DefCamp Cluj-Napoca

Dev Ally, Zero-Days Foe

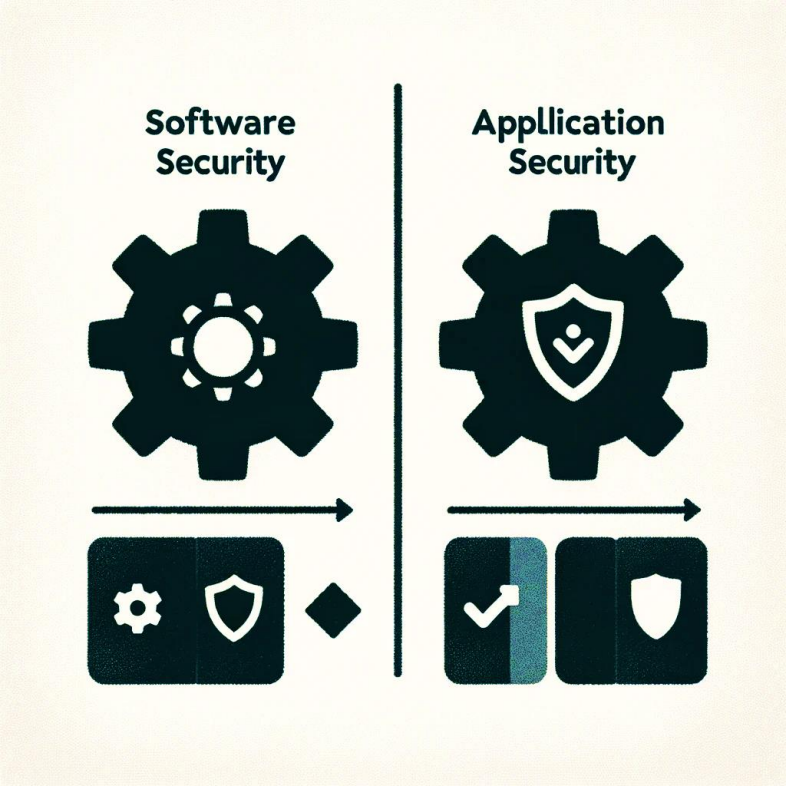
Sponsors…

About me

- 12 years in the field!
- Application Security Lead @ Canon EMEA
- Author and reviewer
- A regular speaker at industry conferences e.g. DefCon3x, Security Bsides6x, Confidence, LeHack, Hacktivity, OWASP global AppSec, IEEE AI/ML, NoNameCon, COSAC, c0c0n, ISACA Euro CACS/CSX and …
- Lifelong learner!!!

