Sentinel

Adaptive NIDS

DEFENDING OUR DIGITAL WAY OF LIFE

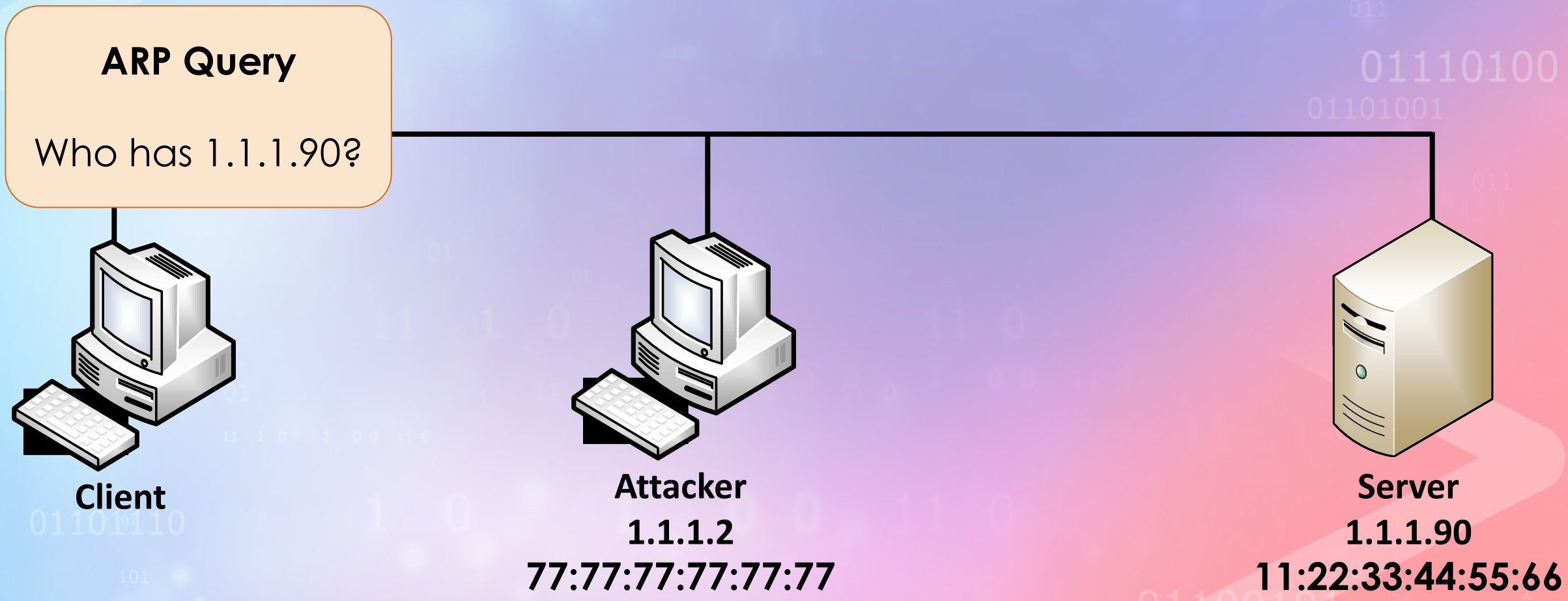
Lesson objectives

→ Explore autonomous **network learning**

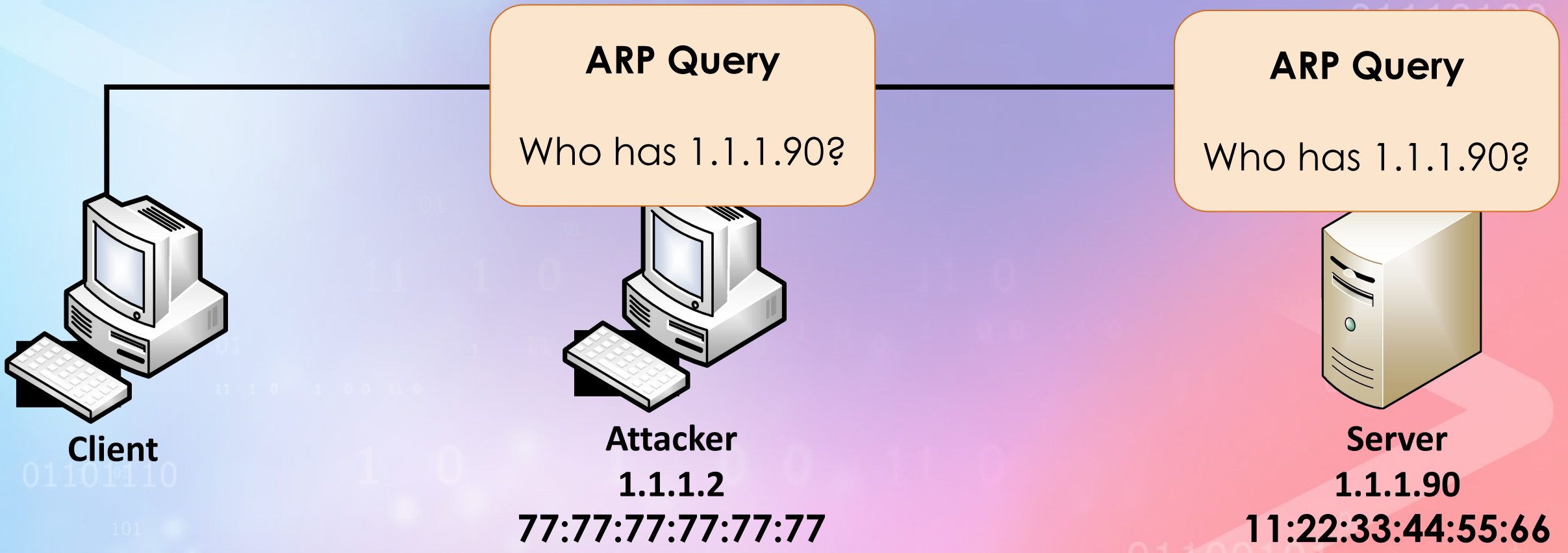
→ Learn about **ARP spoofing** attacks and how to detect

