HTTP forwarding SSL/TLS
- Not all proxy servers support the CONNECT method or limit it to port 443 only

Content-filtering proxy

- After user authentication, HTTP proxy controls over the content that may be relayed
  - In schools, no Facebook or porn websites
    - Blacklists or semantic searches
  - Virus, malware scan
  - No files with executable or watermarking and so on

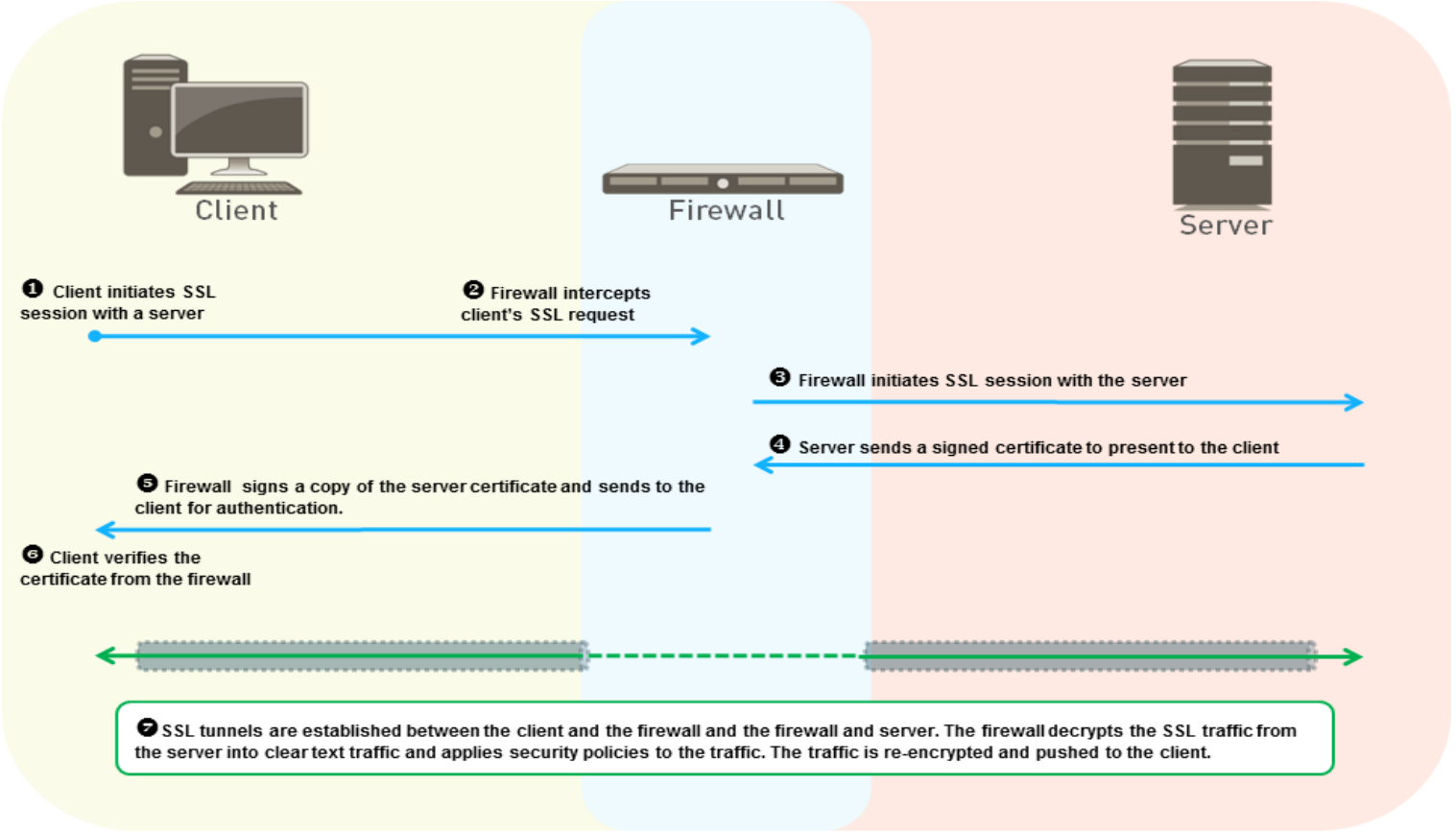
Anonymizer proxy

- A proxy server that acts as an intermediary and privacy shield between a client computer and the rest of the Internet
- It accesses the Internet on the user's behalf, protecting personal information by hiding the client computer identifying information (IP address, firstly)
- The server sees requests coming from the proxy address rather than the actual client IP address
  - Typical use for accessing restricted content (like *The pirate bay* and similar)
    - http://spys.one/en/
    - https://free-proxy-list.net/

SSL Forward proxy

- A way to decrypt and inspect SSL/TLS traffic from internal users to the web, generally implemented in firewalls
- SSL Forward Proxy decryption prevents malware concealed as SSL encrypted traffic from being introduced in the network
- How it works:
  - The proxy uses certificates to establish itself as a trusted third party to the session between the client and the server
  - As the proxy continues to receive SSL traffic from the server that is destined for the client, it decrypts the SSL traffic into clear text traffic and applies decryption and security profiles to the traffic
  - The proxy, then, re-encrypts and forwards the traffic to the client
    - What about trust?

