



Enhancement Proposal

pink fluffy unicorns



OUR TEAM

Benedict Miguel
Jiachen Wang
Keshav Gainda
Ketan Bhandari
Mohaimen Hassan



Pegah Fallah
Shami Sharma
Sidharth Sudarsan
Suryam Thaker
Xiaochuan Wang



01 Proposed Feature

04 Testing Plan

02 Implementation Approaches

- Stakeholders
- Pros and Cons
- Chosen approach

05 Risks and Limitations

03 Potential Impact

- Impacted subsystems

06 Lessons Learned & Conclusion





FreeBSD's Usage As A Server

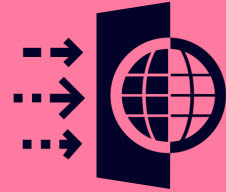
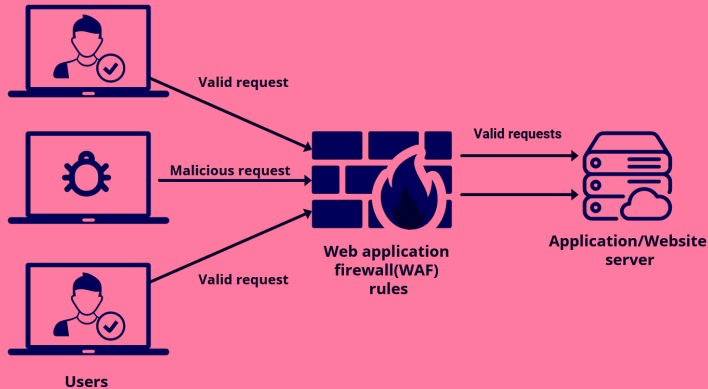
- **Popular server OS**
 - Web servers and other internet services such as Email, FTP, etc.
 - Due to stability, security and scalability
 - **Used by high-traffic sites**
 - Market position shows that although less popular, more high traffic sites use FreeBSD over Unix.
- Concern for security is present
- Need for additional layer of security for any web services running on the system
- Hence, ensuring security and usability for a wider range of users and use cases



Web Application Firewall (WAF)

- FreeBSD is a popular choice for web servers and other network services
 - Challenging to set up and maintain these services *securely*

➤ Built-in Web Application Firewall (WAF)





Third-party Software

- ModSecurity
- NAXSI



- Install and configure with the webserver being used such as *Apache* or *Nginx*
- Can be complex to set up and maintain depending on the scale of the network and the services.
- Aim to offer a simpler solution that is optimized for the FreeBSD platform.





Web Application Firewall (WAF)

- **Improved Security**
 - Additional layer of security for web applications and network services
- **Enhanced Performance**
 - Built-in solution could provide better performance and lower overhead
- **Cost and Management**
 - Save on third-party software payments
 - Simpler management due to absence of a separate security solution
- **Increased adoption**
 - Make FreeBSD an attractive option, increasing adoption and usage



