ISSN: 2454-9940

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

E-Mail :
editor.ijasem@gmail.com
editor@ijasem.org

www.ijasem.org

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN2454-9940

www.ijasem.org

Vol 18, Issue 3, 2024

Checking Security Properties of Cloud Service REST APIs

[1]P MOUNIKA, [2]M.NAGA GAYATHRI

[1](Assistant Professor), MSC, **DANTULURI NARAYANA RAJU COLLEGE(A) PG COURSES, BHIMAVARAM ANDHRA PRADESH**

[2]MSC, scholar, **DANTULURI NARAYANA RAJU COLLEGE(A) PG COURSES, BHIMAVARAM ANDHRA PRADESH**

ABSTRACT:

Most modern cloud and web services are programmatically accessed through REST APIs. This paper discusses how an attacker might compromise a service by exploiting vulnerabilities in its REST API. We introduce four security rules that capture desirable properties of REST APIs and services. We then show how a stateful REST API fuzzer can be extended with active property checkers that automatically test and detect violations of these rules. We discuss how to implement such checkers in a modular and efficient way. Using these checkers, we found new bugs in several deployed production Azure and Office365 cloud services, and we discussed their security implications. All these bugs have been fixed.

1.INTRODUCTION

Cloud computing is exploding. Over the last few years, thousands of new cloud services have been deployed by cloud platform providers, like Amazon Web Services and Microsoft Azure , and by their customers who are “digitally transforming” their businesses by modernizing their processes while collecting and analyzing all kinds of new data. Today, most cloud services are programmatically accessed through REST APIs . REST APIs are implemented on top of the ubiquitous HTTP/S protocol, and offer a uniform way to create (PUT/POST), monitor (GET), manage (PUT/POST/PATCH) and delete (DELETE) cloud resources. Cloud service developers can document their REST APIs and generate sample client code by describing their APIs

ISSN2454-9940
www.ijasem.org
Vol 18, Issue 3, 2024

using an interface-description language such as Swagger (recently renamed OpenAPI) .

A Swagger specification describes how to access a cloud service through its REST API, including what requests the service can handle, what responses may be received, and the response format. How secure are all those APIs? Today, this question is still largely open. Tools for automatically testing cloud services via their REST APIs and checking whether these services are reliable and secure are still in their infancy. Some tools available for testing REST APIs capture live API traffic, and then parse, fuzz, and replay the traffic with the hope of finding bugs . Recently, stateful REST API fuzzing was proposed to specifically test more deeply services deployed behind REST APIs. Given a Swagger specification of a REST API, this approach automatically generates sequences of requests, instead of single requests.

2. EXISTING SYSTEM

Scanning of Swagger-based Representational State Transfer (REST) APIs - In addition to scanning Simple Object Access Protocol (SOAP) web services, Qualys WAS leverages the Swagger specification for testing REST APIs. Users need to only ensure the Swagger version 2.0 file (JSON format) is visible to the scanning service, and the APIs will automatically be tested for common application security flaws. - Enhanced API Scanning with Postman Support - Postman is a widely-used tool for functional testing of REST APIs. A Postman Collection is a file that can be exported from the tool that clubs together related requests (API endpoints) and shares them with other users. These collections are exported in JSON format. With the release of Postman Collection support in Qualys WAS, customers have the option to configure their API scans using the Postman Collection for their API.

DISADVANTAGES OF EXISTING SYSTEM:

- SOAP APIs are largely based and use only HTTP and XML.
- On other hand Soap API requires more resources and bandwidth as it needs
- to convert the data in XML which increases its payload and results in the large sized file.
- On other hand SOAP cannot make use of REST since SOAP is a protocol and REST is an architectural pattern.

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN2454-9940

www.ijasem.org

Vol 18, Issue 3, 2024

3. PROPOSED SYSTEM

REST APIs are implemented on top of the ubiquitous HTTP/S protocol, and offer a uniform way to create (PUT/POST), monitor (GET), manage (PUT/POST/PATCH) and delete (DELETE) cloud resources. Cloud service developers can document their REST APIs and generate sample client code by describing their APIs using an interface-description language such as Swagger (recently renamed OpenAPI) [25]. A Swagger specification describes how to access a cloud service through its REST API, including what requests the service can handle, what responses may be received, and the response format

ADVANTAGES OF PROPOSED SYSTEM:

- REST APIs are usually simple to build and adapt.
- With the initial URI, the client does not require routing information.