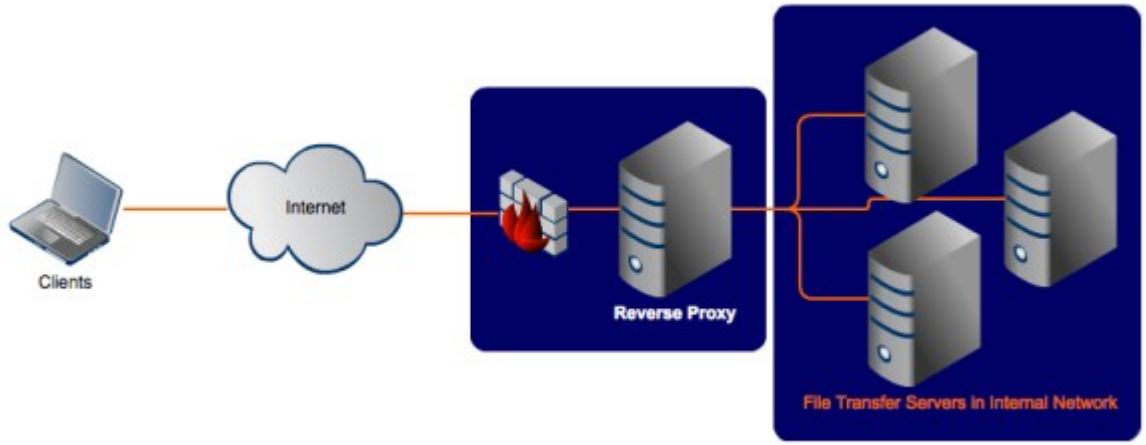
SSL Forward proxy with a figure



https://docs.paloaltonetworks.com/pan-os/9-1/pan-os-admin/decryption/decryption-concepts/ssl-forward-proxy

Reverse proxy

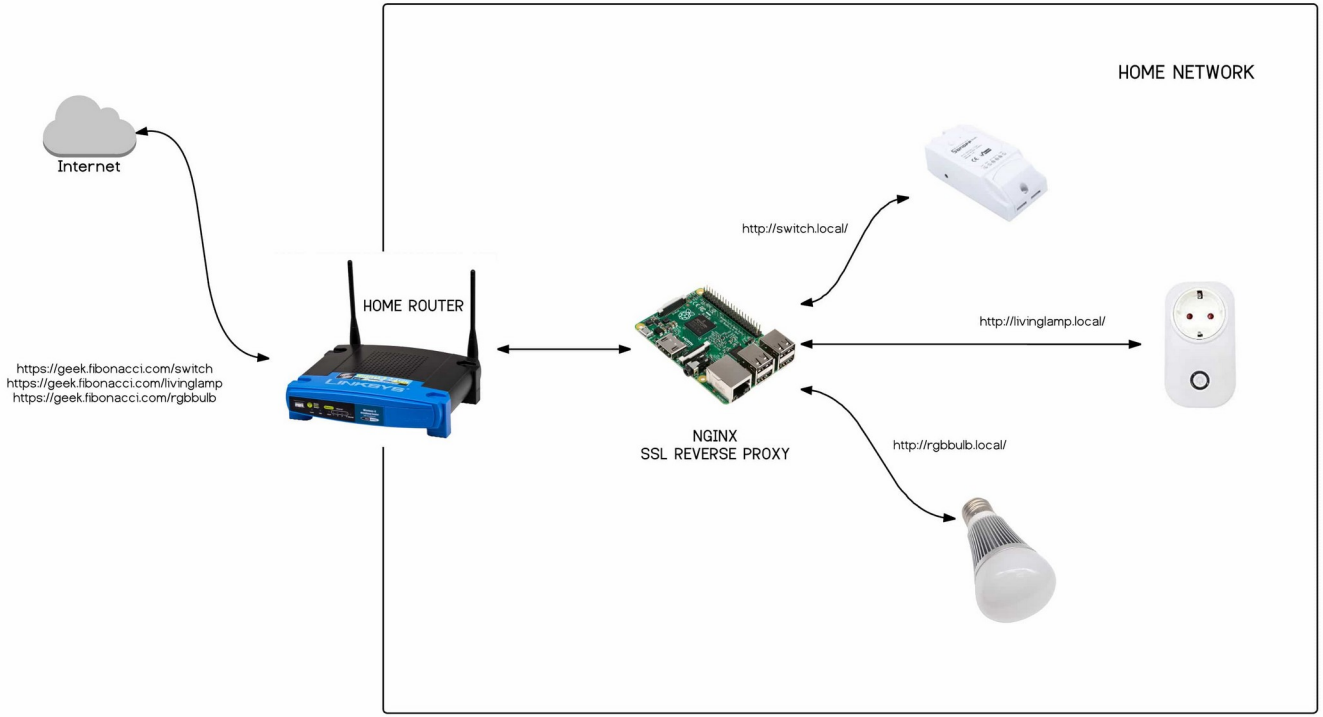
- Forward proxy operates on behalf of the client
- Reverse proxy operates on behalf of the server
- It receives the requests from the outside as if it were the server and then forwards the request to the actual destination (origin) server
- Typical functions:
  - Load balancing
  - Cache static content
  - Compression
  - Accessing several servers into the same URL space
  - Securing of the internal servers
  - Application level controls
  - TLS acceleration



Internal server protection

- The reverse proxy receives the requests from the clients and then issues new, prim and proper requests to the real server
- No direct connection with the outside also means defense against DoS
- Can provide support for HTTPS to servers that only have HTTP
- Can add AAA to services that do not have them
  - Example: a IoT device behind a firewall that must be accessible from the Internet

Reverse proxy for IoT access



https://tinkerman.cat/post/secure-remote-access-to-your-iot-devices

However: don't forget client authentication!!

