High-Level Test Plan for Firewall as a Service (FWaaS)

Summary

This test plan describes the functional testing of Firewall as a Service in AlfaBeta CyberSecurity organization. As firewall services develop with time, now Firewall service moves from on-prem to the cloud to provide wide-range of protection for services running on the cloud. This Test plan's aim is to validate that the feature meets business, technical and regulatory requirements, and that feature itself protects corporate dataflows in every environment and uses granular access control to better prevent potential breaches.

The test plan is based on the Technical Design documents located at the link below:

FrontEnd Story as provided in: https://www.alfabeta.com/firewall-as-a-service

BackEnd Story as provided in:
https://www.alfabeta.com/cyber-hub/network-security/firewall-as-a-service-fwaas/

Note: notation "if applicable" is an assumption based on QA professional experience with other Network Security features (Permissions/VPN/S2S/ZTNA).

1. Objective

The main issue of moving Firewall from on-premise to the Cloud is in my opinion the illusion of losing control on the traffic that goes via servers. All traffic goes via VPN, meaning that cloud services must sift through a lot of traffic, protocols, connections and this may ultimately mean that someone or something with malicious intents can penetrate the system that is located somewhere else, on the cloud. This is why Firewall as a Service must provide not only thoughtful functionality, security, performance, scalability, and reliability of the Firewall as a Service (FWaaS) solution, but also means to the end user to control the situation. The following testing ensures not only that the firewall effectively filters network traffic, prevents unauthorized access, and performs optimally under various conditions, but also gives meaningful tools to the end user to see all the processes that are going on (secure connection, mobile applications that can be run in cloud, programs, services, exclusions, filtering, etc).

2. Scope

This test plan covers:

● FrontEnd application that allows SISO to fully control access to the involved applications (user experience: IT has direct line-of-sight with all corporate data, at rest and in transit) ● Zero-Trust Application Access implementation
● Filtering of inbound and outbound traffic across hybrid clouds (Azure, AWS) ● Policy enforcement
● Functional testing of firewall rules and policies that were created/applied by SISO (Definition of detailed rules for how and when traffic moves inside client's networks).
● Security testing against threats and vulnerabilities (Penetration testing)
● Performance and scalability testing when there the number of active system users grows

exponentially
● Compliance and logging validation based on current regulations in the applicable industry

3. Test Approach

A combination of manual and automated testing methodologies will be used. The approach includes:

● Black-box testing for functional validation (assigned to Manual QA Engineers) ● White-box testing for rule execution and internal processes (Unit tests done by Developers) ● Load testing for performance evaluation (Manual testing using JMeter or other cloud-oriented tools)
● Penetration testing for security assessment (Penetration and Security-assigned engineers)

4. Test Categories and Scenarios (UI)

4.1. Level of Access

● Evaluating level of access based on Client's plan (Premium/Premium Plus/Enterprise/Custom (if applicable))
● Separation of environments between different clients
● Separation of Networks within Client's environment
● Authentication of users (per environment/Network) via using Azure Active Directory: ○ 2FA
  ○ Single-SignOn
● Levels of user access based on permissions (SISO, compliance managers, sysadmins, security auditors, custom users, groups of users)

4.2 Design

● Modern, user-friendly design of the platform that includes smooth operation of all windows, pop-ups and overlays
  ○ Grid (Empty Grid, grid with many rules, pagination)
  ○ Actions on the Grid (Add, View, Edit, Delete Rules)
  ○ Alerts: Verify alerts, notifications, and logs are clear and actionable
  ○ Blocked traffic screen

4.3 Devices

● Company-issues devices
● User's devices (Under policy of "Bring Your Own Device")

4.4 Application

● Cloud Applications (Zoom, Slack, OnePassword, etc)
● Web Applications & SaaS Platforms (Jira, Confluence, ZenDesk)
● On-Premises Applications (Postgres, MongoDD, SalesForce, SAP)
● Remote Access & VPN Solutions (OpenVPN, SSH, etc)
● Security & DevOps Tools (DataDog, Docker, Kubernetes, Kibana, Grafana, etc)