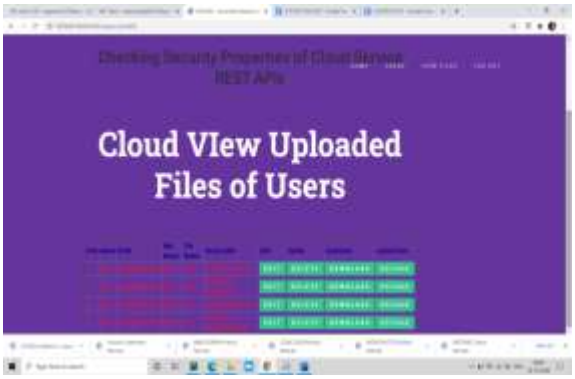
4. OUTPUT SCREEN

Home Page:

User Login Page:

User details:

View-files:



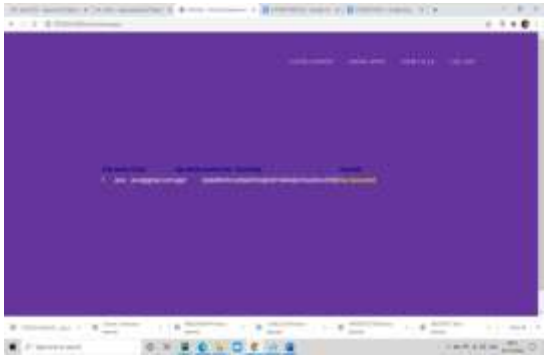
INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

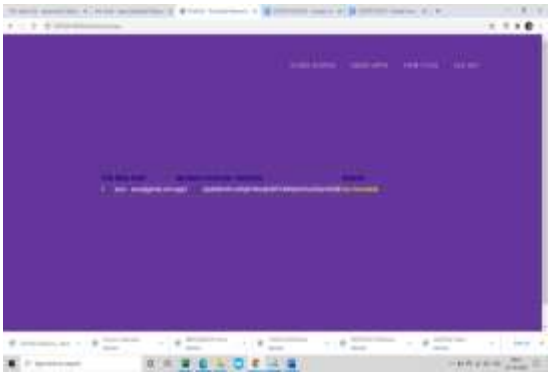
ISSN2454-9940

www.ijasem.org

Vol 18, Issue 3, 2024

User Profile Page:





5. CONCLUSION

We introduced four security rules that capture desirable properties of REST APIs and services. We then showed how a stateful REST API fuzzer can be extended with active property checkers that automatically test and detect violations of these rules. So far, we have fuzzed nearly a dozen production Azure and Office-365 cloud services using the fuzzer and checkers described in this paper. In almost all cases, our fuzzing was able to find about a handful of new bugs in each of these services. About two thirds of those bugs are "500 Internal Server Errors", and about one third are rule violations reported by our new security checkers. We reported all these bugs to the service owners, and all have been fixed. Indeed, violations of the four security rules introduced in this paper are clearly potential security vulnerabilities. The bugs we found have all been taken seriously by the respective service owners: our current bug "fixed/found" ratio is nearly 100%. Moreover, it is safer to fix these bugs rather than risk a live incident – provoked intentionally by an attacker or triggered by accident – with unknown consequences. Finally, it helps that these bugs are easily reproducible and that our fuzzing approach reports no false alarms. How general are these results? To find out, we need to fuzz more services through their REST APIs and check more properties to detect different kinds of bugs and security vulnerabilities. Given the recent explosion of REST APIs for cloud and web services, there is surprisingly little guidance about REST API usage from a security point of view. Our paper makes a step in that direction by contributing four rules whose violations are security-relevant and which are nontrivial to check and satisfy

ISSN2454-9940
www.ijasem.org
Vol 18, Issue 3, 2024

6. REFERENCES

[1] S. Allamaraju. RESTful Web Services Cookbook. O'Reilly, 2010.

[2]Amazon.AWS. https://aws.amazon.com/.

[3]APIFuzzer. https://github.com/KissPeter/APIFuzzer.

[4]AppSpider. https://www.rapid7.com/products/appspider.

[5] V. Atlidakis, P. Godefroid, and M. Polishchuk. RESTler: Stateful REST API Fuzzing. In 41st ACM/IEEE International Conference on Software Engineering (ICSE'2019), May 2019.

[6]BooFuzz. https://github.com/jtpereyda/boofuzz.

[7] Burp Suite. https://portswigger.net/burp.

[8] D. Drusinsky. The Temporal Rover and the ATG Rover. In Proceedings of the 2000 SPIN Workshop, volume 1885 of Lecture Notes in Computer Science, pages 323–330. Springer-Verlag, 2000.

[9] R. T. Fielding. Architectural Styles and the Design of Network-based Software Architectures. PhD Thesis, UC Irvine, 2000.

[10] P. Godefroid, M. Levin, and D. Molnar. Active Property Checking. In Proceedings of EMSOFT'2008 (8th Annual ACM & IEEE Conference on Embedded Software), pages 207–216, Atlanta, October 2008. ACM Press.