→ Understand how to detect **anomalies** in network

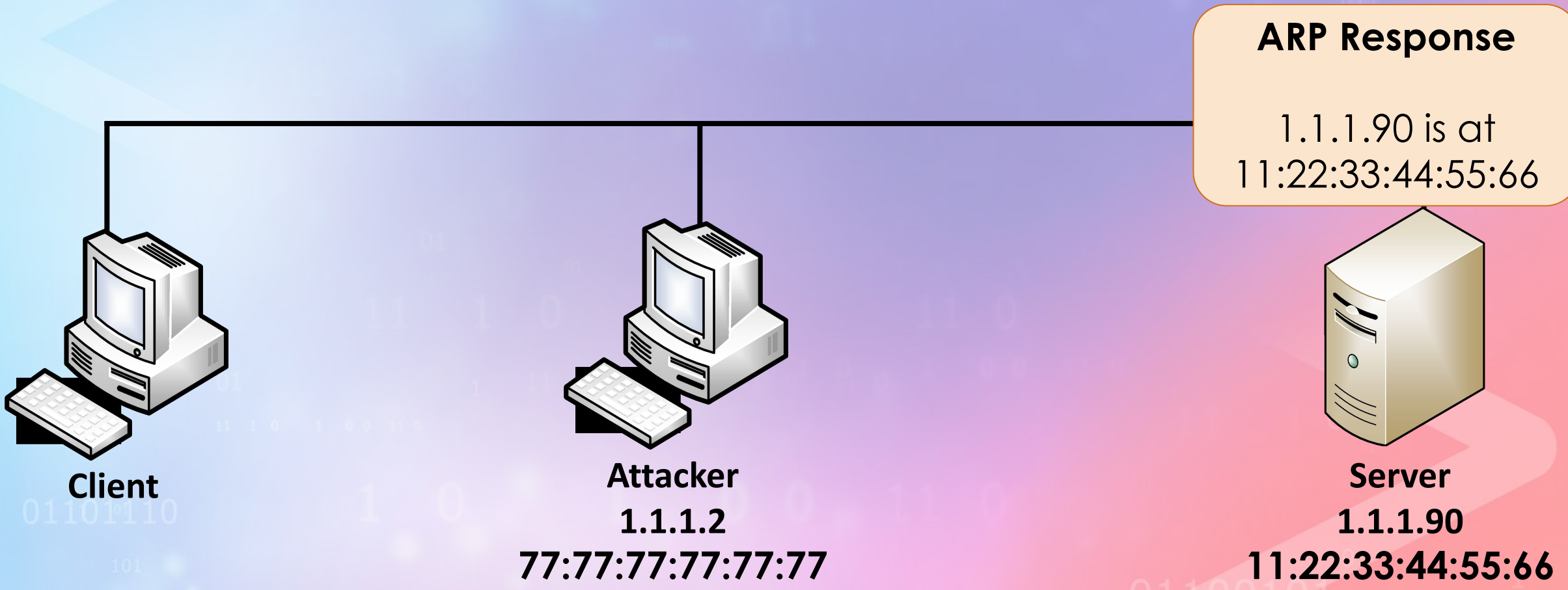
ARP resolving



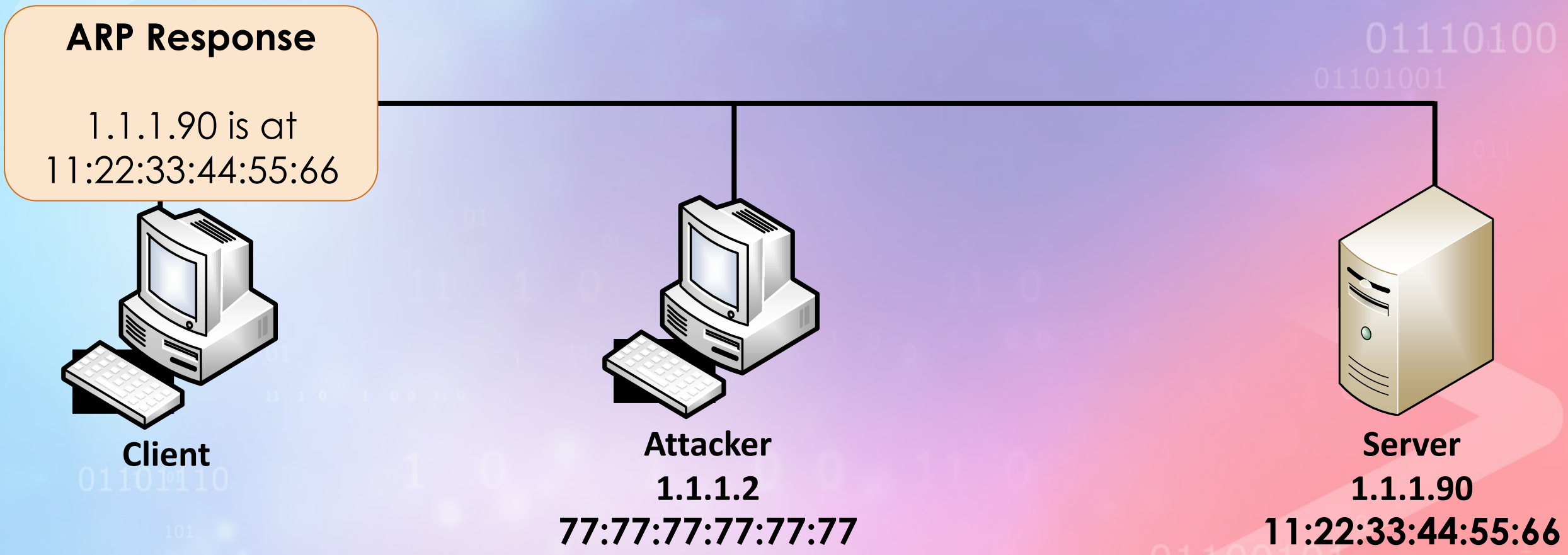
ARP resolving



ARP resolving

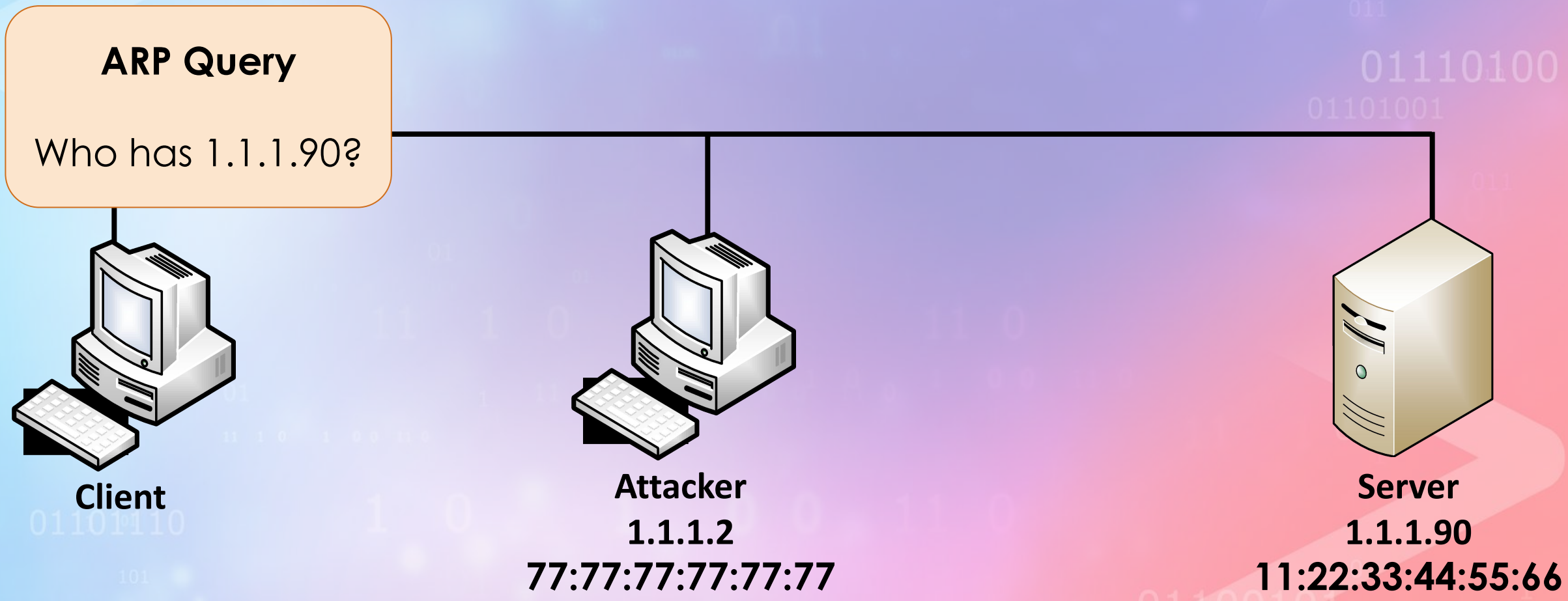


DNS resolving

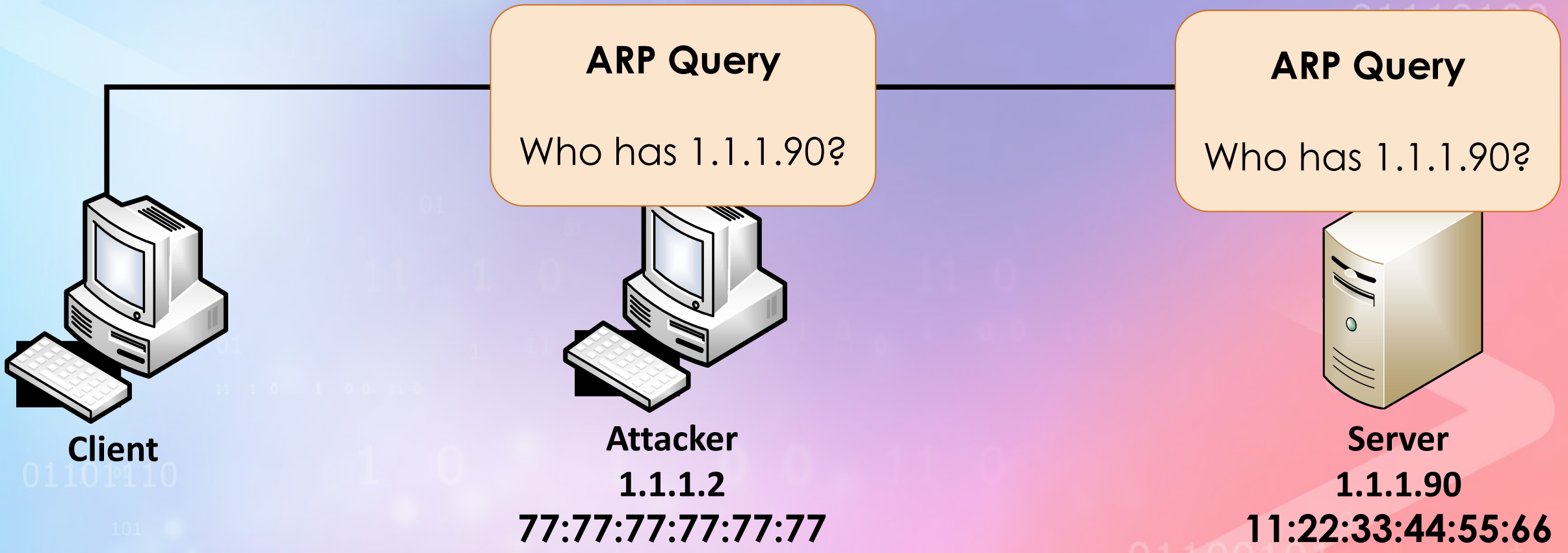


ARP spoofing / poisoning

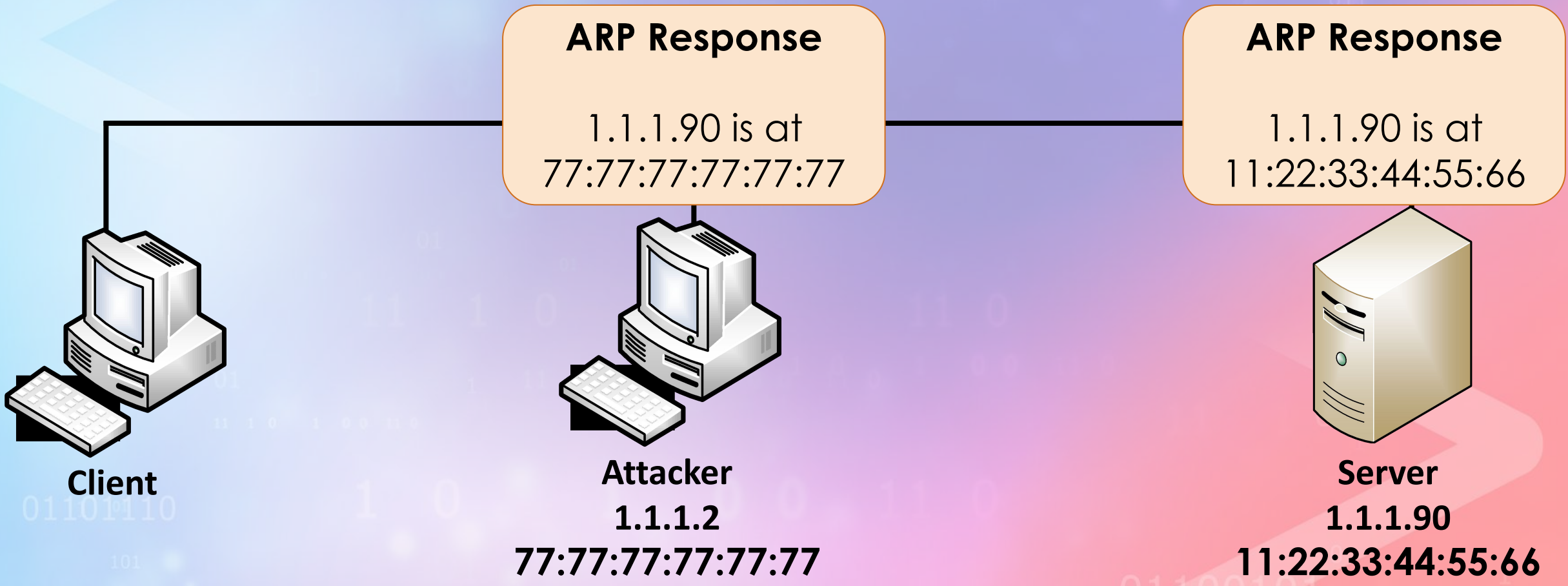
ARP spoofing



ARP spoofing



ARP spoofing



ARP spoofing

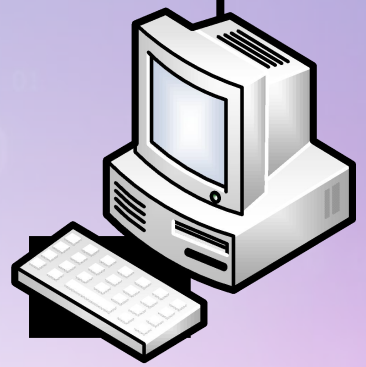
ARP Response

1.1.1.90 is at
77:77:77:77:77:77