Reverse proxy for application control: application firewall

- Application layer firewall operates at the application layer of a protocol stack
  - A WAF (*Web Application Firewall*) inspects the HTTP traffic and prevents attacks, such as SQL injection, cross-site scripting (XSS), file inclusion, and other types of security issues
    - Example: ModSecurity for apache webserver
- It can block application input/output from detected intrusions or malformed communication, or block contents that violate policies
- It can detect whether an unwanted protocol is being provided through on a non-standard port or whether a protocol is being abused in any harmful way

TLS acceleration

- The SSL/TLS "handshake" process uses digital certificates based on asymmetric or public key encryption technology
- Public key encryption is very secure, but also very processor-intensive and thus has a significant negative impact on performance
  - SSL bottlenecks
- Possible solutions:
  - SSL acceleration: use hardware support to perform modular operations with large operands
  - SSL offloading: use a dedicated server only for SSL handshake

SSL offloading

- SSL Termination
  - The proxy decrypts the TLS/SSL-encrypted data and then sends it on to the server in an unencrypted state
  - This also allows IDS or application firewall inspection
- SSL Forwarding (or Bridging or Initiation)
  - The proxy intercepts and decrypts TLS/SSL-encrypted traffic, examines the contents to ensure that it doesn't contain malicious code, then re-encrypts it before sending it on to the server
  - This only allows inspection of TLS/SSL-encrypted data before it reaches the server to prevent application layer attacks hidden inside

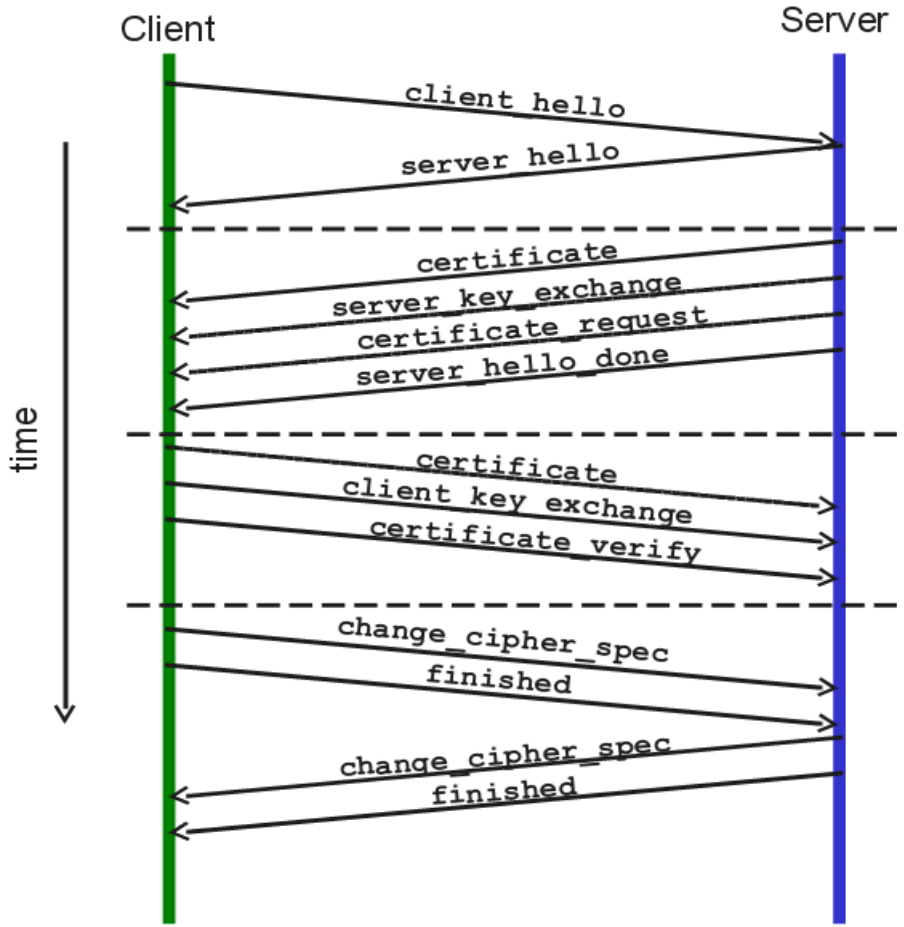
Proxy and HTTPS

- HTTPS traffic (fortunately) cannot be read
- What if we WANT to read a client HTTPS traffic?
- We need to perform a Man-in-the-middle attack
  - PRETEND to be the real server and be the termination of the SSL/TLS connection
- A possible solution is the **SSL bump**
- It consists in using the requested host name to dynamically generate a server certificate and then impersonate the named server
- But: what if we **don't know the host name**?
  - Remember that we start the TLS handshake using an IP address and the hostname is sent in the HTTP request...

