RAPID PENETRATION TEST FOR SECURING CHANNEL APIS IN HYBRID CLOUD (DYNAMIC DEFENSE OF CHANNEL API)

Luiza Nacshon[1] and Anna Sandler[2]

[1]Senior Security Engineer, Red Hat, Israel
[2]Software Engineer, Red Hat, Washington D.C

ABSTRACT

The goal of this research is to explore the security aspects of the hybrid Cloud Channel API world in greater depth and develop a rapid penetration testing tool that will help security researchers test Cloud Channel API security more effectively. The research proposes an innovative proxy-based solution for a rapid reactive test implementing a dynamic defense for channel API in the hybrid cloud.

The proxy-based solution executes security testing rules against the channel API requests and validates weaknesses or vulnerabilities as a dynamic defense. Malicious or vulnerable requests may be denied/dropped/alerted, and the results and decisions will be reflected in the API-management dashboard. In the scope of the paper, we focus on known API attacks and in future work, we are going to have a machine learning algorithm for unknown and new channel API attacks.

KEYWORDS

openshift, channel API, security, hybrid cloud, penetration test

1. INTRODUCTION

With the world moving from big software entities to micro services [1], the need of using APIs (application programming interfaces) for those micro services to communicate, has increased. With that, API security attacks have become frequent [2].

In this paper, we will review the most common cloud channel API security attacks [3], on which we focused in our research. Most of these attacks are classic attacks that were adapted to exploit clouds—in particular, hybrid clouds.

Recently, many companies have begun to move most of their data from local data centers to public-managed clouds [4]. But with these movements, some questions arise; Can we store the data on more than one public cloud provider? What if a company wants sensitive data to be stored locally? How do we orchestrate the data between a couple of separate clouds? a hybrid cloud can solve all of these concerns.

The hybrid cloud has become widely used by companies to store their data [5].

It helps to lower maintenance costs for the data that should stay under private control and gives the ability to store the rest of the data in public data centers.

David C. Wyld et al. (Eds): NIAI, MoWiN, AIAP, SIGML, CNSA, ICCIoT - 2023
pp. 95-104, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.130308

When deploying a hybrid cloud, we need to make sure that the API not only can process data that moves between the different clouds but also protects the existing data and does not let sensitive data leak out.

The goal of this research is to study the base rules of securing a channel API in a hybrid cloud and the importance of each and every component of the API. Then, based on each channel API, we can define security test rules that will be executed on each request.

The first step in our research was to understand and explore the possible attacks on hybrid clouds [6], what is currently covered, and what may not be covered yet by the community.

While exploring various security fields, we found that channel API security in a hybrid cloud lacks research and solutions. Therefore we decided to focus on this area and work on innovative proxy-based research and solutions enabling cloud IT professionals to define security rules that will be executed in real-time on every API request.

The defined rules will identify potential security issues in real-time by performing a rapid penetration test (PT), and the reversed proxy may deny or drop a vulnerable request as a dynamic defense.

The proposed solution is an improvement of the current state-of-the-art solutions such as WAF and IDPs [18,19] in such a way that the test rules run against the channel API requests before the traffic enters the cloud and suspicious requests can be dropped. Another point, the rapid PT reversed proxy can be used as a research tool and is flexible enough to contain different types of detection rules, to test channel APIs in development stage and in production stage.

The propped rapid PT proxy resides between the private and public clouds, and continuously tests all the channel API requests between the clouds.

Our research questions were:

- What rules can be defined to let developers know whether the channel API in the hybrid cloud communication is secured or vulnerable to attacks?
- Where should the reversed proxy that executes those rules be located?
- What common attacks should we focus on in our research?

The rest of the paper proceeds as follows: In Section 2, we will introduce the hybrid cloud, channel API, reversed proxy, and prior works on securing the hybrid cloud. In Section 3, we will introduce the attacks on Cloud Channel API that we have focused on in our research and introduce the security rules executed on the proxy by our rapid PT tool. In Section 4, we will provide an in-depth discussion of our research and solution. In Section 5, we will present the demo and the architecture of the rapid PT tool and reversed proxy.

