Abertay University

A Taxonomy of Approaches for Integrating Attack Awareness in Applications

Tolga Ünlü,
Dr. Lynsay Shepherd,
Dr. Natalie Coull,
Colin McLean

Introduction - Tolga Ünlü

- 1. Year PhD Student @ Abertay University, Scotland
  Supervisors: Dr. Lynsay Shepherd, Dr. Natalie Coull, Colin McLean

- PhD Research Project:
  Investigating Attack Awareness within Web Applications

- Research Interests:
  Application Security, Usable Security for Developers,
  Deception Technology

Agenda

- Problem Statement
- Attack Aware Applications
- Integration Approaches
  - Developer-Driven
  - Agent-Driven
- Discussion
- Conclusion & Future Work

The Security Blind Spots of Applications

Applications are often built without a means of **observing and reacting to security events as they occur.** [1][2]

This has the following consequences for applications that are blind towards security events:

- Attackers probing as they wish → Finding exploitable vulnerabilities
- In Production: No measure of effectivity of security controls
- In Production: No measure of validity of the threat model
- Incident Response: Missing forensic evidence

Attack-Aware Applications

Attack-aware applications detect and respond to attacker activities in real-time through **embedded detectors** [3] or **detection points** [1].

Detectors: Security controls that check for **indicators of attacker activity.**

```
if(attack_indicator){
    log(“Attacker activity detected!”);
    respond();
}
```

Attack-Aware Applications

The **application context** can be utilized to define a set of observable attack indicators for application-level intrusion detection [4].

In the current context:

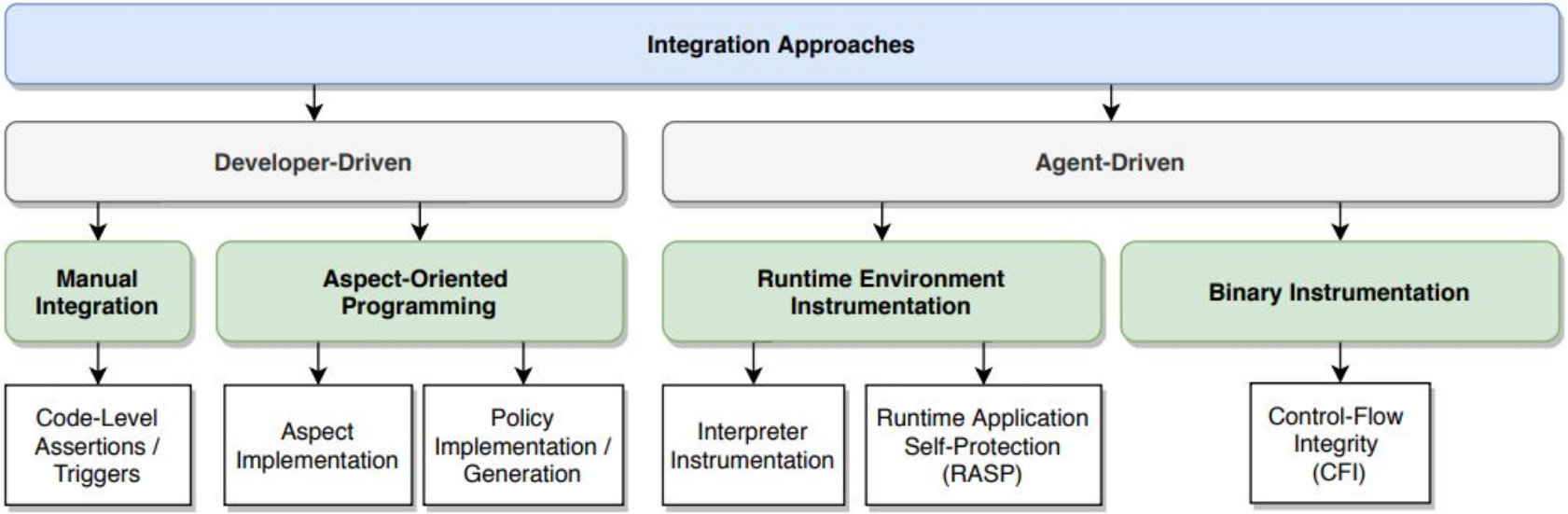
- What actions are possible?
- Which values can a user provide?
- What is the expected exec. order of actions?
- Should this action be executed at all?
- Which user roles are required for the actions?
- …