HTTPS certificate dilemma

- Virtual host: the same webserver can host multiple websites
  - According to the hostname field in the HTTP header, the server understands the requested website
- If HTTPS is used, the SSL/TLS connection requires a certificate to be sent by the server, but…
- Which certificate has to be sent?
  - A certificate valid for all the websites is out of discussion
  - The hostname is within the HTTP traffic but it is **encrypted**
- Then?

TLS Handshake → BEFORE ANY HTTP packet



1) Hello phase

2) Server authentication

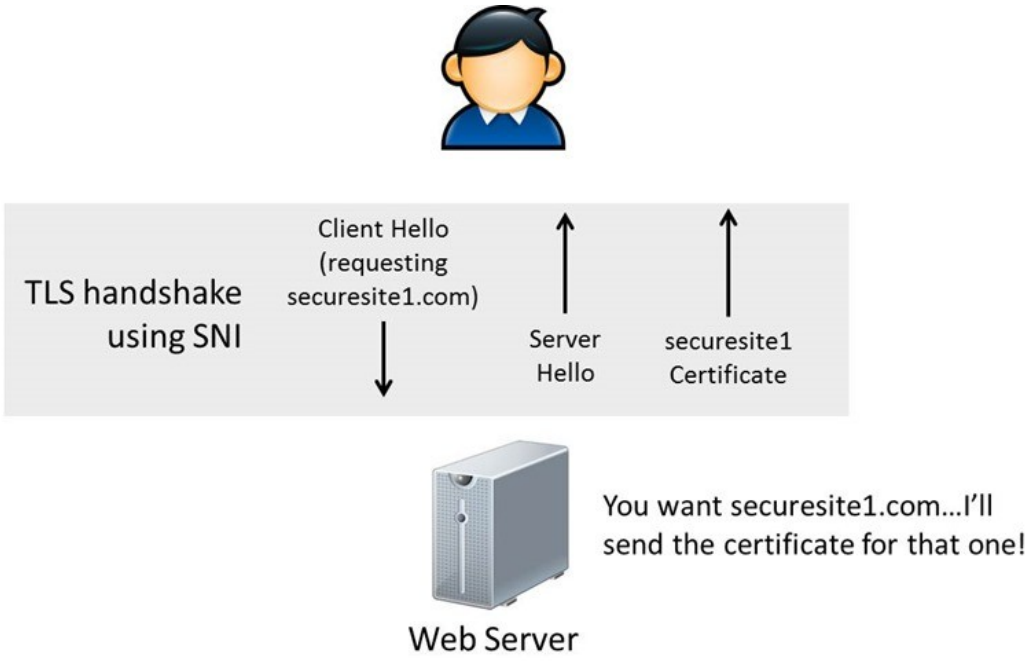
3) Client authentication

4) Finish

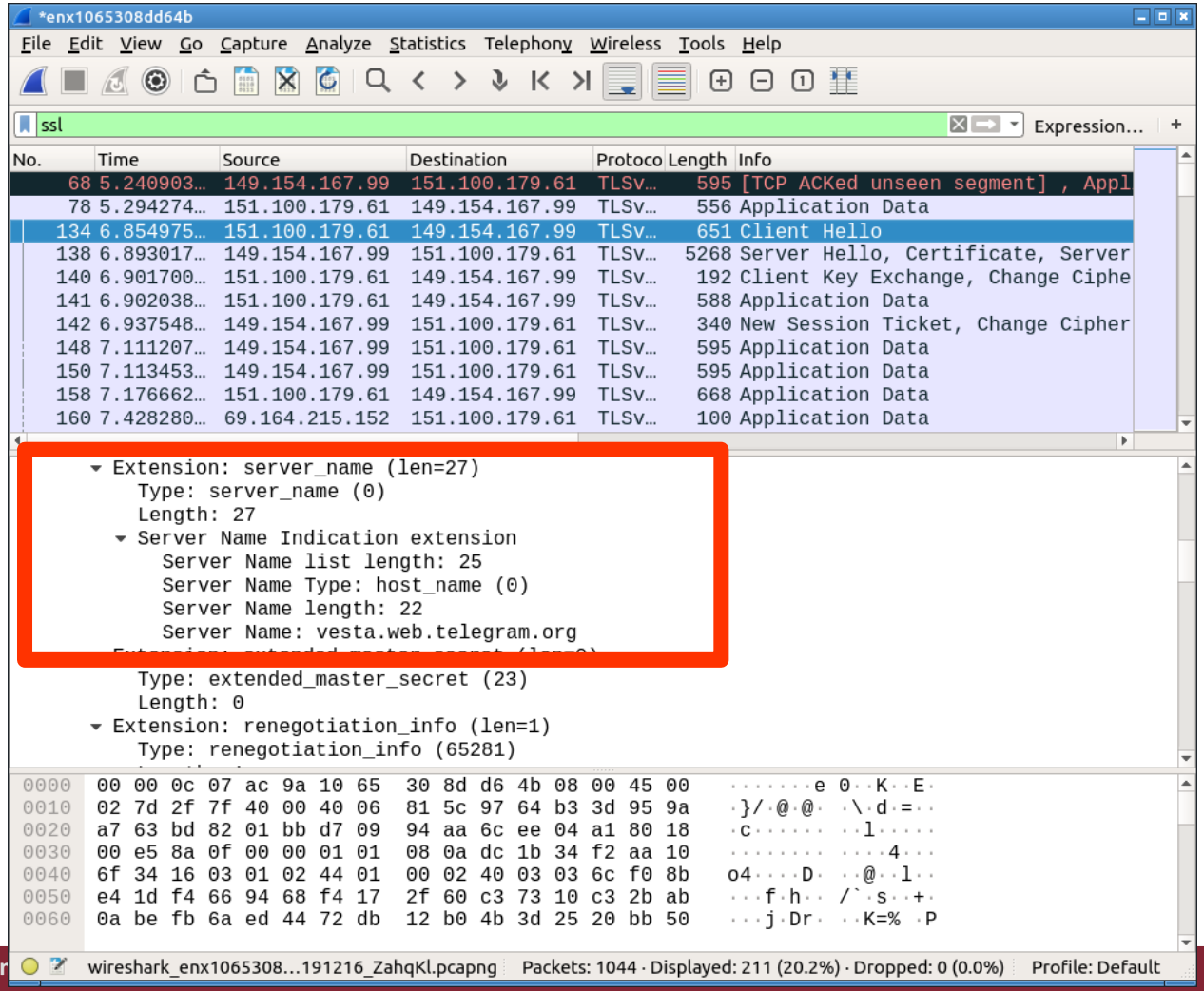
Server Name Indication

- Server Name Indication (SNI) is an extension to TLS by which a client indicates which hostname it is attempting to connect to at **the start of the handshaking** process
  - It is in clear text
- TLS Extensions RFC:
  - https://tools.ietf.org/html/rfc3546#section-3.1
- This allows a server to present **multiple certificates** on the same IP address and TCP port number and hence allows multiple secure (HTTPS) websites (or any other service over TLS) to be served by the same IP address **without requiring** all those sites to use the same certificate

SNI example


TLS handshake using SNI
Client Hello (requesting securesite1.com)
Server Hello
securesite1 Certificate
You want securesite1.com...I'll send the certificate for that one!
Web Server
https://www.securesite1.com
https://www.securesite2.com
https://www.securesite3.com

SNI capture


*enx1065308dd64b
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ssl
Expression…
No. Time Source Destination Protoco Length Info
68 5.240903… 149.154.167.99 151.100.179.61 TLSv… 595 [TCP ACKed unseen segment] , Appl
78 5.294274… 151.100.179.61 149.154.167.99 TLSv… 556 Application Data
134 6.854975… 151.100.179.61 149.154.167.99 TLSv… 651 Client Hello
138 6.893017… 149.154.167.99 151.100.179.61 TLSv… 5268 Server Hello, Certificate, Server
140 6.901700… 151.100.179.61 149.154.167.99 TLSv… 192 Client Key Exchange, Change Ciphe
141 6.902038… 151.100.179.61 149.154.167.99 TLSv… 588 Application Data
