Defending Multi-Cloud Applications Against Man-in-the-Middle Attacks

Morgan Reece
mlr687@msstate.edu
Mississippi State University
Mississippi State, MS, USA

Theodore Lander
tel127@msstate.edu
Mississippi State University
Mississippi State, MS, USA

Sudip Mittal
mittal@cse.msstate.edu
Mississippi State University
Mississippi State, MS, USA

Nidhi Rastogi
nxrvse@rit.edu
Rochester Institute of Technology
Rochester, NY, USA

Josiah Dykstra
josiahdykstra@acm.org
National Security Agency
Fort Meade, MD, USA

Andy Sampson
agsamps@uwe.nsa.gov
National Security Agency
Fort Meade, MD, USA

ABSTRACT

Multi-cloud applications have become ubiquitous in today's organizations. Multi-cloud applications are being deployed across cloud service provider platforms to deliver services to all aspects of business. With the expansive use of multi-cloud environments, security is at the forefront of concerns when deploying and managing access to multi-cloud applications and the expanded attack surface of these applications. Attackers can exploit vulnerabilities in multi-cloud environments that expose privileged information to inevitable attack.

In this paper we develop a multi-cloud victim web application deployed as component services. These services are deployed on different cloud service providers. Being deployed on the different cloud service providers expands the attack surface of the multi-cloud victim web application. Using the victim multi-cloud application, we demonstrate a man-in-the-middle attack showing the stealing of privileged credentials. Utilizing ParrotOS as the exploitation server, we demonstrate an attack on an application deployed across three cloud service providers: AWS, Azure, and Rackspace. Having successfully attacked the application, we then implement mitigations and verify the protection by attacking the protected application.

CCS CONCEPTS

• **Security and privacy** → **Access control**; *Multi-factor authentication*; *Web protocol security*; *Security protocols.*

KEYWORDS

multi-cloud; man-in-the-middle; ARP poisoning; identity security

ACM Reference Format:
Morgan Reece, Theodore Lander, Sudip Mittal, Nidhi Rastogi, Josiah Dykstra, and Andy Sampson. 2024. Defending Multi-Cloud Applications Against Man-in-the-Middle Attacks. In *Proceedings of the 29th ACM Symposium on Access Control Models and Technologies (SACMAT 2024), May 15–17, 2024, San Antonio, TX, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649158.3657051

SACMAT 2024, May 15–17, 2024, San Antonio, TX, USA

ACM ISBN 979-8-4007-0491-8/24/05
https://doi.org/10.1145/3649158.3657051

1 INTRODUCTION

The proliferation of cloud-hosted applications continues to increase. While traditional cloud-hosted applications were designed to run as a *single-cloud service*, recently more applications are being deployed to different Cloud Service Providers (CSPs). Called *multi-cloud applications*, these software programs leverage resources and services from multiple cloud providers to fulfill their functionality. For example, organizations leverage *multi-cloud environment*, running applications like human resources on Amazon Web Services (AWS) and IT Service Management on Microsoft Azure [21]. Although this provides interconnectedness, interoperability, and data sharing, the complexity of the multi-cloud environment increases the vulnerability to various threats such as credential theft, privilege escalation, and man-in-the-middle attacks.

In this paper, we present a generic multi-cloud architecture for simulating and testing attacks and their mitigation techniques (see Figure 1). This architecture is implemented in our victim web application. The victim web application is comprised of a web service, an application service, an email service, and a database service, hosted across multiple cloud providers.

Using the victim web application based on the multi-cloud architecture outlined, we execute a Man-In-The-Middle (MITM) attack using ARP Poisoning, which allows the attacker to capture privileged credentials and compromise the database. The purpose of demonstrating the MITM attack is to show the stealing of privileged credentials and how security controls can mitigate vulnerabilities as well as maintain security in a multi-cloud application. Additionally, we show that security controls should encompass strong access control, network security, and encryption of traffic at multiple levels such as at the CSP, the application, and the APIs. It is worth noting that while encryption and access control pose challenges in multi-cloud applications due to differences in implementations by CSPs, using TLS and IAM best practices can significantly improve the security of multi-cloud applications. The main contributions of this paper include:

(1) We develop a multi-cloud-based victim web application deployed as component services on disparate cloud service providers. allowing for the expansion and exposure of the attack surface unique to the multi-cloud environment.
(2) We demonstrate a successful MITM attack that exploits ARP Poisoning in a multi-cloud environment, and bring attention

to the vulnerabilities and relevance of strong access control and network security.

(3) We propose and experimentally analyze a practical mitigation approach for the MITM attack in a multi-cloud environment. We focus on strong access control, network security, and encryption for a holistic security approach.

