# Client-Server Model

Ahmet Kaşif
Bursa Teknik Üniversitesi
Bilgisayar Mühendisliği Bölümü
Ekim 2023

# What is the client-server model?

The client-server model is a distributed system architecture that separates the application into two main components:

- **Clients**: Responsible for making requests to servers
- **Servers**: Responsible for processing those requests and sending back responses.

# Benefits of the client-server model

- **Scalability**: The client-server model is highly scalable, meaning that it can be easily adapted to handle a large number of users and requests.
- **Availability**: Client-server systems are typically more available than centralized systems, because they can continue to operate even if one or more servers fail.
- **Security**: Client-server systems can be more secure than centralized systems, because they can be designed to restrict access to sensitive data and resources.
- **Flexibility**: The client-server model is very flexible, and can be used to implement a wide variety of applications.

# Examples of the client-server model

- **Web applications**: When you visit a website, your web browser (the client) sends a request to the web server (the server) for the HTML, CSS, and JavaScript files that make up the website. The web server then sends those files back to your browser, which renders them to display the website.
- **Email**: When you send an email, your email client (the client) sends a request to the email server (the server) to send the email to the recipient. The email server then sends the email to the recipient's email server, which stores it until the recipient is ready to read it.
- **File sharing**: When you share a file with someone over a network, your computer (the client) sends a request to the other computer (the server) to download the file. The server then sends the file to your computer.

# How does the client-server model work?

1. The client sends a request to the server. The request can be in the form of an HTTP request, a database query, or some other type of message.
2. The server receives the request and processes it. This may involve retrieving data from a database, performing calculations, or interacting with other servers.
3. The server sends a response back to the client. The response can be in the form of an HTML page, a JSON object, or some other type of message.
4. The client receives the response and displays it to the user.

# Client-side security

- Don't expose sensitive data on the client-side. This includes things like passwords, credit card numbers, and social security numbers. If you need to collect or store this type of data on the client-side, make sure to encrypt it.
- Retrieve only the necessary data for the screen. Don't retrieve more data from the server than you need to display on the current screen. This will help to reduce the risk of data breaches.
- Use strong input validation. Make sure to validate all user input before sending it to the server. This will help to prevent attackers from injecting malicious code into your application.
- Use a content security policy (CSP). A CSP is a security policy that restricts the types of resources that a web browser can load. This can help to prevent attackers from injecting malicious code into your application via cross-site scripting (XSS) attacks.

# Server-side security

- Keep your software up to date. Software vendors regularly release security patches to fix known vulnerabilities. Make sure to install these patches as soon as they are available.
- Use strong authentication and authorization. Make sure that only authorized users can access your server and its resources. You should also use strong passwords and two-factor authentication whenever possible.
- Use a firewall. A firewall can help to protect your server from unauthorized access and attacks.
- Encrypt sensitive data. Any sensitive data that is stored on your server should be encrypted. This will help to protect it from unauthorized access, even if your server is compromised.
- Monitor your server logs. Regularly review your server logs for any suspicious activity. This can help you to identify and respond to attacks early on.

# Firewalls

A firewall is a network security device that monitors and controls incoming and outgoing network traffic based on predetermined security rules. A firewall typically establishes a barrier between a trusted internal network and an untrusted outside network, such as the Internet.

Firewalls can be implemented in hardware, software, or a combination of both. Hardware firewalls are dedicated devices that are typically installed at the edge of a network. Software firewalls can be installed on individual computers or servers.

# Different approaches to firewalls

- **Packet filtering**: Firewalls can inspect individual data packets to determine whether they should be allowed or blocked. Packet filtering can be based on a variety of factors, such as the source and destination IP addresses, the port numbers, and the type of traffic.
- **Stateful inspection**: Stateful inspection firewalls maintain a state table of all active network connections. This allows them to track the flow of traffic and make more informed decisions about whether to allow or block packets.
- **Application inspection**: Application inspection firewalls can inspect the contents of data packets to identify specific applications and protocols. This allows them to block malicious traffic, such as viruses and malware, even if it is disguised to look like legitimate traffic.