4.5 Rules

● Default state of rules (Allow/Deny)
● Search and Filters in the Rules
● Duplication of the rules - insure that least-privilege access is applied ● Deletion of the Rules
● Checking activity in the cloud after Rule was created/altered/deleted **4.7**

Destinations

4.7 Encryption

4.8 Supported Protocols

● TCP
● UDP
● RDP
● DNS
● ICMP
● Peer-to-peer (if applicable)
● FTP (if applicable)

4.9 Ranges

● IP Ranges
● Port ranges

4.10 DNS Filtering

● Lists
● Manually added sites
● Block/Allow sites

4.11 API

● Analize customer-server APIs for carrying correct information that then is going to be sent to Firewall servers and recorded in DB

4.12 Logs

● Analyzing logs to make sure that there is no major errors recorded, no suspicions spikes in activities
● Ensure blocked traffic generates proper error messages.

4.13 Stress

○ Large amount of data
○ Large amount of concurrent users
○ Large amount of concurrent transactions

4.14 Accessibility (if applicable):

- ○ Tab navigation
- ○ Screen reader

4.15 Used Tools:

- ○ Netcat
- ○ Wireshark (if applicable, based on Company policy)
- ○ jMeter (if applicable)
- ○ Selenium IDE for repetitive UI tests (if applicable)

4.16 Browser supported (Chrome, Chromium, Safari, Edge, Firefox)

4.17 Audit log (correctly record all and any actions performed in the system by users)

5. Test Categories and Scenarios (Logic)

5.1. Functional Testing (BackEnd, Logic)

● Verify that firewall rules correctly allow or block traffic based on IP, port, and protocol. ● Validate rule prioritization and conflict resolution (for example, duplication of rules). ● Test inbound and outbound traffic filtering.
● Ensure Deep Packet Inspection (DPI) correctly identifies threats (if applicable). ● Verify integration with cloud services and APIs (if applicable).
● Verify integration with databases and how information is absorbed with DB and how this information is disseminated on client's Networks

5.2 Security Testing

● Verify that all connections are secured (all TLS/SSL certificates are up-to-date) ● Test against common attacks (e.g., DDoS, Java and SQL Injection, anything that can be performed by QA Engineer, etc).
● Conduct penetration testing to simulate real-world threats.
● Validate SSL/TLS inspection and encryption handling.
● Ensure proper logging and alerting of security incidents.

5.3 Performance and Scalability Testing

● Measure latency and packet loss under normal and peak loads.
● Load balancing: validate that load balancer works correctly with the chosen scenario for high-volume traffic scenarios
● Test auto-scaling capabilities for dynamic traffic loads.
● Perform stress testing to identify breaking points.
● Tests of costs for variety of traffic (work-related or not (for example, Torrent traffic) **5.4**

Compliance and Logging Validation

● Verify that logs do not contain sensitive information (logins, passwords, phone numbers, etc). ● Ensure all firewall activities are logged correctly.
● Test log retrieval, filtering, and storage mechanisms.

6. Test Environment

● **Hardware/Software Requirements**: Cloud-based and on-premises firewall deployment. ● **Test Tools**: Network traffic simulators, penetration testing tools, log analysis tools. ● **Test Data**: Predefined traffic patterns, attack vectors, real-world workloads.

7. Test Execution Plan

● Define test cases for each category.
● Execute tests in a controlled environment.
● Involve relevant people for the test outside the QA scope
● Document and analyze test results.
● Identify and resolve issues before deployment.

8. Risks and Mitigation Strategies

● **Risk**: False positives blocking legitimate traffic.
    Mitigation: Fine-tune rules and conduct user acceptance testing.
● **Risk**: Performance degradation under high load.
    Mitigation: Optimize configurations and enable auto-scaling.
● **Risk**: Firewall bypass due to misconfiguration.
    Mitigation: Implement regular audits and automated compliance checks.

9. Conclusion

This test plan ensures that Firewall as a Service provided by AlfaBeta CyberSecurity is robust, secure, and scalable, providing effective network protection while maintaining optimal performance, with some prominent features like usage of Azure AD that provides seamless integration of 2FA. Continuous monitoring and improvements will be applied based on test findings.