142 6.937548… 149.154.167.99 151.100.179.61 TLSv… 340 New Session Ticket, Change Cipher
148 7.111207… 149.154.167.99 151.100.179.61 TLSv… 595 Application Data
150 7.113453… 149.154.167.99 151.100.179.61 TLSv… 595 Application Data
158 7.176662… 151.100.179.61 149.154.167.99 TLSv… 668 Application Data
160 7.428280… 69.164.215.152 151.100.179.61 TLSv… 100 Application Data
Extension: server_name (len=27)
Type: server_name (0)
Length: 27
Server Name Indication extension
Server Name list length: 25
Server Name Type: host_name (0)
Server Name length: 22
Server Name: vesta.web.telegram.org
Type: extended_master_secret (23)
Length: 0
Extension: renegotiation_info (len=1)
Type: renegotiation_info (65281)
0000 00 00 0c 07 ac 9a 10 65 30 8d d6 4b 08 00 45 00
0010 02 7d 2f 7f 40 00 40 06 81 5c 97 64 b3 3d 95 9a
0020 a7 63 bd 82 01 bb d7 09 94 aa 6c ee 04 a1 80 18
0030 00 e5 8a 0f 00 00 01 01 08 0a dc 1b 34 f2 aa 10
0040 6f 34 16 03 01 02 44 01 00 02 40 03 03 6c f0 8b
0050 e4 1d f4 66 94 68 f4 17 2f 60 c3 73 10 c3 2b ab
0060 0a be fb 6a ed 44 72 db 12 b0 4b 3d 25 20 bb 50
wireshark_enx1065308…191216_ZahqKl.pcapng Packets: 1044 · Displayed: 211 (20.2%) · Dropped: 0 (0.0%) Profile: Default

SNI implications

- An eavesdropper can see which site is being requested
- This helps security companies provide a filtering feature and governments implement censorship
  - Trick: Domain Fronting
    - In the SNI use the name of a CDN server and then use a different host in the real HTTP request, always within the same CDN
- An upgrade called Encrypted SNI (ESNI) is being rolled out in an "experimental phase" to address this risk of domain eavesdropping

https://attack.mitre.org/techniques/T1172/

https://www.cloudflare.com/learning/ssl/what-is-encrypted-sni/

SOCKS proxy

- Circuit level gateway
  - Works at the session layer of the OSI model, or as a "shim-layer" between the application layer and the transport layer of the TCP/IP stack
- Similar to the HTTP CONNECT proxy
  - A bit more versatile:
    - Many authentication mechanisms
    - Can tunnel TCP, but also UDP and IPv6 (SOCKS5)
    - Can also work as a reverse proxy
- Implemented in SSH, putty and Tor