2. HYBRID CLOUD AND CHANNEL API SECURITY

2.1. Security on Hybrid Cloud

A hybrid cloud incorporates some degree of workload portability, orchestration, and management across two or more environments: public, private, or on-premise clouds. The hybrid cloud gives companies the speed and scalability of the public cloud and the control, reliability, and privacy of the private cloud.

Orchestrating data between multiple storage services is subject to security breaches. A security breach in the public cloud can lead to unauthorized access to the private cloud's clusters. Vice versa, a private cloud cluster that is not well secured can lead to unauthorized access to the public cloud.

Incorrect user access control management to one cloud can cause unauthorized access to another cloud. While incorrect data organization may cause a data leak, data can also be lost during the orchestration between the clouds.

To secure a hybrid cloud, we need to understand how these clouds communicate with each other and study the channel API. We can see from the statistics collected by the IDC and introduced by Balasubramanian et al. [6] that security challenges in hybrid clouds seem to be the most common challenges. We also can learn from their work that the number of organizations using cloud-based services, especially hybrid clouds, is growing.

Therefore, motivation is high for attackers to attack the cloud services and gain control of the data in cases where some data resides in a private cloud environment and some in a public cloud environment. Thus we have the motivation to secure the channel-level API between private and public clouds.

Most of the research on hybrid cloud security focuses on authentication, encryption, and storage security [8] [9]. Li et al. [9] presented an interesting prototype for proof of ownership to support authorized deduplication. Our research is focused on securing the channel API between the private cloud and public cloud by finding security weaknesses, vulnerabilities, or potential attacks in real-time.

2.2. Security on Cloud Channel API

To process IO data, the cloud uses an API—a set of definitions and protocols for building and integrating application software that lets a company's product or service (in this case, a cloud) communicate with other products and services without having to know how they're implemented.

APIs are the present and the future; they are the way companies will continue to communicate. The growth in usage of APIs has continued to increase, and the rise of API attacks is happening right now. Many different security companies now offer API security, and some solutions are focusing on cloud APIs [9]. In this paper, we are focusing on rapid pentest security for Cloud Channel API.

The Cloud Channel API enables cloud users to have a single unified resale platform and APIs across all cloud providers' services, e.g., Google Cloud, Workspace, Maps, and Chrome.

Cloud Channel API requires an on-prem client to authenticates with a cloud service before accessing any data. With Cloud Channel API, companies are able to transfer data from many cloud services of different cloud providers. We didn't find any work that focused on testing the security issues of the channel APIs.

Most proposed solutions focus on encryption of the transferred data and access controls. Our goal in this paper is to propose a solution that will perform dynamic defense tests on the most common attacks on cloud APIs. The reversed proxy will execute the security test rules on the channel API request. If the channel API request is found to be malicious or weak, or a vulnerability is found, the requests may be Dropped or Denied and a proper error message will be sent to the admin. The admin can define the required actions.

2.3. Reversed Proxy and Rules

A reversed proxy is an intermediate server between a client and a cluster or a service. A reversed proxy can route requests, depending on the URL, to various cloud services, or it can be there simply to "protect" against some attacks or analyze traffic. A reversed proxy must receive a request, process it, perform some action on it, and forward it to a backend.

In this paper, we are using the reversed proxy for the dynamic defense of the Cloud Channel API, since the reversed proxy server hides the location of the protected on-prem clusters or cloud clusters and monitors the traffic rate and payload to detect potential attacks against the protected clusters.

We use the reversed proxy by integrating the rapid PT tool with predetermined security test rules on the requests coming from any source. The detection process is based on predefined rules, which are executed by the rapid PT tool on the coming request through the channel API.