Kernel Module

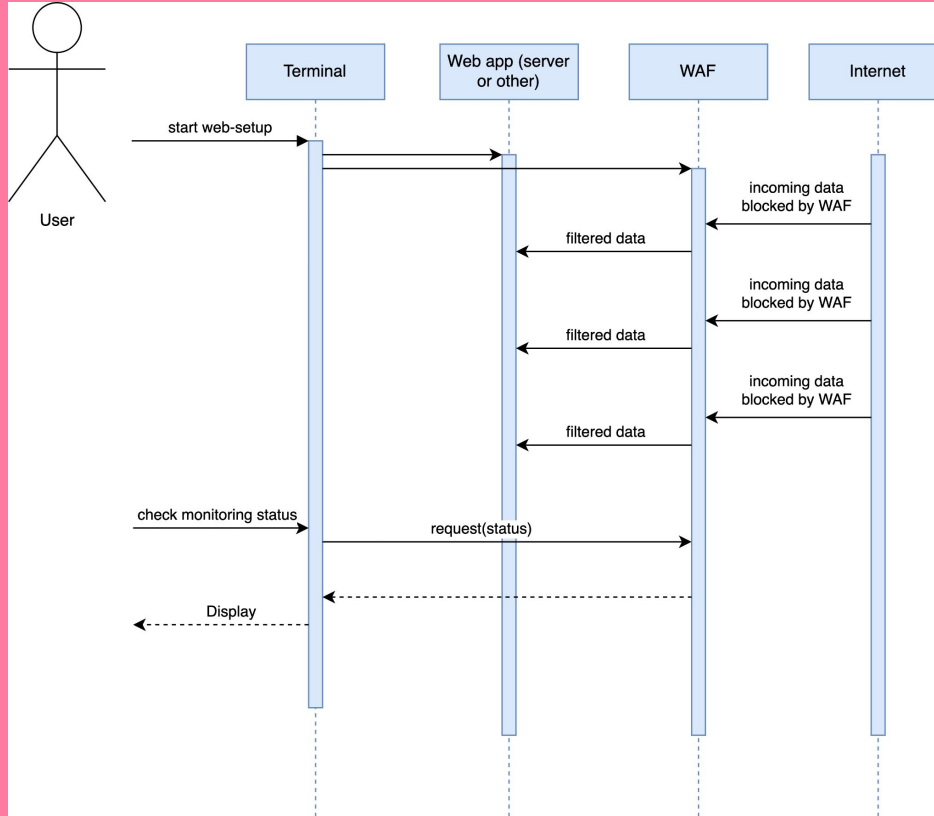
- Modify the kernel with a module to include WAF functionality

Pros

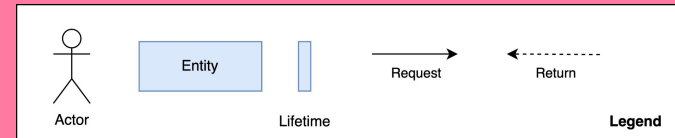
- Lower overhead: Runs in kernel space with direct access to hardware
- Integration with the kernel can provide fine-grained control
- Can be tricky for attackers to tamper with: loaded into protected kernel space
- Available to all userland applications without additional setup

Cons

- Requires specialized knowledge: more difficult to develop and maintain
- Vulnerabilities can impact other subsystem: requires intensive testing
- Requires a system reboot to patch/update, disrupting system operations



Use case: Kernel Module WAF





Userland Application

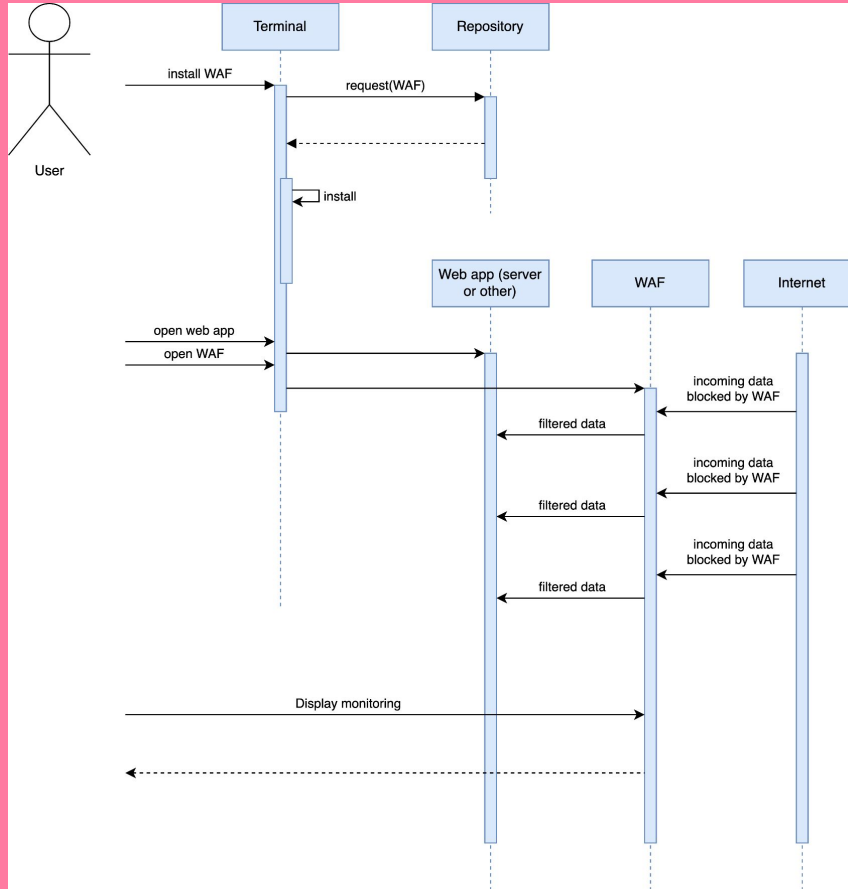
- Develop a userland application providing WAF functionality on top of the OS
- Runs outside the kernel