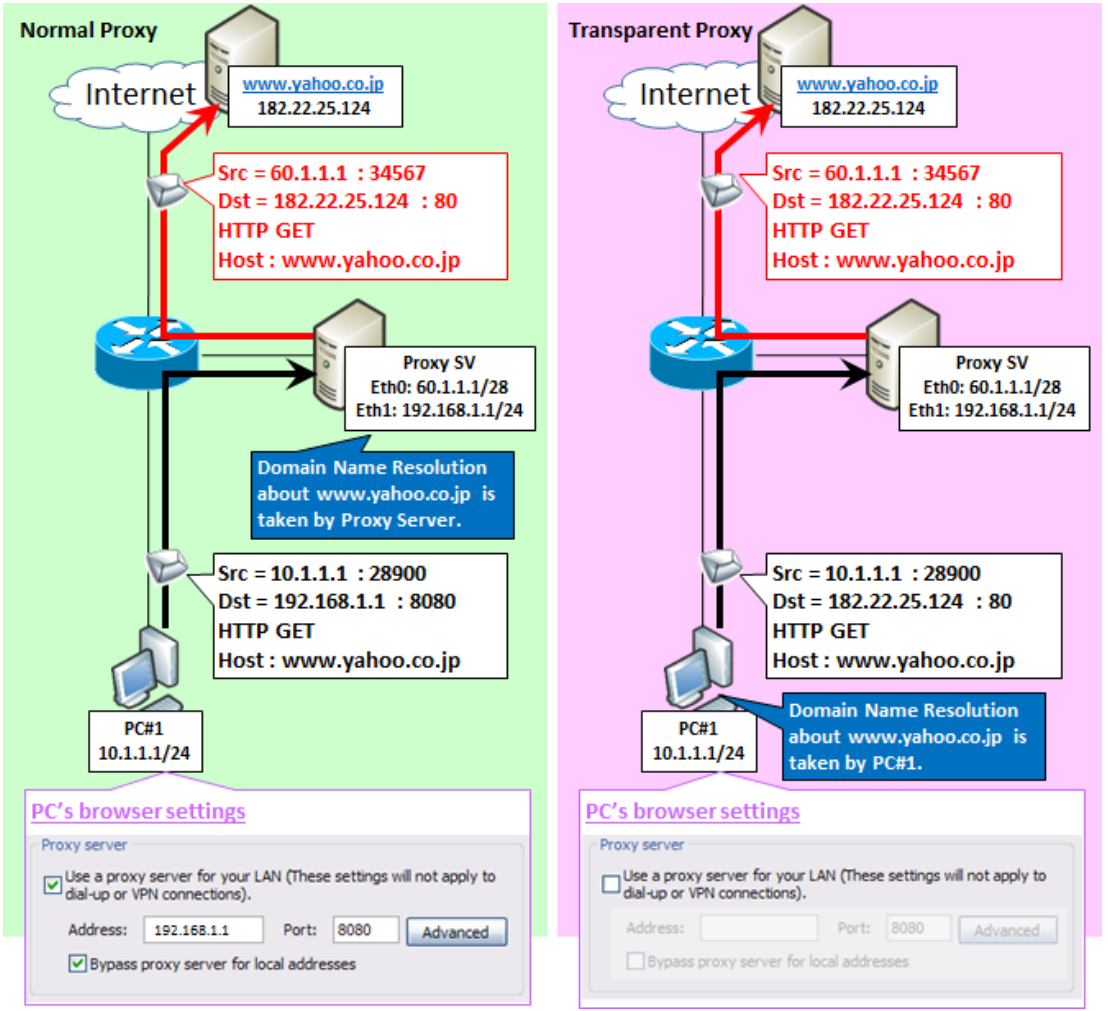
Transparent proxy

- Forward proxy need proxy-capable client programs and/or user education so that users know how to use the proxies
- A transparent proxy is made for normal user procedures and normal client applications
  - *A transparent application proxy is often described as a system that appears like a packet filter to clients, and like a classical proxy to servers* (RFC1919)
- Alternative names:
  - Intercepting proxy
  - Inline proxy
  - Forced proxy
- Common practice for some ISPs (mainly for mobile users)

How transparent proxy works

- At the start of a session, a TCP packet with a source address of Client and a destination address of Server travels to the proxy system, expecting to cross it just like a normal IP gateway
- The proxy's TCP/IP software stack sees this incoming packets for a destination address that is NOT one of its own addresses
- Forward or drop the packet? → ACCEPT!
  - It PRETENDS to be the Server
  - And then operates like a standard proxy, as a middle-point between Client and Server
- What is the difference with NAT?