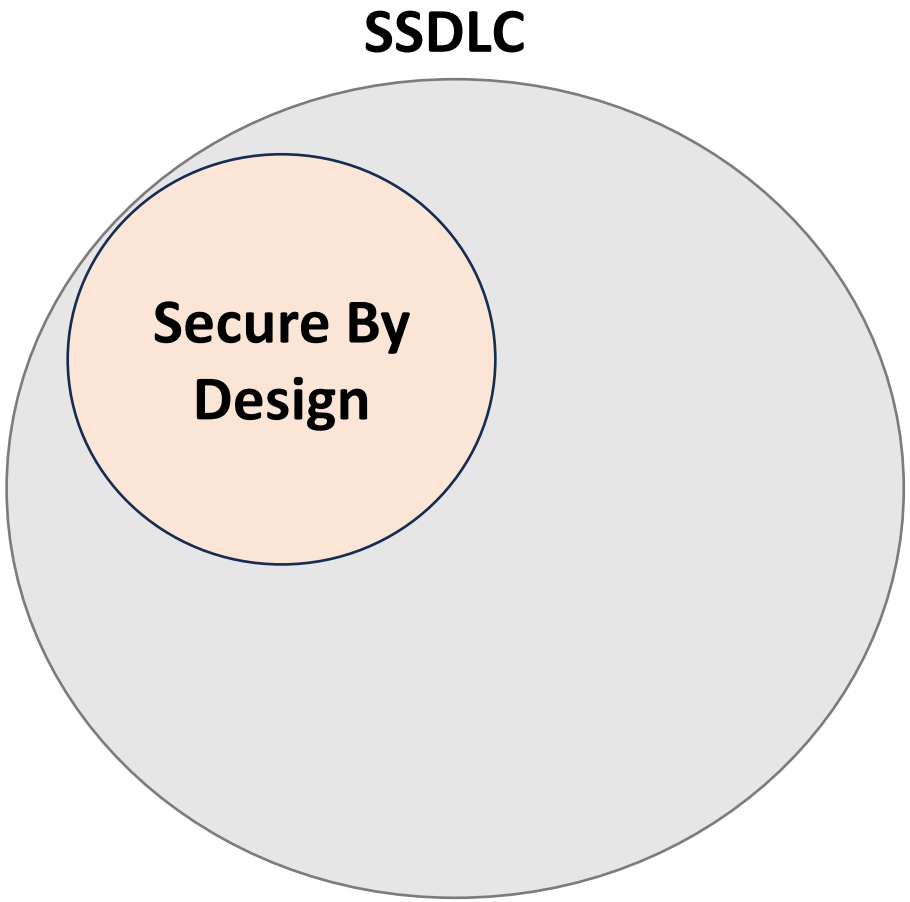
AppSec

- Software security covers full development lifecycle.
- Application security focuses on post-deployment.
- Software security includes pre-deployment practices.
- Application security involves post-deployment activities.



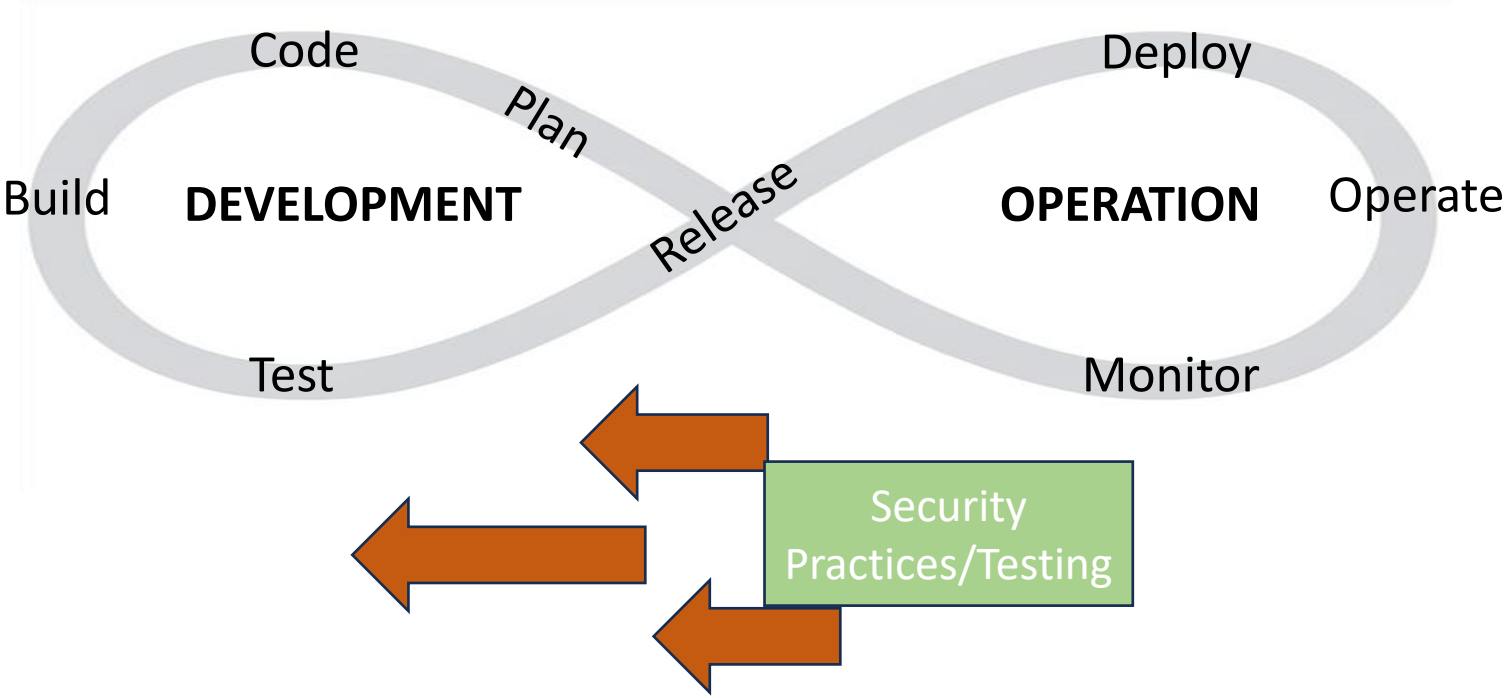
The ecosystem

- SSDLC
- Secure By Design
- SAST/DAST…
- …
- …



Appsec ecosystem

- Shifting left !