The rest of the paper is structured as follows. In Section 2, we dive into the background and existing research in multi-cloud attack security and vulnerability mitigation strategies. Next, in Section 3, we discuss the multi-cloud architecture and how it is used to build and deploy a multi-cloud-based victim web application for attack demonstration. In Section 4, we provide the details to execute a MITM attack. In Section 5, we present the mitigation strategies and finally, in Section 6, we summarize our research and suggest future work.

2 BACKGROUND & RELATED WORK

In this section, we first contrast multi-cloud applications with single-cloud applications. We then present the formal architecture of a generic multi-cloud application, APP_{MC}. This architecture serves as a basis for our victim web application (V_{webapp}, Section 3). In addition, we include a comprehensive examination of security issues in multi-cloud applications.

2.1 Single vs. Multi-Cloud Applications

A multi-cloud application is a software program that executes its functions by utilizing the resources and services of multiple cloud providers. Multi-cloud applications offer a range of advantages compared to single-cloud applications. These include cost optimization, improved performance and scalability, enhanced availability and resiliency, and the ability to avoid vendor lock-in [7]. By selecting the most economically viable services for various components of the application, multi-cloud can yield substantial cost reductions in comparison to a single-vendor approach. Different cloud providers demonstrate proficiency in distinct domains. Multi-cloud deployments enable the utilization of the unique advantages of each platform, resulting in enhanced performance. Deploying the application across multiple cloud providers mitigates dependence on a single platform, thereby improving resilience to failures and reducing the likelihood of downtime. With multi-cloud, the application is not constrained to the ecosystem of a single vendor. This affords greater autonomy in selecting services that align with system requirements and in negotiating favorable pricing [8].

Conversely, managing and optimizing a multi-cloud application can be complex, requiring skilled personnel and specialized tools [16]. Ensuring data security and access control across multiple cloud platforms with differing security standards necessitates careful implementation and execution [12]. Multiple contracts, billing systems, and support channels are often part of multi-cloud, which makes operational management difficult and time-consuming.

2.2 Multi-Cloud Application Architecture

A generic multi-cloud application APP_{MC}, shown in Figure 1, has multiple component services $\{s_1, s_2, s_3, ..., s_k\} \epsilon S$, hosted on different cloud platforms $\{c_1, c_2, c_3, ..., c_n\} \epsilon C$. Typically, a many-to-many map exists between the sets of cloud providers C and component services S.

The component services S, hosted by the cloud providers C, communicate with each other using Application Programming Interfaces (APIs) over the Internet. These multiple communication paths between component services can be represented as $\{p_{user}, p_{1,1}, p_{1,2}, ..., p_{i,j}, ..., p_{k,k}\} \epsilon P$, where $p_{i,j}$ is the communication path between component services s_i and s_j. p_{user} is the connection utilized by the user to communicate with APP_{MC}. Generally, multiple communication paths are programmed into APP_{MC} by its developers based on the multi-cloud application software requirements.

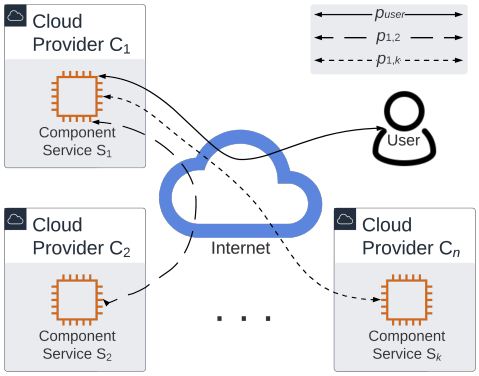

Figure 1: Multi-Cloud Architecture The implementation of an application across a multi-cloud environment can have its k number of component services, $\{s_1, s_2, s_3, ..., s_k\} \epsilon S$, distributed across n number of cloud providers, $\{c_1, c_2, c_3, ..., c_n\} \epsilon C$. Communication paths between component services are represented as $\{p_{1,1}, p_{1,2}, ..., p_{i,j}, ..., p_{k,k}\} \epsilon P$, where $p_{i,j}$ is the communication path between component services s_i and s_j. The user communicates with component service 1 over path p_{user}.

In practice, most of the deployed multi-cloud applications have three distinct types of component services. A *web service* manages user interaction through a website or a mobile application. Multiple *application services* process computational logic. *Data repositories and database services* store APP_{MC} data for manipulation and retrieval. The following subsection addresses the vulnerabilities that arise in this architecture as a result of its design decisions.

