Attack Graph Generation for Microservice Architecture

Authors:

Amjad Ibrahim, M.Sc.
Stevica Bozhinoski
Prof. Dr. Alexander Pretschner

Content

- Motivation
- Background
- Method
- Evaluation
- Related Work
- Conclusion

Content

- Motivation
- Background
- Method
- Evaluation
- Related Work
- Conclusion

Motivation

- Microservices are increasingly dominating the field of service systems
- Increase of utilizing third-party components
- Result in greater attack surface
- Need for an automatic detection of potential vulnerabilities

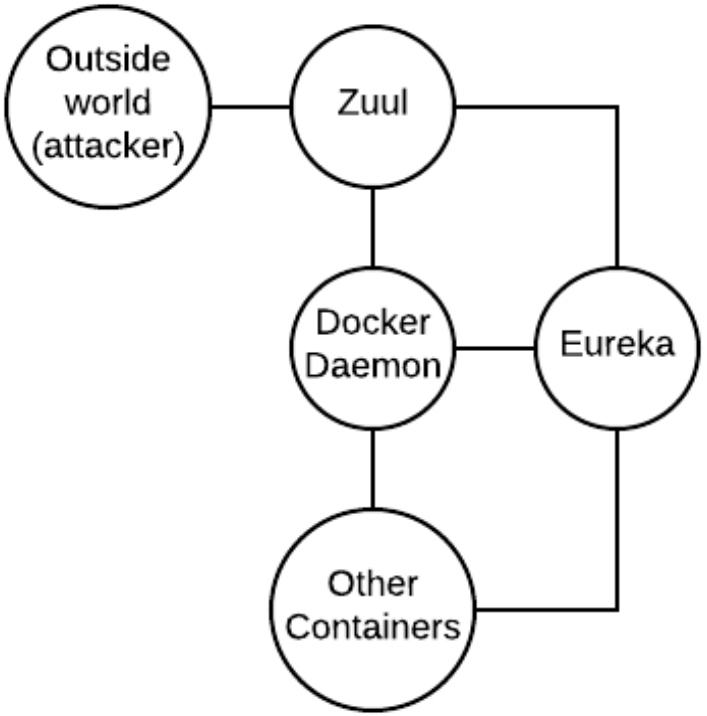
Content

- Motivation
- Background
- Method
- Evaluation
- Related Work
- Conclusion

Background

- Microservices
- Vulnerability Scanners
- Attack Graphs

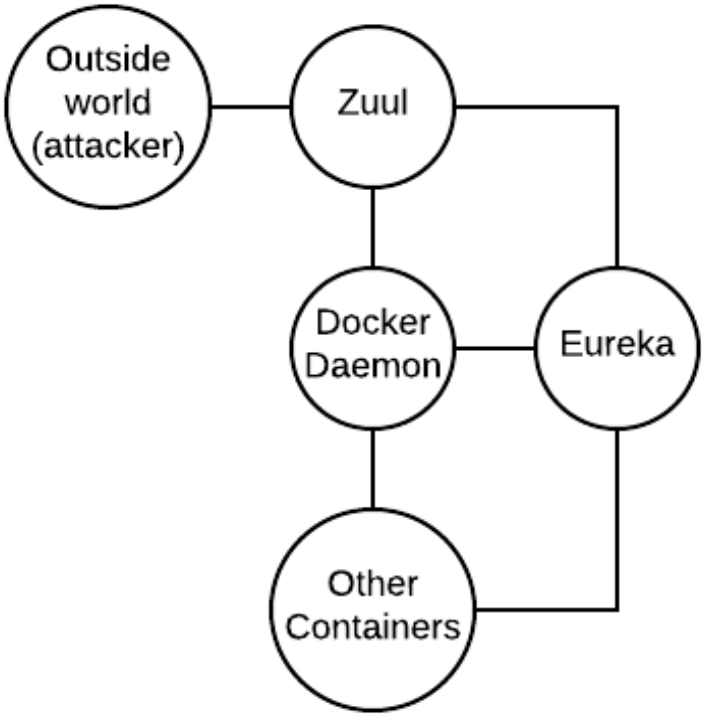
Example



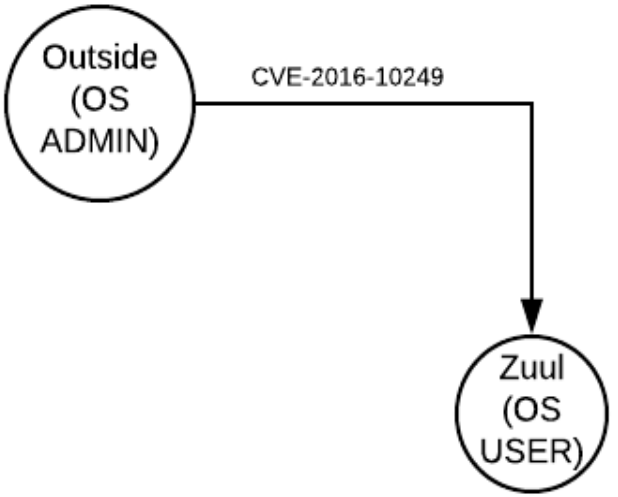