Appsec ecosystem

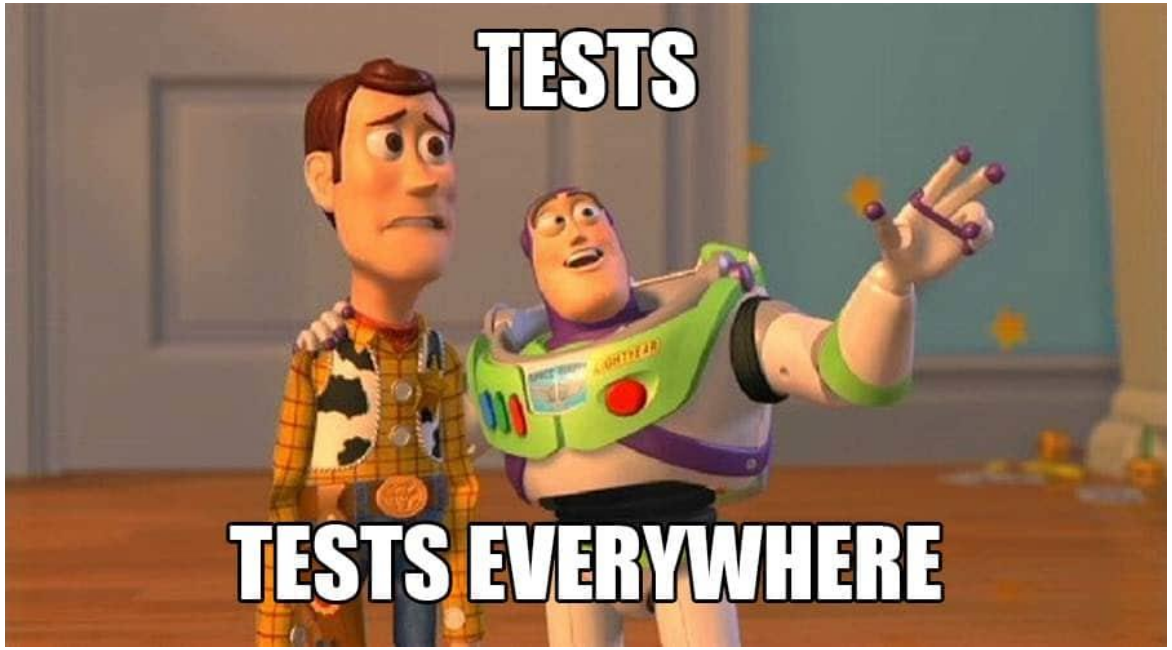
Continuous Improvement

- Vulnerability Management outcomes
- Hardening
- Annual Pentests
- Periodic Security Assessments

Appsec ecosystem

Security testing

- **Automated**
  - **SAST**
  - **DAST**
  - **IAST**
  - **SCA**
  - **???**
- **Manual**
  - **Code Review**
  - **PT**