A reversed proxy was used in several prior works for dynamic defense from attacks. For example, Wurzinger et al. [10] presented a reversed proxy for cross-site web attack mitigation. Zhao et al. [11] presented an innovative solution: a Cookie-Proxy, including a secure cookie protocol to protect a website against SSLStrip attacks. All these prior works use the reversed proxy solution for dynamic defense against web attacks. There is no other solution for dynamic defense against Cloud Channel API attacks.

3. SECURITY RISKS ON CLOUD CHANNEL API

Classic cyber attacks are re-innovated with every new technology on the market. Those attacks are more powerful than ever due to the increasing amount of data and applications moving to cloud technology.

In this section, we present the potential attacks and risks on Cloud Channel API, on which we focused our study. We'll review some of those attacks and how they suit themselves to the modern hybrid cloud architecture and the channel API.

If a channel API in a hybrid cloud is attacked, the whole chain of communication between the private cluster and the public cluster will be disturbed and may cause data loss and denial of activities to all connected services.

3.1. DoS (Denial of Service)

Denial of service, also known as DoS, is a type of attack in which a large amount of packets is sent to a server that can not handle this traffic and causes it to fail—or, as the name states, go out of service. From an API point of view, a Distributed DoS attack can be performed with a large number of API calls at the same time, which can cause the API service to fail.

3.2. Injections

By using injection flaws, such as SQL, NoSQL, command injection, etc, the attacker's malicious data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

3.3. Mass Assignment

Binding client-provided data to data models without proper properties filtering can lead to mass assignment, allowing attackers to modify object properties they are not supposed to.

3.4. Improper Assets Management

APIs tend to expose more endpoints than traditional web applications, making proper and updated documentation highly important.

3.5. Excessive Data Exposure

Developers tend to expose all object properties through an API without considering their sensitivity and relying on clients to perform the data filtering before displaying it to the user.

In our study, we decided to focus on those possible attacks and risks on the channel API. We researched the best rules that our rapid PT tooling should execute on the channel API calls reaching out to the reversed proxy. Those risks may happen due to developers' mistakes or misconfiguration of the channel API.

4. RAPID PENETRATION TESTING ON CLOUD CHANNEL API

In this section, we will elaborate on the research and the theoretical part of the rapid PT, the reversed proxy, and the security rules.

In our study, once a channel API request is processed, the reversed proxy can perform some actions on the request due to its configuration. It is important to emphasize that in many cases, the rules of a reversed proxy are path (location) based: if the path is path A, then apply one action, if path B, apply another action.

Depending on the implementation or on the configuration, a reversed proxy applies rules based on a processed (parsed, URL-decoded, normalized) path or on an unprocessed path. It’s also important to consider if it is case-sensitive or not. For example, will the next path be treated as equal to the previous one by a reversed proxy?

Our next research step was finding the best rule patterns to identify malicious activities in the channel API calls. The rules are executed by the reversed proxy, with the rapid PT on channel API requests reaching out to the reversed proxy. The requests that are reaching out to the reversed proxy are any connections from private clusters to public clusters and vice versa.

On any incoming channel API request, the reversed proxy executes the security test rules on the API requests. If malicious activity is found in the request, the request may be denied or dropped, and we will see an error message in the API-management dashboard with the possible vulnerability or risk. It is upon the admin to decide the actions that should apply upon detection.

Now we will define the parameters extracted by the reversed proxy on all incoming requests:

a) Identify Path - Header <IP.source, IP.destination>
b) PathA - IP.private.cluster, PathB - IP.public.cluster
c) Port - any
d) Protocol - TCP/UDP/SSH
e) Packet.Payload - data within the packet

There are differences between the public and private paths in case the admin would like to perform different security rules checks on each path.