Determine and Monitor
Security Invariants

“X must always be true/false”

Approaches for Attack Awareness Integration

Guidance for researchers and developers to determine the appropriate solution based on their technical and usability requirements



Developer-Driven Integration

The integration of attack awareness is done manually by the developers of an application

Manual Integration
Detectors are directly implemented in the application code

Aspect-Oriented Programming
Detectors are implemented as “aspects”
→ Run aspect before/after function of interest @ runtime

+ Utilization of Application Expertise and Frameworks
+ Business Logic Attack/Probing Detection
+ Usable Security Control Format

- Additional Task for Developers
- Security Expertise Required for Certain Attacks (e.g. Injection Attacks)
- Manual / Limited Automation

Agent-Driven Integration

The integration of attack awareness is done automatically by a software agent on behalf of the developer

Runtime Environment Instrumentation
Software agent is part of the runtime environment → Affects all running applications

Binary Instrumentation
Software agent injects detectors into an applications binary code

+ Low Setup Cost (Plug & Play)
+ Automatic Injection Attack Detection
+ No Code Modification Required

- Inadequate Detection Techniques
- Platform/Technology Specific
- Inadequate in Certain Environments

Discussion

Detectors for business logic attacks and probing behavior need to be manually implemented due to their custom nature.

→ **Detecting a few distinct attacker probes could be sufficient to mitigate further attacks**

Detectors for these:

- Are a few lines of code at most (including response logic)
- Don’t introduce significant complexity
- Are performant as they execute only when attackers run into them

But requires **manual development** and is an **additional task on top of others.** [5]

Conclusion & Future Research

Attack awareness can be integrated in applications using a developer-driven or agent-driven approach.

Further research will focus on reducing the integration effort and aligning the integration with common practices.

Utilizing Application Frameworks and their Components

→ Form the Basis of many Applications
→ Reusable Components for Common Practices (e.g. Integrating Attack Awareness via Dependency Injection [6])
→ Mitigations within the Framework increase Applications Security [7]
→ Frictionless for Developers

Thank you!

Contact, Feedback, Collaboration:
tolgadevsec.github.io

References

[**1**] C. Watson, M. Coates, J. Melton, and D. Groves. *Creating Attack-Aware Software Applications with Real-Time Defenses*. 24:14–18, 2011.

[**2**] *A10:2017-Insufficient Monitoring and Logging | OWASP*, 2017. https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A10-Insufficient_Logging%252526Monitoring

[**3**] F. Kerschbaum, E. H. Spafford, and D. Zamboni. *Using Internal Sensors and Embedded Detectors for Intrusion Detection*. Journal of Computer Security, 10(1-2):23–70, 2002.

[**4**] R. Sielken and A. Jones. *Application Intrusion Detection Systems: The Next Step*. ACM Transactions on Information and System Security, 1999.

[**5**] C. Hall, L. Shepherd, and N. Coull. *BlackWatch: Increasing Attack Awareness within Web Applications*. Future Internet, 11(2):44, 2019.

References

[**6**] W. Kim, C. S. Moon, S. Chung, T. Escrig, and B. Endicott-Popovsky. *Scalable and Reusable Attack Aware Software*. In 2012 ASE/IEEE International Conference on BioMedical Computing (BioMedCom), pages 101–104. IEEE, 2012

[**7**] K. Peguero, N. Zhang, and X. Cheng. *An Empirical Study of the Framework Impact on the Security of JavaScript Web Applications*. In Companion Proceedings of the The Web Conference 2018, pages 753–758, 2018