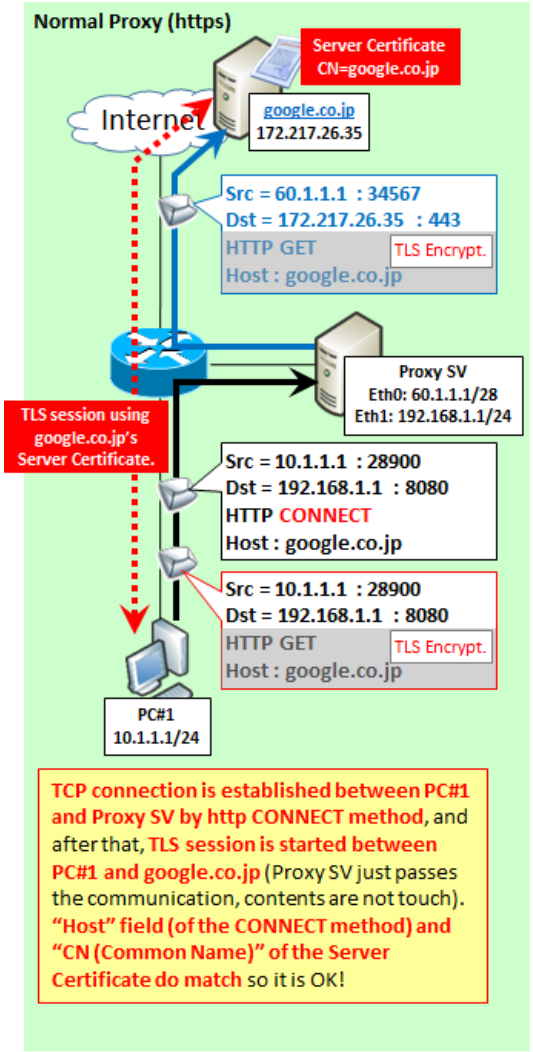
Normal proxy vs. Transparent proxy

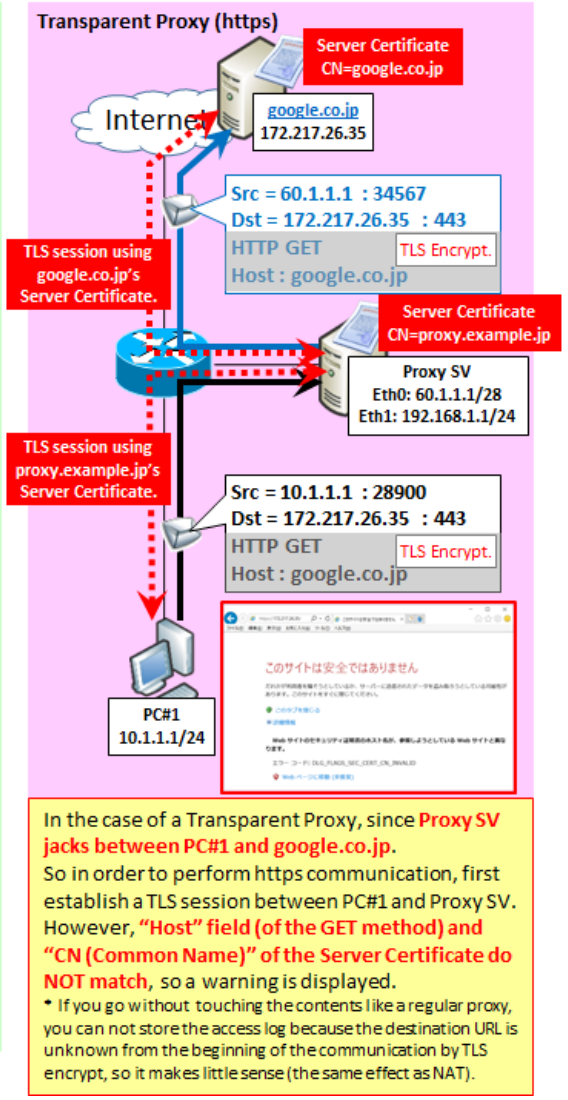


Transparent proxy caveats

- Proxy intercepts the HTTP requests that has the IP address of the Server
  - Again, it does not know the hostname, but has to look for in within the HTTP request
  - What about HTTPS? → See reverse proxy and SSL bump
- How to intercept the HTTP/HTTPS requests?
  - If we do the interception at the default route, we will break other protocols such as SMTP, POP, IMAP
  - Then?
    - Policy-based routing (PBR)