Let's now define the security rules for a possible risk or attack:

Action;Protocol <SIG.Attack.type, Packet.Payload, SID, Filter>

a) *Action - DROP, DENY, ALERT*
b) *SID - Counts the number of instances that arrived for the same destination in X seconds. The admin can define the SID threshold, which will be relevant only for DOS attacks*
c) *SIG.Attack. type - attack signatures executed by the rapid pentest tool, in the reversed proxy and runs the signatures against the* Packet.Payload
d) Filter - Boolean if a filter is applied by the admin on an IP.source

IF SIG.Attack.type DOS and ! Filter(IP.source) -> DROP

IF SIG.Attack.type Injections and ! Filter(IP.source) -> ALERT&DROP

Example for SIG.attack.type == Injections:

exec(Packet.Payload ({15}%b)(len={17}%b)({4}%B)(type=%B)(msg={len-5}%B))

Another example of a SIG.Attack.type rule executed by the rapid pentest tool on the Pack.Payload to test whether the channel API endpoint is vulnerable to SQL injection:

exec(Packet.Payload(…login.asp?User=X'OR'1'/*&Pass=Y*/='1)

As we can see, the network admin should have full flexibility to configure the rapid PT by adding filters, defining SID, and updating signatures. The network admin should get alerts on vulnerabilities or attack detections to the managed API dashboard, which manages the channel API, and get alerts from the reversed proxy.

If the network admin decides not to DROP packets, it is up to them to get proper actions on the alerts.

Once the research phase was completed, we developed a proof of concept (POC), a reversed proxy based on NGINX that helps security engineers test the channel API in the cloud products in an easy way. The POC is presented in Section 5.

We developed a rapid PT tool that works as a proxy between a hybrid cloud and the channel API [Figure 1]. We wrote simple tests for the POC focused on checking the different attacks: DoS and three types of injections (Section 4).

5. TECHNICAL OVERVIEW

In this section, we will provide in-depth technical details on the POC prototype of the rapid PT and the reversed proxy for securing the channel API in a hybrid cloud.

We have deployed two separate Red Hat OpenShift (OCP) clusters [12]: one cluster deployed on a Red Hat cloud lab and one on a private host.

On each cluster, we deployed:

- NodeJS Pod, which serves as an API application.
- MySQL Pod, which stores raw data for the application.
- A container image running a Golang service that is publicly exposed and serves as the channel API endpoint.

The core of the system is the PT service that runs the tests by executing the security rules. The system executes approximately 14 tests that create and send channel API queries with different attack vectors: SQL injections, JSON injections, and a large number of queries at one time to simulate a DoS attack. We used Suricata [14] and Snort [15] tools to generate the security rules.

Here is an example of a DOS detection rule:

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"LOCAL DOS SYN packet flood inbound, Potential DOS"; flow:to_server; flags: S,12; threshold: type both, track by_dst, count 5000, seconds 5; classtype:misc-activity; sid:5;)

To build TCP flood attack packets, we used a packet builder online tool called FrameIP [16], and we included the IP addresses of our private cluster in the source IP address field in the IP header. We generated the following command on the online FrameIP tool to generate TCP flood traffic to the destination private cluster <TCP port 80, IP address 192.168.1.20>:

>frameip –interface 1 –send_mode 1 –loops 0 –wait 0 –ip_type 17 –ip_source r –ip_destination 192.168.1.20 – udp_port_destination 80

In our POC [Figure 1] there is a reversed proxy based on NGINX with an open socket that continuously listens for responses coming from the channel API. On each incoming response, the rapid pentest running on the reversed proxy will execute the test rules on the packet header and the payload.

The PT tool determines the success or failure of a test by the mapping shown in [Table 1]. If the API returned data even though the query was injected, the test would fail, and the dashboard would show red. If the API blocked the query, it means that a suitable rule was set up to block this kind of attack, the test passed, and the dashboard would show green.