Topology

Resulting Attack Graph

Example

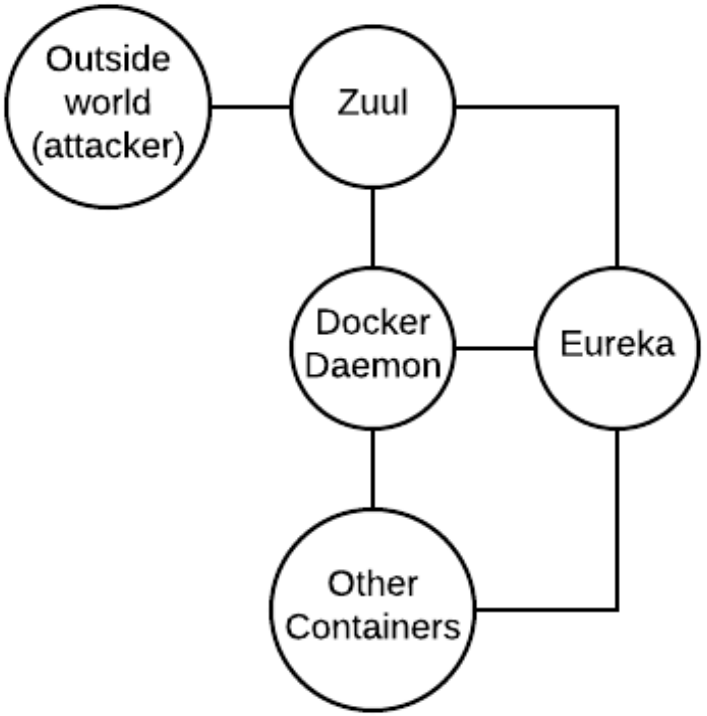


Topology

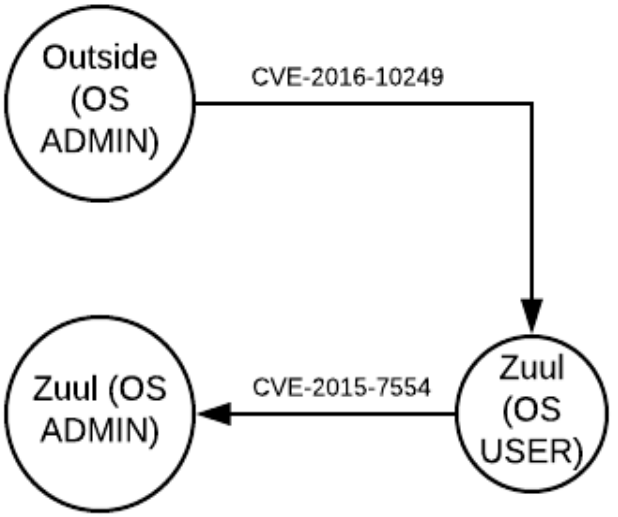


Resulting Attack Graph

Example

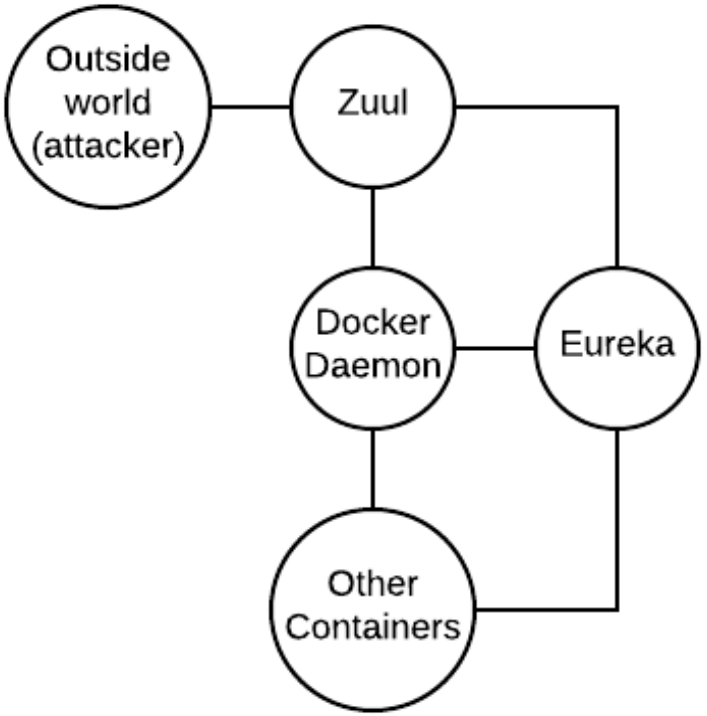


Topology

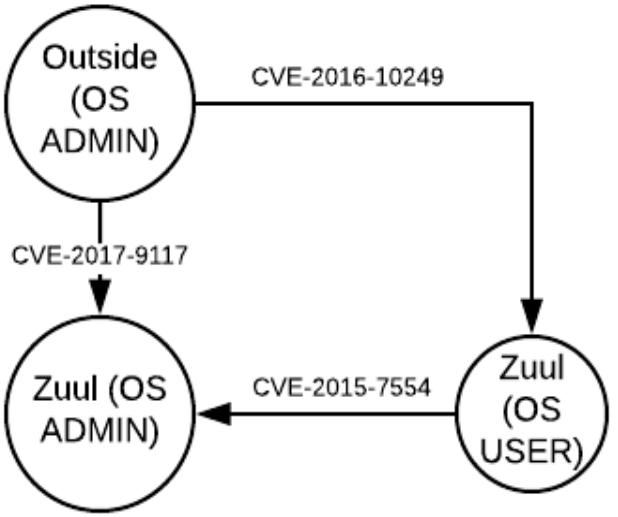


Resulting Attack Graph

Example

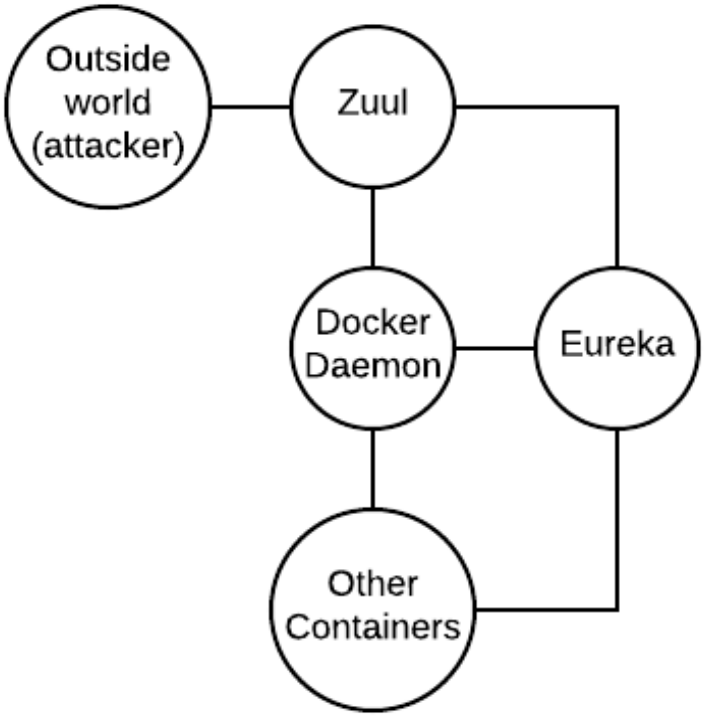


Topology

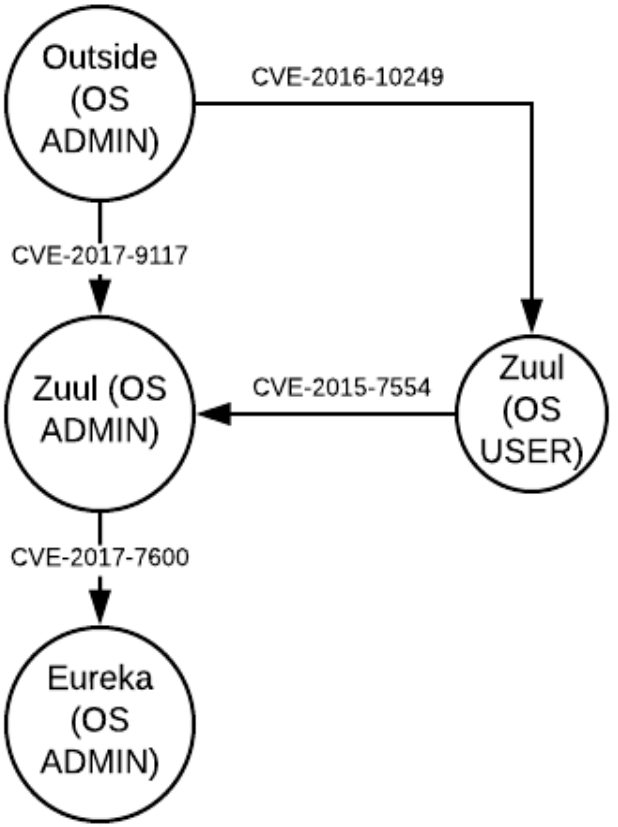


Resulting Attack Graph

Example

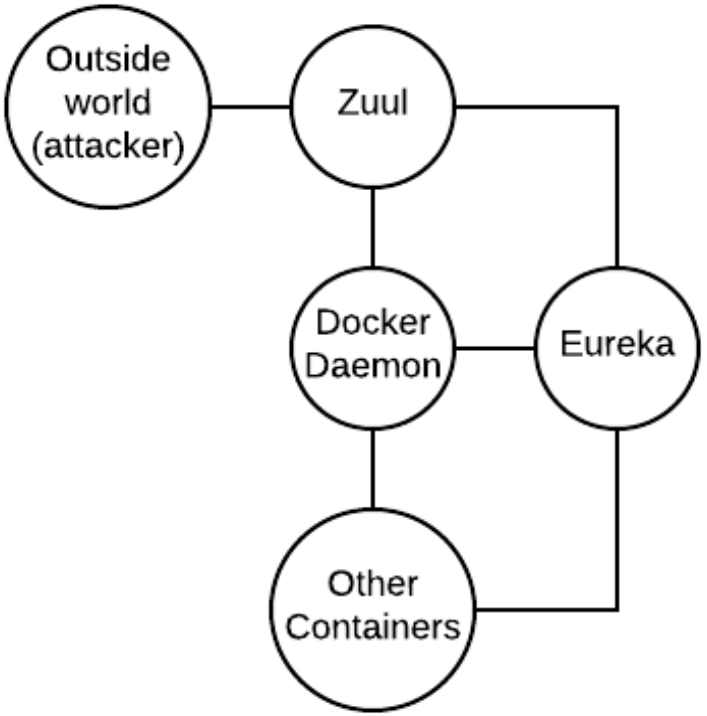


Topology

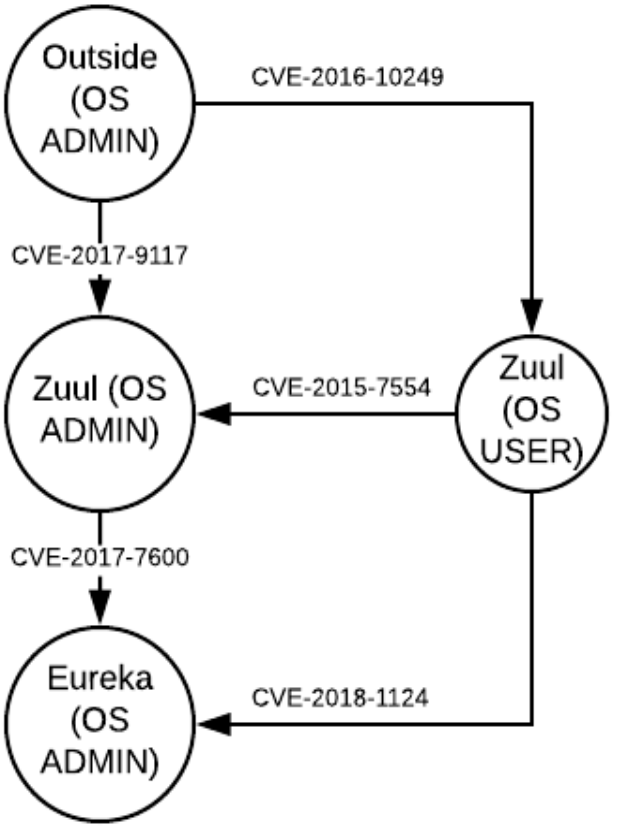


Resulting Attack Graph

Example



Topology



Resulting Attack Graph

Content

- Motivation
- Background
- Method
- Evaluation
- Related Work
- Conclusion

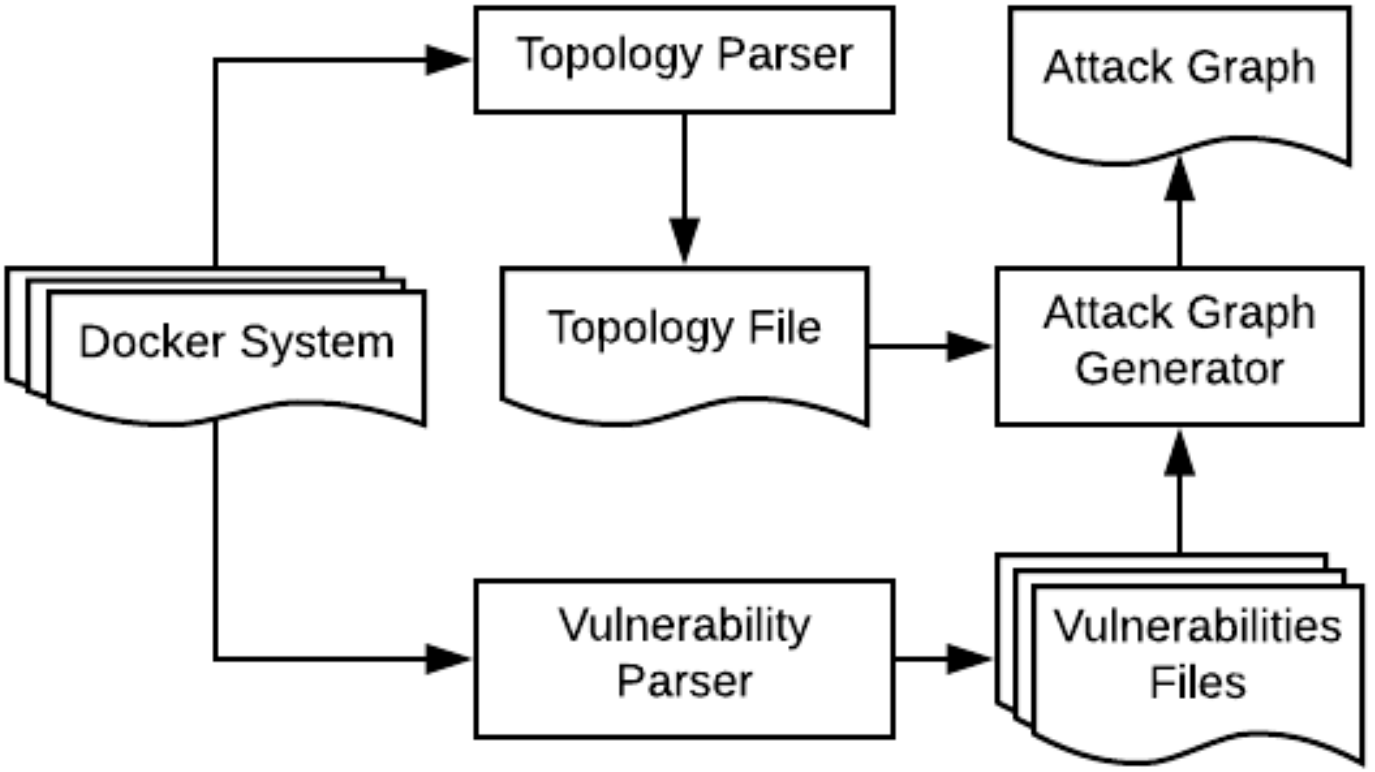
Method

- From Network Nodes to Microservices
- System Overview
- Attack Graph Generation for Docker Containers