2.3 Security Concerns in Multi-Cloud Applications

In Section 2.1, we presented differences between single and multi-cloud applications. The architectural and implementation differences identified do not mitigate vulnerabilities; rather, they contribute to the vulnerabilities. Multi-cloud applications inherit all the security issues found in single-cloud environments [5]. In our efforts to find supporting literature for our research, we discovered that there is little research on multi-cloud risk and vulnerability analysis, let alone multi-cloud exploitation.

Reece et al. present a multi-cloud risk and vulnerability analysis model that integrates well-known industry standard frameworks

to produce a holistic risk model [21]. The current fragmented approach leads to gaps in vulnerability identification and mitigation, leaving security holes within the multi-cloud environment. The paper addresses the need for this integrated approach by using STRIDE, DREAD, and the MITRE ATT&CK frameworks to identify, qualify, and mitigate risks in the multi-cloud environment. The six attack vector categories that are identified and analyzed are cloud architecture, APIs, authentication, automation, management differences, and cybersecurity legislation.

Afolaranmi et al. present a method for enhancing the security of a multi-cloud environment by utilizing a security evaluation framework. The presence of buffer overflow and cross-site scripting attacks is emphasized, as they can have a significant impact on multi-cloud environments. To address this issue, the developed framework incorporates essential components, including operational and architectural perspectives [3].

Lingle et al. in their paper provide an overview of Security as a Service (SECaaS) [13]. The preliminary examination reveals the existing fragmented service offerings, such as logging as a service, IAMaaS, SOCaaS, DLP, and IPS/IDS services, which are included by each cloud service provider in their cloud offerings.

The subsequent stage of the analysis involves introducing a novel third-party SECaaS (Security as a Service) that utilizes innovative cloud security techniques.

IAM and its associated policies dictate the authorization process for individuals seeking access to cloud resources. Additionally, they represent a primary focus for potential attackers. This risk can be exacerbated by multi-cloud strategies, which increase complexity and make it more difficult to enforce a uniform IAM policy across multiple cloud environments [2]. Attackers can exploit weak IAM policies or poorly managed identities to gain unauthorized access to resources and data. For instance, a common attack is identity spoofing, where an attacker impersonates a legitimate user to bypass access control measures [15]. In multi-cloud environments, the risks are intensified due to the presence of individual Identity and Access Management (IAM) systems for each cloud platform, which can complicate the coordination of their administration.

Vulnerabilities in a multi-cloud environment stem from differences in how CSPs implement common technologies, management control schemes, and communication between component services deployed in difference CSPs [21]. MITM attacks target exploiting the expanded attack surface that is created when component services are deployed in different CSPs. The MITM attack can be characterized as a 'fishing' expedition where the attacker hopes to find privileged information in the traffic that they have captured while running the attack. In this paper, we exploit weaknesses in the APP_{MC} architecture. We next present our victim web application, upon which we execute a MITM attack.

3 MULTI-CLOUD BASED VICTIM WEB APPLICATION

To demonstrate the MITM attack in Section 4, we first describe a multi-cloud victim web application (V_{webapp}). V_{webapp} implements the multi-cloud architecture (Section 2.2). The component services of V_{webapp} serve distinct functions demonstrative of a multi-cloud application: *web service and application service*, *email service*, and *database service*. V_{webapp} setup leverages three cloud providers: Microsoft Azure, Amazon Web Services (AWS), and Rackspace (Figure 2).

V_{webapp} web service is a python-flask application that provides a user interface. It runs on a Linux server deployed in the Azure cloud [17]. The web service renders the web pages, takes the input from the user, and sends the data to the V_{webapp} application service for processing. The V_{webapp} application service written in Python receives user data from the web service. Once data is received, the application service executes the requested operation on the data. V_{webapp} application service communicates with two other services; database service and email service. V_{webapp} uses the Internet for users to connect and the communication needs of the different services hosted by the different cloud providers. User authentication is provided through username/password credentials communicated to V_{webapp} over HTTPS.

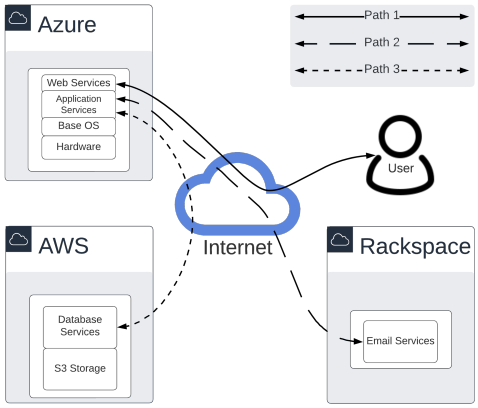

Figure 2: Multi-Cloud based Victim WebApp Utilizing three cloud service providers hosting four services. Communication between services takes place over the Internet via APIs.