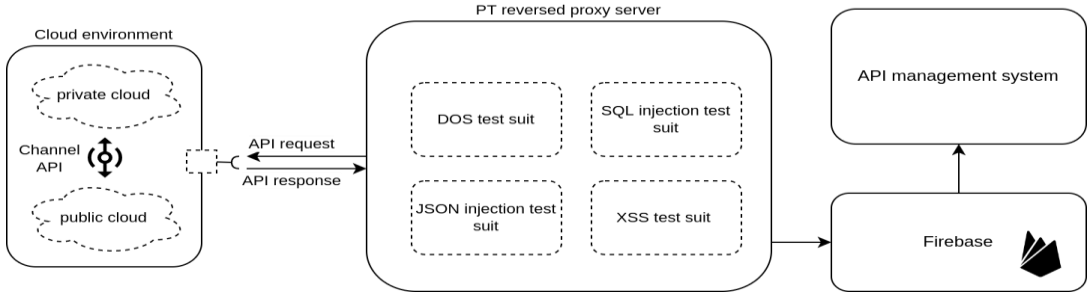

Figure 1. High-level architecture

All of the results are collected into a Firebase DB and displayed to the user in an interactive UI. In addition, each user in the system can add/drop or edit APIs, manage test executions, and look at previous runs.

Table 1. Tests Classification

Query type	Results	
	failed test	passed test
Injected API query	True negative	False positive
Legit API query	False negative	True positive

5.1. Technologies used in the POC

Development Tools

- Dashboard: Flask with bootstrap, Jinja, and JSON
- Users Database: Firebase
- API: v1 - Flask app to SQL DB pod, v2 -Go
- Test Application - NodeJS and MongoDB
- Testing environment: Python 3.7
- Reversed proxy: NGINX server

Development Environment

- Private cloud: Minishift on localhost
- Public cloud: OpenShift Container Platform (OCP) cluster in Dev lab

6. CONCLUSIONS

Working on this project was not easy. We faced a lot of challenges during the research and development phases. The first problem we encountered was the near-total lack of research on this subject. We found cloud security research, hybrid cloud research, cloud API security, or just web API security, but no research or a software solution focused on hybrid Cloud Channel API security, especially cloud-to-cloud channel API.

When we started working on the POC, we had to deploy an environment to test our hypotheses. We tried several workframes:

- IBM cloud: We didn't have enough resources to run an OpenShift cluster, and this option has dropped.
- MOC: The Massachusetts Open Cloud gave us some machines deployed on OpenStack. Deploying OpenShift on OpenStack in this environment was a bit tricky, and we decided to pass.
- Red Hat lab: We requested access to Red Hat's training lab shared clusters, which were blocked by SSH connections
- OCP on the cloud: Deploying the OCP cluster in the QE lab was the easiest way to run a public cloud and was used for the final POC
- Minishift: This was the best match for running a private cloud

After picking up the deployments that fitted us the most, we encountered problems such as:

- How can we create a connection between those deployments?
- How do we choose the right operator to test data I/O?
- Can we run the proxy on the system?

Resolving deployment issues took us over 3 months and was the main concern in the POC. After solving those issues and bringing the demo product to run, we tested several sets of data connected to different API endpoints.

7. SUMMARY AND FUTURE WORK

Working on this project was a unique experience, as we were looking into a new, almost non-researched subject. We had to match facts and concepts from different resources in the academy and industry, talking to professional people from Red Hat Research, R&D, security, quality engineering, and even IBM to create a new product to be developed in the future.

Given the importance of security in the cloud computing world, we found a surprising lack of solutions for securing channel API in the hybrid cloud, especially in live communication.

In the next phase of the research POC, we will likely spend more time finding the best signature to detect various injection attacks.

In the future, we are hoping that the tool developed from this research will be part of the Red Hat Open Sources StackRox [17], and we hope other companies and researchers will find it useful as well. In addition, in the future we are planning to implement a machine learning algorithm on the Rapid PT proxy to have a solution to test and detect also unknown Channel API attacks.
When we started the research we presented the research poster and proposal in the Red Hat research blog [20]. Any comments and suggestions are very welcome.