Pros

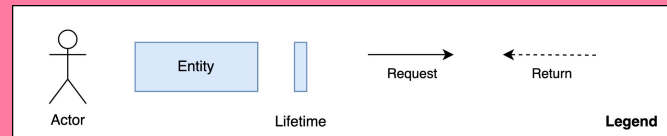
- Simpler to develop and maintain a userland application compared to a kernel module
- Risk of causing system crashes is reduced compared to kernel modules, as it runs in a protected environment
- Easier to patch/update without requiring a full system reboot

Cons

- Not as efficient in terms of performance and system utilization: Less direct access
- Subject to certain risks such as buffer overflows
- Limitations to WAF functionality in terms of advanced network security features



Use case: Userland WAF





Stakeholders

- **FreeBSD Community**
 - Primary stakeholder for any changes to the FreeBSD kernel
 - Responsible for reviewing/testing changes in code
- **Security Professionals**
 - Experts part of the community can identify potential risks/vulnerabilities
 - Provide recommendations in the development
- **System Administrators, Web Developers and the Average User**
 - As the end users, they would be configuring and maintaining
 - Provide feedback on usability and performance
- **Third-party Vendors**
 - Interest in integrating their solutions with the WAF module
 - Provide feedback on improvement for better support



Kernel Module

- A kernel module is developed to provide WAF functionality directly in the kernel.
- Two Approaches as a Kernel Module:
 - **Dedicated Subsystem**
 - **Integration within Existing Subsystem**
- Rationale:
 - Avoid major changes to overall architecture
 - Limit the impact on other interactions and dependency relationships
 - Does not call for its own subsystem within the current architecture



Kernel Module

- Part of the **Network** subsystem
 - Closer to the TCP/IP stack
 - Inspects and filters network traffic
 - Allows function at a lower level:
Potentially providing better performance and lower latency
- Alternative:
 - Part of the **Security** subsystem
 - Although responsible for enforcing security policies, there might be a lack of necessary network stack knowledge to effectively filter traffic.



Kernel Module: Proposed Implementation

- **Main Module**

- Kernel module to provide a low-level interface for intercepting network traffic at the network stack level. Responsible for capturing network traffic and passing it to the WAF engine for analysis.

- **WAF Engine**

- Responsible for analyzing incoming network traffic and determining whether it is malicious or not. Use a set of rules to determine whether a request is allowed or blocked. Highly configurable to allow for fine-grained control over the rules and policies applied.

- **Rule Management**

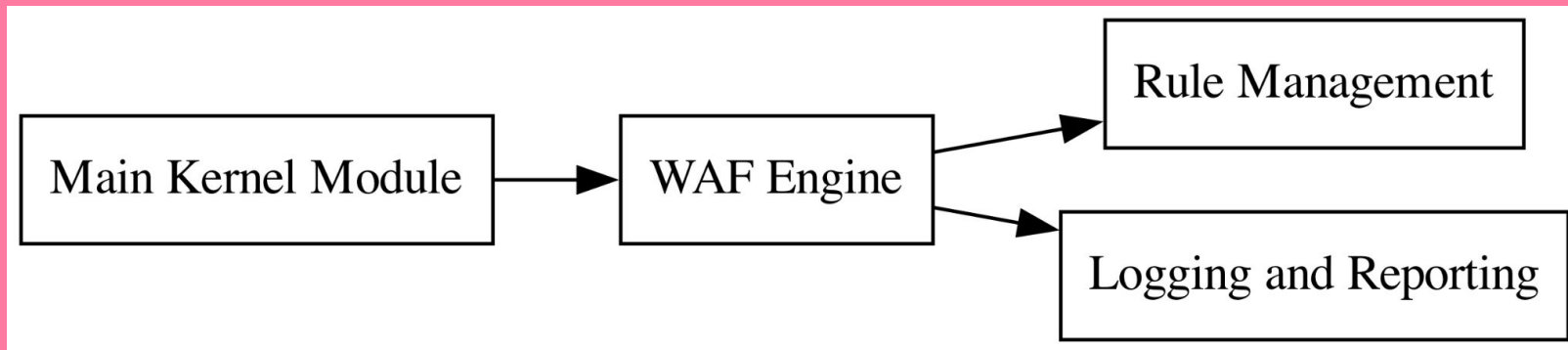
- Responsible for managing the rules used by the engine. Allow user-friendly creation/modification and enable/disable rules.