https://mailtrap.io/blog/qa-testing-memes/

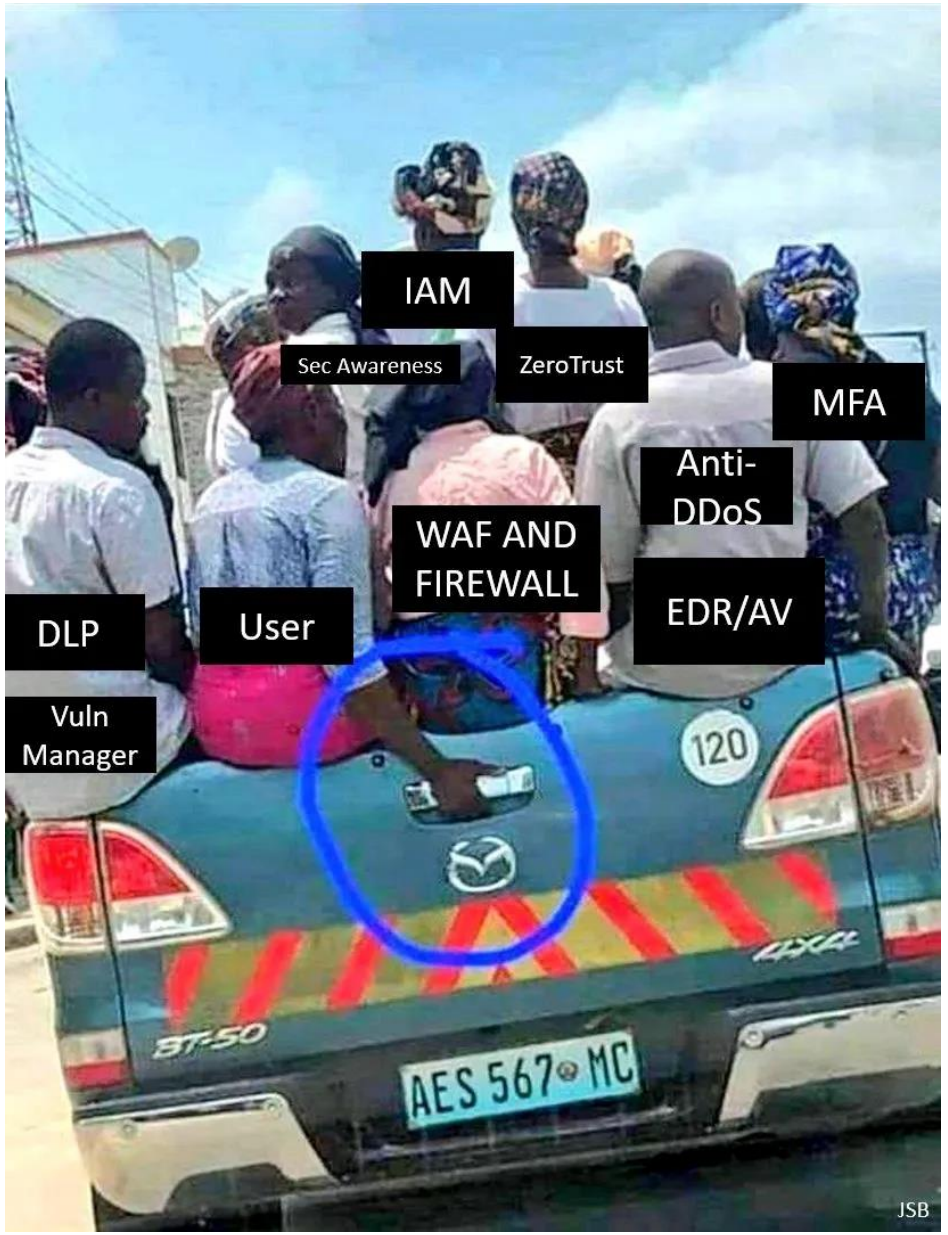
Appsec ecosystem

- Inventory of all libraries, dependencies, etc. => SBOM

- Identify and scan all pf open source components => SCA

Appsec ecosystem

- Security training and awareness
  - You need it!