  - Topology Parser
  - Vulnerability Parser
  - Attack Graph Generator

From Network Nodes to Microservices

- Adapt attack graph generator used in computer networks for microservices and map:
  - Hosts -> containers
  - Physical host link -> connection between containers
- Four level of privileges – combination of access level(User, Admin) and access mode (virtual macine, host)
- Preconditions/Postconditions from manually selected rules

System Overview



Topology Parser

- Utilizing information of the docker-compose.yml file
- Networks keyword – connection between components
- Published ports
- Privileged access

Vulnerability Parser

- Using Clair to generate vulnerabilities for each file
- Input: a given image
- Output: list of CVE-IDs, descriptions and attack vectors of the given image

Attack Graph Generator

- Input: list of vulnerabilities and topology
- Output: an attack graph
- Breadth-first search (BFS) algorithm
  - Monothonicity property
  - Termination
  - O(|N| + |E|) N – nodes, E – edges

Content

- Motivation
- Background
- Method
- Evaluation
- Related Work
- Conclusion

Evaluation

- Evaluation of the proposed attack graph generator system
  - Application on four systems with various technologies, number of containers and vulnerabilities
  - Testing the system scalability with increasing number of containers and connections

Use Cases

Name	Description	Technology Stack	No. Containers	No. vuln.	GitHub link
Netflix OSS	Combination of containers provided by Netflix.	Spring Cloud, Netflix Ribbon, Spring Cloud Netflix, Netflix's Eureka	10	4111	https://github.com/Oreste-Luci/netflix-oss-example