- **Logging & Reporting**

- Responsible for collecting and storing logs generated by the engine. The reporting system would allow users to view and analyze these logs in a user-friendly format.



Kernel Module: Proposed Implementation





Kernel Module: Design Guidance

- **pfSense firewall:**
 - Specialized OS designed specifically for network security
 - Includes WAF as part of their core features
 - Distribution is based on FreeBSD
- Design decisions may not be directly applicable
- However, we can gain valuable insight into:
 - Design considerations
 - Potential Challenges





Potential Impact on Subsystems

Network Subsystem

- WAF handles web traffic interception, processing and filtering
- Needs to interact with TCP/IP, sockets and network interfaces

Security Subsystem

- Interacts with security architecture
- Access control, data protection, policy enforcement

Process Management Subsystem

- Assists with handling of WAF processes and threads when extending the functionality of Security



Potential Impact on Subsystems

Memory Subsystem

- Allocate memory as necessary (eg. manage security rules, store temporary data, processing web traffic)
- Caching user authentication data or web content
- Managing WAF buffers with web traffic data

System I/O Subsystem

- Hardware-level network I/O operations



Effects on Concurrency

There will be no significant effects on concurrency.

- Since freebsd has a locking system(lockf), so the concurrency can be guaranteed. The reason is that locking system can lock the file when the file is being processed.
- Each process will not influence each other, the mutex lock can make sure that when one process is processing the other processes will not influence the processing process.
- There might be some problems that may impact the concurrency, for example some unexpected conflict between new subsystem with the old subsystem may have a bad impact.



Testing the impact of interactions between the WAF and other features

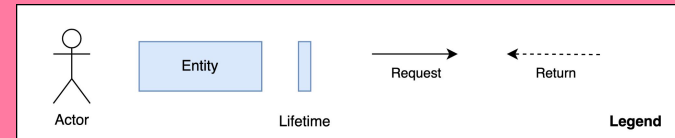
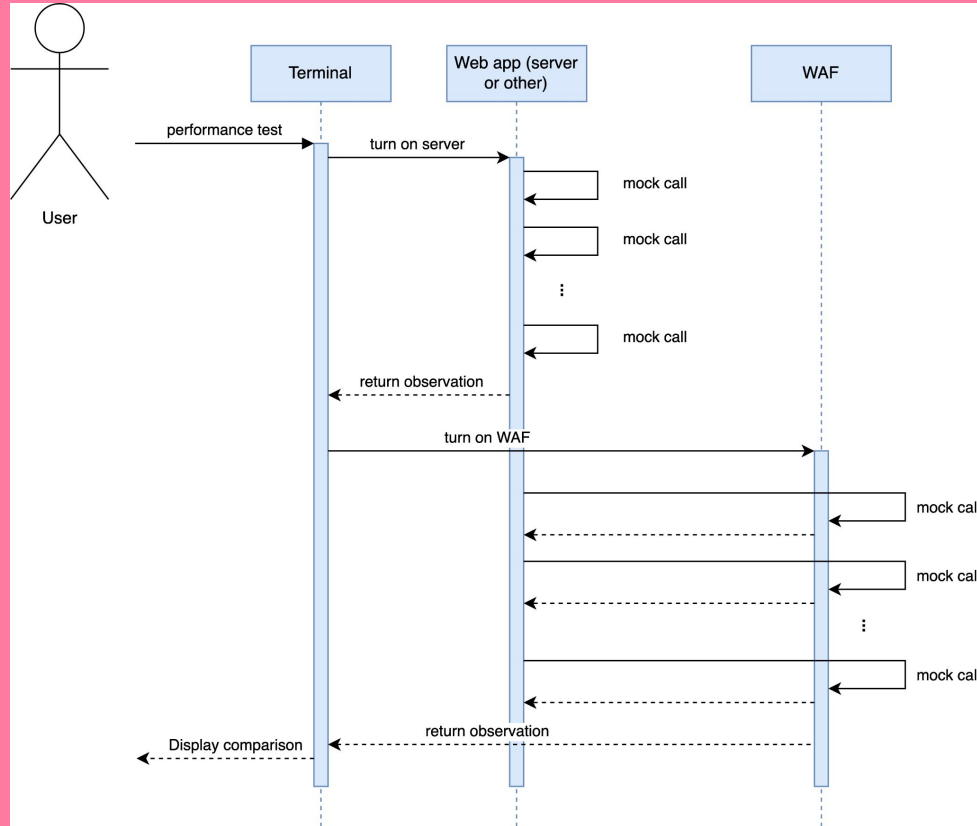
- Security Subsystem
 - Penetration testing : simulation of attacks against web applications and services protected by the WAF.
 - Vulnerability Scanning : Using vulnerability scanning tools to identify vulnerabilities in the subsystem.
 - Event Logging and Analysis
- Network Subsystem
 - Network Traffic Testing : measuring the impact of the WAF on network traffic throughput, latency, and packet loss
 - Protocol Testing
 - Load Testing
- Process Management Subsystem
 - Resource Usage Testing : benchmarking tests to measure the resource usage of web traffic with and without the WAF enabled.
 - Process monitoring and analysis