https://twitter.com/ItKraftz/status/1623155209114644480

Tooling disadvantages

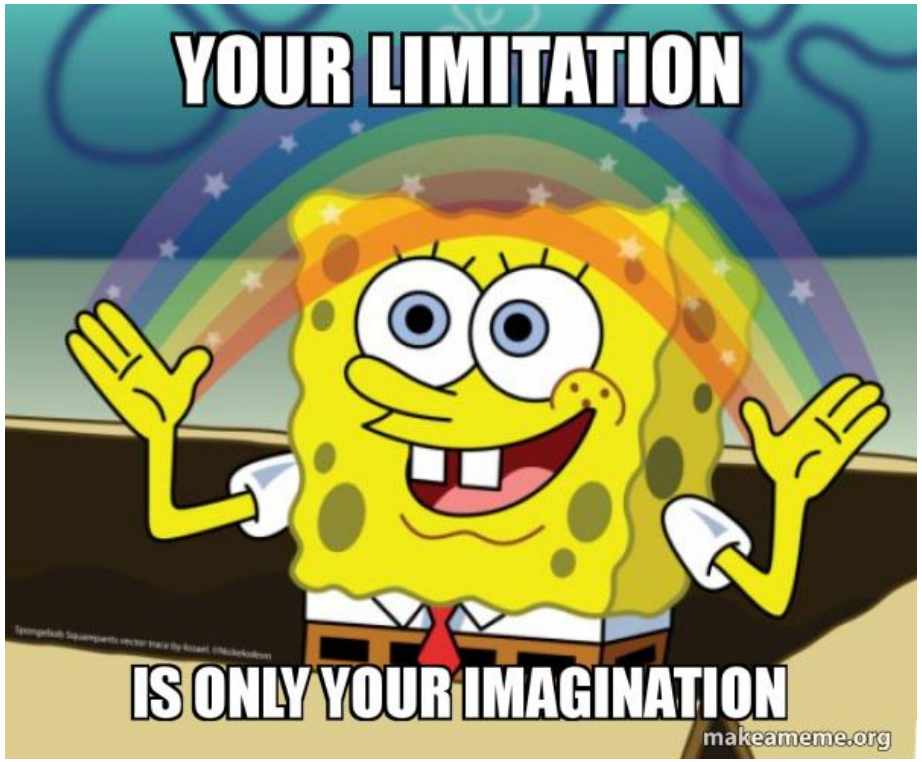
- Cost
- Limited to predefined patterns and signature
- High number of FP
- AI is just a fancy button (fun pic here!)
- Integration difficulties

https://giphy.com/stickers/meme-monkey-spiderlex-5vaSWRP4iRf8IrMs5l


Is that fair enough?!
Made with Piñata Farms

No!

SAST/DAST/IAST = Limitations



SAST limitations

- Unable to Detect Runtime and Environmental Vulnerabilities (Obvious)
- Limited Coverage of Modern Attack Vectors (lack of API security check, not really updated for new tech stacks and frameworks)
- High False Positive and Negative Rates (FP or Noise and FN or lack of detection)
- Scalability and performance issues (large codebases, Incremental scanning is still an issue)

DAST limitations

- Unable to Detect Runtime and Environmental Vulnerabilities (Obvious)
- Limited Coverage of Modern Attack Vectors (lack of API security check, not really updated for new tech stacks and frameworks)
- High False Positive and Negative Rates (FP or Noise and FN or lack of detection)
- Scalability and performance issues (large codebases, Incremental scanning is still an issue)