ACKNOWLEDGMENT

We would like to thank the CTO office in Red Hat. Special thanks to Idan Levi and Stav Schneider for supporting us and our research and development environments. We also would like to thank Ariel University, the department of computer science, and the Cyber Innovation Center, especially Dr. Amit Dvir, for their academic support of our research. Thanks to Fabio Leite for the management support and reviews, and Shaun Stromer for her help reviewing and commenting on the paper. We would also like to thank Avihu Goren for taking over the PoC development and helping us achieve the results we present in the paper.

REFERENCES

[1] SELMADJI, Anfel, et al. Re-architecting oo software into microservices. In: European Conference on Service-Oriented and Cloud Computing. Springer, Cham, 2018. p. 65-73.
[2] BENZAID, Chafika; TALEB, Tarik. ZSM security: Threat surface and best practices. IEEE Network, 2020, 34.3: 124-133.
[3] https://owasp.org/www-project-api-security/
[4] HELLERSTEIN, Joseph M., et al. Serverless computing: One step forward, two steps back. arXiv preprint arXiv:1812.03651, 2018.
[5] HUANG, Xueli; DU, Xiaojiang. Achieving big data privacy via hybrid cloud. In: 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, 2014. p. 512-517.
[6] Balasubramanian, R., and M. Aramudhan. "Security issues: public vs. private vs. hybrid cloud computing." International Journal of Computer Applications 55.13 (2012).
[7] DUBEY, Anukrati; SHRIVASTAVA, Gunjita; SAHU, Sandeep. Security in hybrid cloud. Global Journal of Computer Science and Technology, 2013.
[8] LI, Jin, et al. A hybrid cloud approach for secure authorized deduplication. IEEE transactions on parallel and distributed systems, 2014, 26.5: 1206-1216.

[9] SIRISHA, Avvari; KUMARI, G. Geetha. API access control in cloud using the role-based access control model. In: Trendz in Information Sciences & Computing (TISC2010). IEEE, 2010. p. 135-137.
[10] WURZINGER, Peter, et al. SWAP: Mitigating XSS attacks using a reversed proxy. In: 2009 ICSE Workshop on Software Engineering for Secure Systems. IEEE, 2009. p. 33-39.
[11] ZHAO, Sendong, et al. Cookie-Proxy: a scheme to prevent SSLStrip attack. In: International Conference on Information and Communications Security. Springer, Berlin, Heidelberg, 2012. p. 365-372.
[12] https://www.redhat.com/en/technologies/cloud-computing/openshift
[13] https://kubernetes.io/docs/concepts/workloads/pods/
[14] https://github.com/arvindpj007/Suricata-Detect-DoS-Attack
[15] https://www.snort.org/
[16] https://www.frameip.com/
[17] https://www.redhat.com/en/blog/red-hat-releases-open-source-stackrox-community
[18] Clincy, V., & Shahriar, H. (2018, July). Web application firewall: Network security models and configuration. In 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC) (Vol. 1, pp. 835-836). IEEE.
[19] Woo, S. H. (2012). A Study on Security Capability of IDPS. Journal of the Institute of Electronics Engineers of Korea CI, 49(4), 9-15.
[20] /blog/2021/11/26/rapid-penetration-testing-tool-for-securing-apis-in-cloud-communications/

AUTHORS

Luiza Nacshon - Luiza has a degree in software engineering and a master's degree in science with a thesis in cybersecurity engineering from Ben Gurion University, Israel. Luiza has over 15 years of experience in the cybersecurity and research fields.

Anna Sandler - Anna is a software engineer at Red Hat software Production team; she graduated from Ariel University, Israel, with a bachelor's degree in computer science and mathematics with a specialization in Cybersecurity. Currently, she is pursuing her master's degree in Cybersecurity Risk Management at Georgetown University in Washington, D.C.