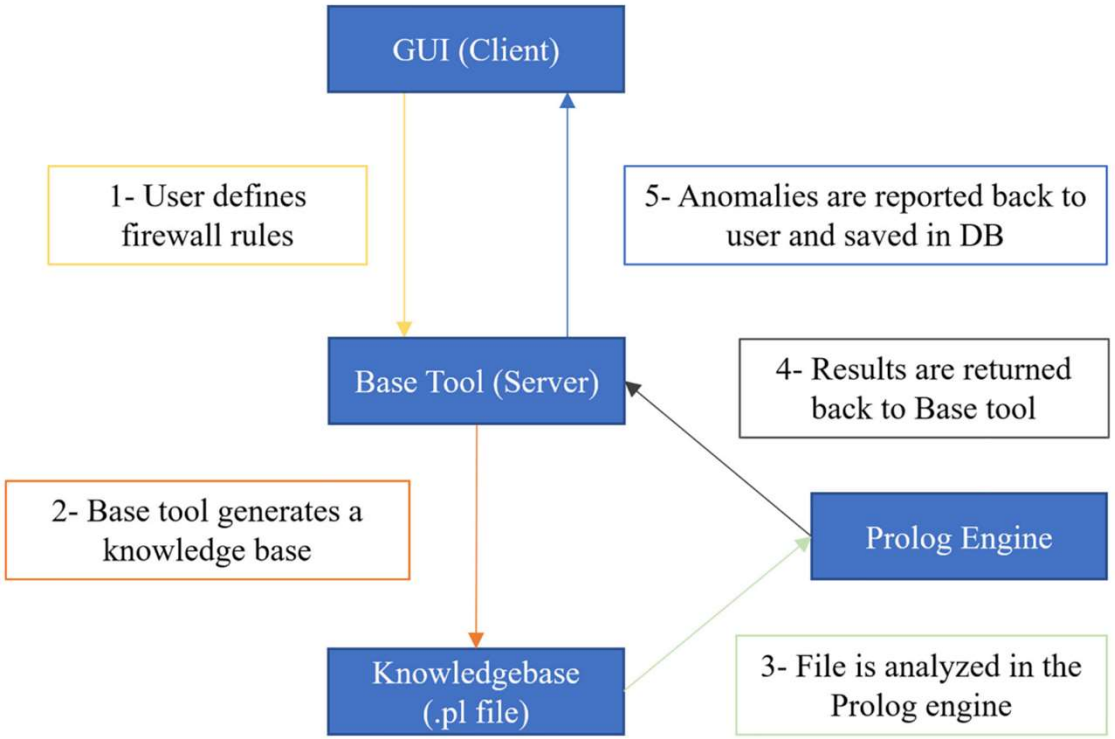
# A case study: Firewall Optimization

- Packet-filtering firewalls, policy rules are implemented to monitor changes to the network and preserve the required security level.
- With the rapid increase of the size of the policy rules, firewall policy anomalies occur more frequently.
- An anomaly detection framework for detecting intra-firewall policy anomaly rules is developed.

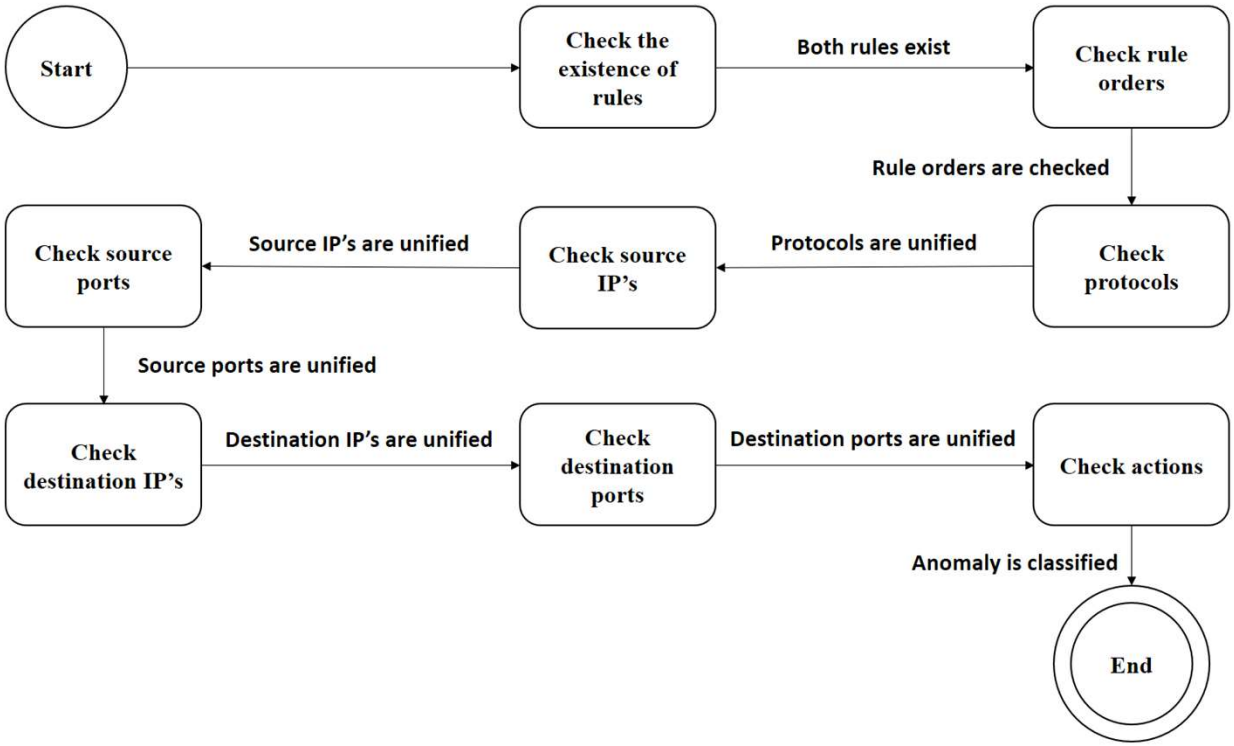
# A packet-filtering firewall (iptables)