Testing the impact of interactions between the WAF and other features

- System I/O subsystem
 - I/O Performance testing
 - Error handling testing
- Memory Subsystem
 - Memory usage testing
 - Memory leak testing

Use case: Performance Test



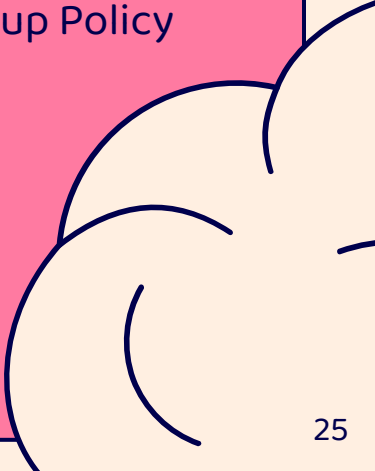


Risks and Limitations

- Risk involved in implementing the feature are:
 - Performance Impact
 - Generate false positive
 - Configuration complexity
 - Attackers could evade
- Limitations of the feature are:
 - Not a Substitute for Securing Coding
 - Limited Protection
 - Resource Constraints
 - Compatibility issue



Lessons Learned

- Understanding of stakeholders and their concerns
 - Learning how to make decisions by considering multiple approaches and evaluating the impact on subsystems, performance, and maintenance
 - Learned how to evaluate resource requirements
 - Learned how to anticipate challenges, risks, and limitations
 - Making revisions to the proposal based on research about tradeoffs (eg. previous enhancement idea was adding Active Directory and Group Policy but found the community agrees on using LDAP/Samba)
- 



Conclusion

- By comparing the two approaches we list, we find that kernel module can be more efficient, since kernel module can be more efficient and be available for all the users.
- Since it is at the kernel level, so it will be easier to interact with the other subsystem
- There are still some limitations such as the conflict with other subsystem.



References

1. <https://cgit.freebsd.org/src/tree>
2. <https://docs.freebsd.org/en/books/arch-handbook/book/>
3. *Marshall Kirk McKusick, George V. Neville-Neil, Robert N.M. Watson (2015)*
The Design and Implementation of the FreeBSD® Operating System, 2 e.d.
4. <https://web.cs.ucla.edu/classes/spring16/cs111/supp/dynamicmodules.html>
5. <https://www.cloudflare.com/en-ca/learning/ddos/glossary/web-application-firewall-waf/>
6. <https://www.pfsense.org/>