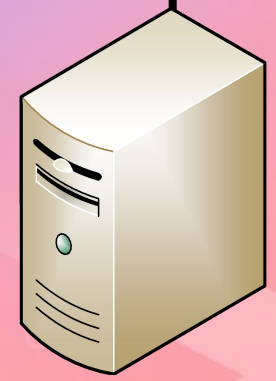
ARP Response

1.1.1.90 is at
11:22:33:44:55:66

Client



Attacker
1.1.1.2
77:77:77:77:77:77



Server
1.1.1.90
11:22:33:44:55:66

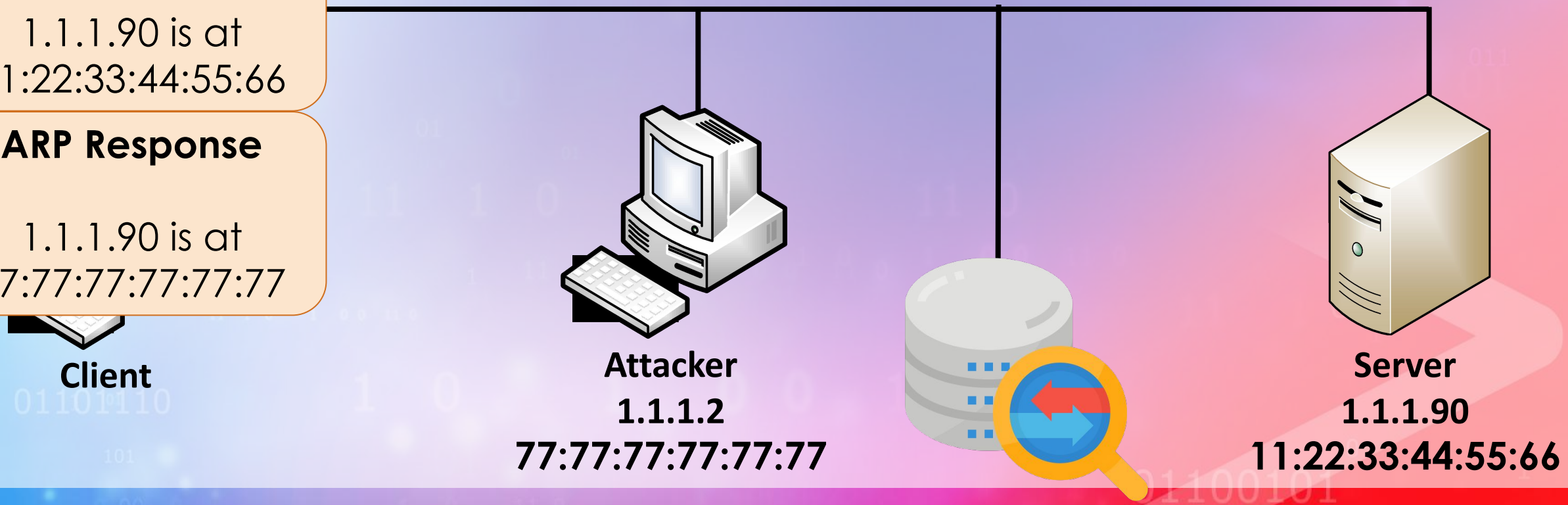
How could we detect ARP spoofing?

ARP Response

1.1.1.90 is at
11:22:33:44:55:66

ARP Response

1.1.1.90 is at
77:77:77:77:77:77



Client

Attacker
1.1.1.2
77:77:77:77:77:77

Server
1.1.1.90
11:22:33:44:55:66

Method #1 - Static definition

- The network administrator knows the static IP and MAC of the server