HTTPS in normal vs. Transparent Proxy





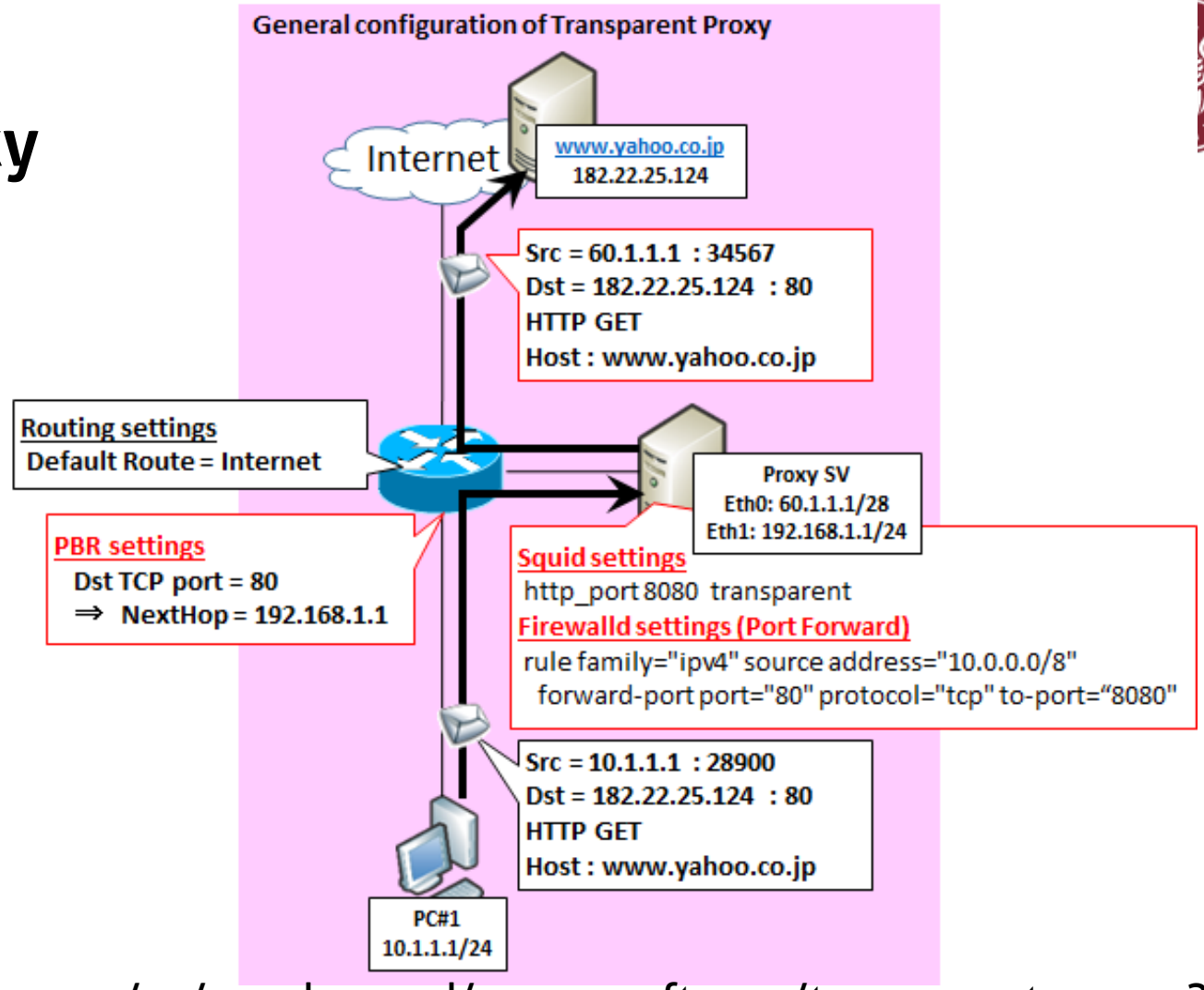
Policy based routing

- In traditional routing: "All routing is a destination-driven process"
- What about routing decisions based on other info?
  - Quality of service
  - Source routing
  - Traffic shaping
  - Etc.
- Policy based routing goes beyond simple destination-driven routing

PBR and transparent proxy

- We can use as policy:
  - any TPC connection with destination port 80 MUST be forwarded to the proxy
- Then we need a different routing table, with a different next hop → the proxy
- This can be done via iptables and the "packet marking"

Configuration of a Transparent Proxy



https://milestone-of-se.nesuke.com/en/sv-advanced/server-software/transparent-proxy-2/

ICAP: Internet Content Adaptation Protocol

- Defined in RFC-3507, it is a protocol aimed at providing simple object-based content vectoring for HTTP services
- ICAP is, in essence, a lightweight protocol for executing a "remote procedure call" on HTTP messages
- It allows ICAP clients to pass HTTP messages to ICAP servers for some sort of transformation or other processing ("adaptation")
- The ICAP server executes its transformation service on messages and sends back responses to the client, usually with modified messages
  - Typically, the adapted messages are either HTTP requests or HTTP responses

ICAP examples

- Coupled with transparent proxy (acting as a ICAP client), it allows to transparently provide additional functionalities, using a standardized interface towards ICAP servers
- Examples:
  - Simple transformations of content can be performed near the edge of the network instead of requiring an updated copy of an object from an origin server
    - Ex: translations to other languages, formatting for different devices, inclusions of different advertisements, and so on
  - Expensive operations on the content can be performed by "surrogates" instead of the origin server
    - Ex: file scan for viruses, check file upload/download, and so on
  - Checking if requested URIs are allowed or not
    - Ex: parental control, content filtering, and so on

That’s all for today

- Questions?
  - See you next lecture!
- References:
  - Ari Luotonen, Kevin Altis, World-Wide Web Proxies, 1994
  - http://httpd.apache.org/docs/current/mod/mod_proxy.html
  - https://en.wikipedia.org/wiki/Proxy_server
  - Classical vs Transparent IP Proxies, RFC 1919
  - SOCKS 5, RFC 1928
  - HTTP 1.1, RFC 7230
  - Policy based routing and Linux advanced routing and traffic control
  - ICAP, RFC 3507

Student's Opinions Questionnaires (OPIS)

- For the Practical Network Defense course
- In infostud follow the following instructions https://www.uniroma1.it/sites/default/files/field_file_allegati/vadevecum_opis_eng_27_11_2018_002_modalita_compatibilita.pdf
    - use this course code **YHES2SB9**
- **Be (pro-)positive!**

YHES2SB9