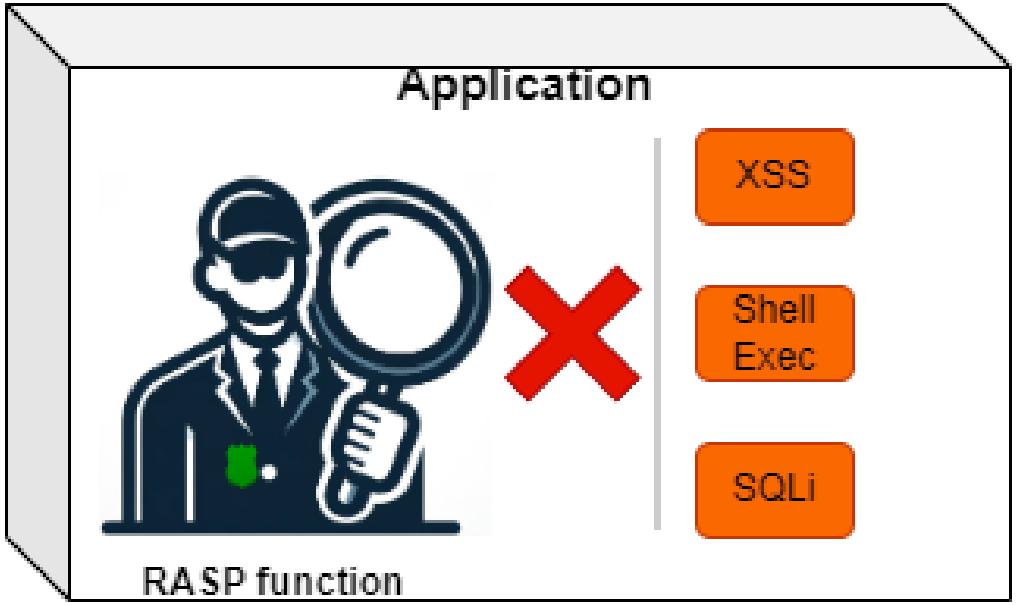
IAST limitations

- Dependence on Real Traffic [IAST's effectiveness often depends on the application encountering real or simulated traffic that covers all aspects of its functionality. Inadequate traffic can lead to incomplete testing and missed vulnerabilities.]
- Cost Issues [IAST tools can be expensive to implement and maintain. The costs not only include licensing fees but also the resources needed to manage and operate the tool within the development lifecycle.]
- Performance overhead [Running IAST tools can introduce performance overhead on the application being tested, potentially affecting its responsiveness and increasing resource usage, which can be problematic for production environments.]

RASP

- What is rasp

RASP embeds in applications to instantly detect and block threats during runtime by understanding the app's logic, effectively protecting against vulnerabilities with precision.



RASP benefits

- **Real-Time Protection:** Like a vigilant guardian within your app, ready to fend off attackers on the spot.
- **Flexibility:** Adapts like a chameleon, fitting perfectly into various environments and requirements without hassle.
- **DevSecOps and SSDLC Friendly:** Acts as a supportive team player in development and security processes, making sure security is baked in from the start.
- **Automatic Security Updates:** Provides your app with a steady flow of health checks and defense enhancements, seamlessly.
- **Compliance Friendly:** Keeps a detailed log of security measures, simplifying regulatory compliance.
- **Insightful Reporting:** Offers detailed analysis of security incidents, akin to having a personal security analyst explaining what went wrong and how.

RASP vs SAST/DAST/IAST

- Protects applications in real-time by working from within.
- Automatically responds to threats without needing application modifications.
- **Comparative Analysis**
- **SAST**: Pre-deployment, source code analysis. Does not offer real-time protection.
- **DAST**: Post-deployment, external testing. Ident

- SAST/DAST/IAST focus on pre/post-deployment testing, identifying vulnerabilities without the capability to respond in real-time.

RASP

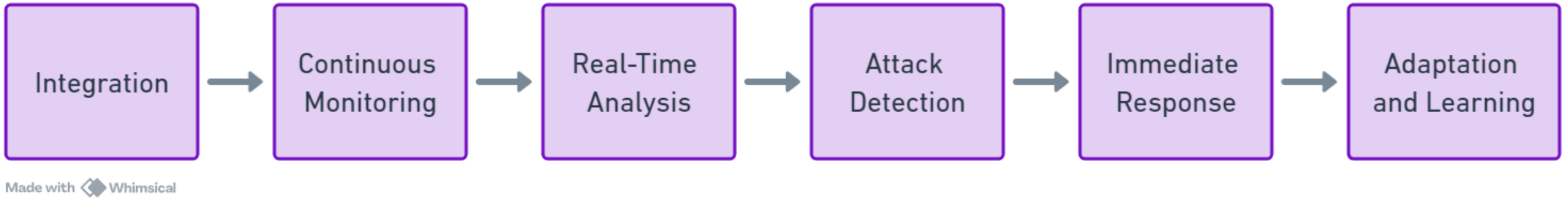
- Is it still proactive or reactive…!?!?!?!??!

https://i.imgflip.com/59f53e.png

RASP approach

- [RASP (Runtime Application Self-Protection)** is a security technology that integrates with an application or its runtime environment to control application execution and detect and prevent real-time attacks. Its approach to identifying and mitigating security issues involves monitoring the application's behavior and context to make intelligent decisions about threats in real time.]