| Order | Protocol | Source IP | Source Port | Destination IP | Destination Port | Action |
|---|---|---|---|---|---|---|
| **1** | TCP | 192.152.1.* | ANY | 128.172.26.* | 80 | *ALLOW* |
| **2** | TCP | 192.152.1.72 | ANY | 128.172.*.* | 80 | *DENY* |
| **3** | TCP | 192.152.1.* | ANY | 128.172.26.2 | 80 | *ALLOW* |
| **4** | TCP | ANY | ANY | ANY | 80 | *DENY* |
| **5** | TCP | 151.*.*.* | ANY | 108.56.56.1 | 53 | *DENY* |
| **6** | TCP | 151.126.*.* | ANY | 108.56.56.1 | 53 | *ALLOW* |
| **7** | TCP | 151.51.37.* | ANY | 108.56.*.* | 53 | *ALLOW* |
| **8** | UDP | 216.22.14.* | ANY | 124.24.*.* | 53 | *ALLOW* |
| **9** | UDP | 216.22.14.1 | ANY | 124.24.*.* | 53 | *DENY* |
| **10** | UDP | 216.22.14.1 | ANY | 124.*.*.* | 53 | *DENY* |
| **11** | UDP | 25.12.*.* | ANY | 124.*.*.* | ANY | *ALLOW* |
| **12** | ANY | ANY | ANY | ANY | ANY | *DENY* |

# Client-Server based application structure for the firewall optimization platform



# Firewall Optimization: The algorithm



# Results of detected anomalies

| # | $R_1$ | $R_2$ | $R_3$ | $R_4$ | $R_5$ | $R_6$ | $R_7$ | $R_8$ | $R_9$ | $R_{10}$ | $R_{11}$ | $R_{12}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $R_1$ | - | C | R | G | - | - | - | G | - | - | - | - |
| $R_2$ | - | - | C | R | - | - | - | - | - | - | - | R |
| $R_3$ | - | - | - | G | - | - | - | - | - | - | - | G |
| $R_4$ | - | - | - | - | - | - | - | - | - | - | - | R |
| $R_5$ | - | - | - | - | - | S | C | - | - | - | - | R |
| $R_6$ | - | - | - | - | - | - | - | - | - | - | - | G |
| $R_7$ | - | - | - | - | - | - | - | - | - | - | - | G |
| $R_8$ | - | - | - | - | - | - | - | - | S | C | - | G |
| $R_9$ | - | - | - | - | - | - | - | - | - | R | - | R |
| $R_{10}$ | - | - | - | - | - | - | - | - | - | - | - | R |
| $R_{11}$ | - | - | - | - | - | - | - | - | - | - | - | G |
| $R_{12}$ | - | - | - | - | - | - | - | - | - | - | - | - |

S: Shadowing, R: Redundancy, G: Generalization, C: Correlation, - : No Anomaly

# Thanks for listening

## References:


- Togay, C., Kasif, A., Catal, C., & Tekinerdogan, B. (2021). A firewall policy anomaly detection framework for reliable network security. *IEEE Transactions on Reliability*, *71*(1), 339-347.