Atsea Sample Shop App	An example online store application.	Spring Boot, React, NGINX, PostgreSQL	4	120	https://github.com/dockersamples/atsea-sample-shop-app
JavaEE demo	An application for browsing movies along with other related functions.	Java EE application, React, Tomcat EE	2	149	https://github.com/dockersamples/javaee-demo
PHPMailer and Samba	An artificial example created from two separate containers. We use an augmented version for the scalability tests.	PHPMailer(email creation and transfer class for PHP), Samba(SMB/CIFS networking protocol)	2	548	https://github.com/opsxcq/exploit-CVE-2016-10033 https://github.com/opsxcq/exploit-CVE-2017-7494

Scalability

- Testing performance for increasing number of containers in fully-connected fashion
- 20, 50, 100, 500 and 1000 fully-connected containers
- Topology parsing and vulnerability preprocessing result in linear increase in time
- Attack generation time increases with number of connections
- 1000 fully-connected containers in 13 minutes

Content

- Motivation
- Background
- Method
- Evaluation
- Related Work
- Conclusion

Related Work

- Model Checker (Sheyner et al.)
  - Four hosts, eight atomic attacks in two hours
- Logical attack graph (Ou et al.)
  - 1000 fully-connected nodes in 1000 seconds
- Attack graph generator based on BFS (Ingols et al.)
  - 8901 nodes (23315 edges) in 0.5 seconds

Content

- Motivation
- Background
- Method
- Evaluation
- Related Work
- Conclusion

Conclusion

- Successful attack graph generation in microservice systems
  - Heterogeneous microservice systems with different technologies
  - Fully automated
  - Scalable solution for bigger systems

Thank you for your attention

- Questions?