RASP Steps



1. RASP is integrated with the application or its runtime environment.

2. The application's operations and data flows are continuously monitored.

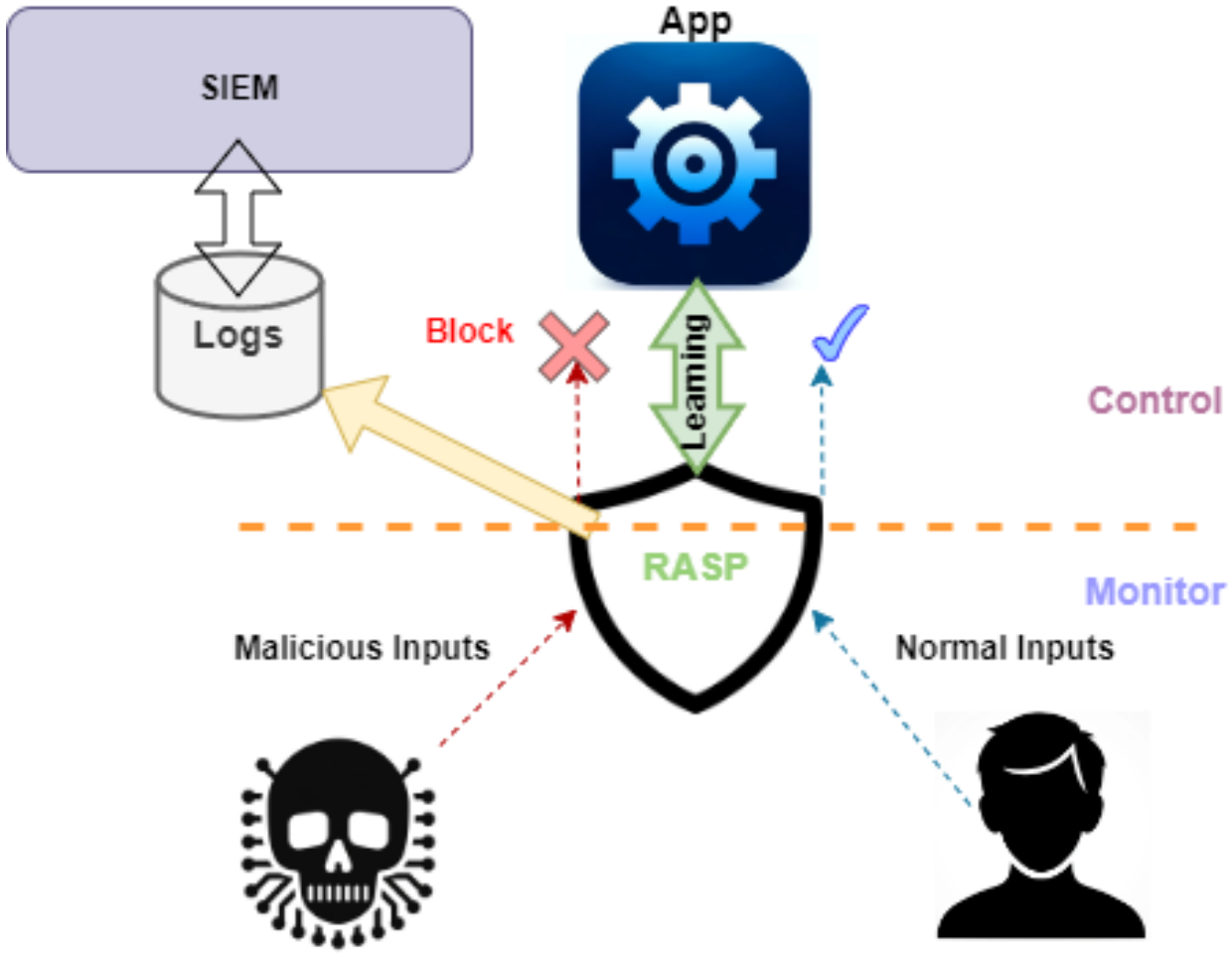
3. All requests and behaviors are analyzed in real time to assess for potential threats.

4. The system uses its understanding of normal application behavior to identify anomalies that may represent attacks.

5. Appropriate actions are taken automatically, such as session termination, input sanitization, or alerts to administrators.

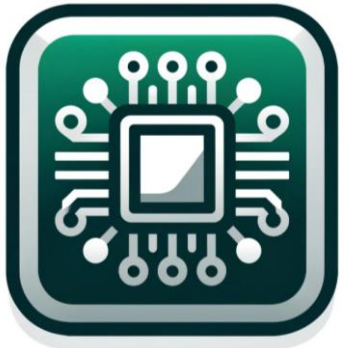
6. The system learns from past attacks to improve its threat detection and response strategies over time.

RASP Big Pic


SIEM
App
Logs
Block
Learning
Control
RASP
Monitor
Malicious Inputs
Normal Inputs

Rasp for all seasons

App Server



Embedded agents

IIS

NGINX
Part of F5

Server Plugins/Add-ons:

modsecurity
Open Source Web Application Firewall

Non-Web App/Desktop

Local agent

Background services

Mobile Apps

iOS

Cloud Native

Microservices

IOT

Rasp for all seasons cont.