- Why not provide this information to the NIDS?
- Set up expected IP:MAC pairs in the NIDS user interface
  - And if we see an ARP response with a known IP but wrong MAC
  - Alert!

Dynamic learning

- An ARP poisoning attack is an **anomaly**
  - IP X is typically associated with MAC Y
  - But now that there's a response saying it's at MAC Z, indicates an **anomaly**.
- Modern NIDS can learn the network traffic
- And detect anomalies autonomously

Baseline

- First, observe traffic over a set period to establish a baseline of normal activity
- After this "learning phase", start alerting on anything that deviates from the baseline

What else could we learn?

The big question - what to learn?

- We could dynamically learn many things about network traffic
- There should be a justification
  - i.e. it can lead to a detection
  - Should we track user agents?
    - An anomaly in user agent doesn't indicate malicious activity
- This is the importance of knowing your enemy

Dynamically learning thresholds

- We previously defined static thresholds and window sizes
  - To detect ARP/port scans and SYN flood
- But this could also be learned dynamically
- Removes the effort of fine-tuning thresholds
- Let the code figure it out!

Dynamically learning protocol ports

- We previously defined expected protocol ports
  - e.g. HTTP is always on port 80
- But this could also be learned dynamically
- Deep packet inspection can detect specific protocols
  - And identify the port they're communicating in

Q&A