IoT

Embedded in firmware

APIs

API Gateway

Rasp pre-deployment

- Application compatibility [Ensure the RASP solution is compatible with the programming languages and frameworks of your applications. Compatibility reduces integration issues and functional discrepancies.]
- Performance Benchmarks [Establish performance benchmarks to evaluate the impact of RASP on application response times and resource usage. RASP tools should not significantly degrade the performance of the application they protect.]
- Operational Integration [Assess the ease of integrating RASP solutions with existing CI/CD pipelines, DevOps practices, and application deployment strategies.]

Rasp pre-deployment

- Approach and components

RASP techniques

ACTIVE or **PASSIVE**

PROTECTION or **MONITORING**

Active mode

- High application resources usage
- Expected latency and performance impact
- Real-time detection and prevention

Passive mode

- Limited application resources usage
- Low latency
- Generating logs
- Learning

RASP techniques

- Traffic Analysis => DPI, signature-based detection
- Behavioral Analysis => Historical analysis through application baselines
- Anomaly Detection/Heuristics => machine learning algorithms and predefined models
- Signature Analysis => Known payloads and attack patterns

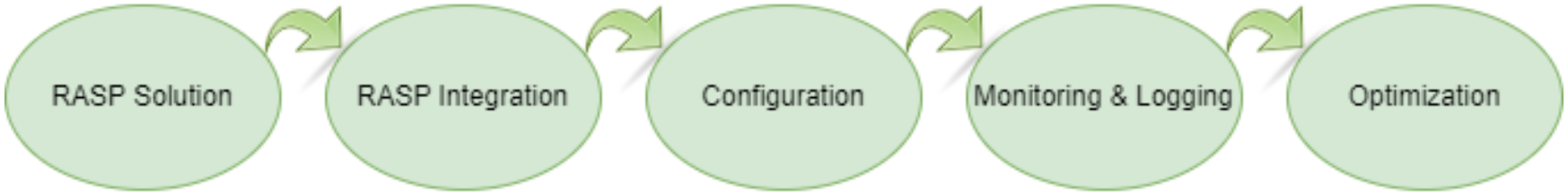
RASP techniques cont.

- Code Analysis!!!? => What about runtime!?, Bytecode analysis

- Memory Analysis => Inspection of memory allocations and accesses BoF, heap manipulations, and…

RASP in action

- RASP implementations



RASP in action

- RASP integrations
  - Compatibility Assessment
  - Performance Impact Assessment
  - False Positive/Negative monitoring
  - Policy Optimization
  - Alerts Handling
  - Compliance and Reporting

Real world examples

- https://github.com/talsec/Free-RASP-Community
- https://github.com/baidu/openrasp
- https://github.com/paraxialio/exploit_guard

RASP VS IAST

Feature	IAST	RASP
Functionality	Security testing tool	Security prevention tool
Detection	Detects security vulnerabilities	Detects security threats and attacks
Operation	Analyzes application during runtime	Integrates within applications to monitor runtime
Integration	Typically used in development/testing phase	Integrated within production applications
Focus	Focuses on identifying potential weaknesses	Focuses on actively protecting applications

RASP vs WAF

Feature	WAF	RASP
Integration	Perimeter security (Maybe 1st layer)	Last layer of defense
Operation	Alerting based on patterns by HTTP(S) requests inspection	Alerts and blocks known and potential threats by App behavior inspection
Attack Handling	Blocks attacks with existing rules	Blocks both known attacks and zero-day attacks
Automation	Requires manual rule configuration	Operates without human intervention
Security Approach	Reactive; based on known payloads	Proactive and comprehensive; handles unknown threats

RASP vs WAF

- Log4J is not a good example and I think WAF is a better solution since it deployed at the network perimeter.
- consider a sophisticated attack that uses legitimate application functionality in an unintended way to perform unauthorized actions, such as manipulating internal API calls or abusing application logic to escalate privileges. RASP, by understanding the application's normal behavior and context, can detect and block such misuse in real-time, even when specific attack signatures are not yet known or are too complex for a WAF to discern.

Conclusion

References