The application service leverages APIs to communicate with the database, and the email services hosted on AWS and Rackspace respectively. The application service utilizes username and password credentials to authenticate via the database service API. The connection to the database service uses the standard MySQL port and API over the Internet. We use MySQL as our database service hosted on AWS [6].

The database service utilizes the AWS S3 storage system for its block storage of tables and data. The V_{webapp} application service communicates with the email service over the Internet by sending email messages using the Simple Mail Transport Protocol (SMTP) [11] and receiving email messages using the Internet Message Access Protocol (IMAP) [1] API. The email service is hosted by Rackspace [20], and connections to the email service are authenticated using credentials unique to the user sending or receiving the email.

V_{webapp} experimentation environment allows us to demonstrate attacks on the Internet-exposed interface between the application service and the database (see Section 4). Our V_{webapp} implementation, typical of multi-cloud deployment environments, provides us

with a representative attack environment. These systemic vulnerabilities are rooted in data-sharing requirements in multi-cloud environments [21]. When information is exchanged among component services that are hosted on distinct cloud providers, trust is required among the various component services of the victim web application. Through access control, the confidentiality, availability, and integrity of the information are validated, thereby establishing trust between the different component services. Trust is a prerequisite for information exchange among component services of V_{webapp} that are hosted on distinct cloud providers [18]. By implementing access control that verify the information's availability, confidentiality, and integrity, trust can be established among component services.

The interconnections and implementation details of the V_{webapp} deployment are also shown in Figure 2. The Internet-exposed data communications paths that are susceptible to attacks have been defined below:

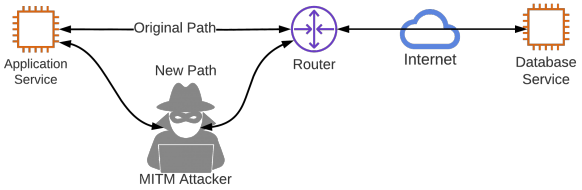

Figure 3: Attack Diagram: The MITM attack removes the original path between the application service and the router and replaces it with a path through the attacker.

- **Path 1 - User to/from Web Service**: The V_{webapp} web services run in Azure and receive the user input over the Internet (Figure 2). This is a common vector for many types of attacks, including DDoS, traffic sniffing, credential stealing, and cross-site scripting.
- **Path 2 - Application Service to/from Database Service**: The application service, which is also deployed on the Azure cloud server communicates with the AWS database service over the Internet through an API identified as *Path 2* in Figure 2. This path is susceptible to attacks like MITM attacks, substitution attacks, and privilege elevation.
- **Path 3 - Application Service to/from Email Service**: The application service utilizes *Path 3* to communicate email operation commands to the email service. Utilizing the information in the database service, the application service configures the authentication with the email service to allow the user to send, read, and manage their emails. *Path 3* utilizes the SMTP (Simple Mail Transfer Protocol) and the IMAP (Internet Message Access Protocol) API to communicate the email operations requested by the user through the user interface. A common attack on *Path 3* is packet sniffing, which includes credential stealing and other traffic interception or injection attacks.

4 MAN IN THE MIDDLE ATTACK DEMONSTRATION

A MITM attack was successfully executed on the V_{webapp} utilizing ARP Poisoning. The attack on the application service was launched from the attack server, which is located on the same subnet as the application service. The attack was executed on the path between the application service, hosted on Azure, and the database, hosted on AWS, along *Path 2* as seen in Figure 2. We used Ettercap [19] application running on a ParrotOS [14] Linux attack server to execute the ARP Poisoning attack and capture the re-routed network traffic with Wireshark. Ettercap sends ARP messages that replace the router's MAC address with the attack server's MAC address in the applications service's ARP table.

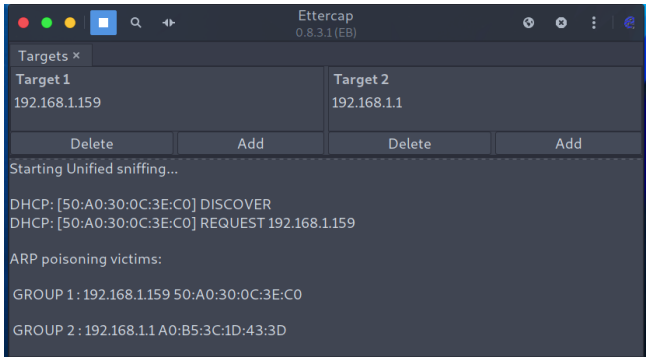

Figure 4: Ettercap ARP Poisoning: Using Ettercap, the attacker can send ARP commands to the application service and replace the MAC address of the router with the MAC address of the attack server.

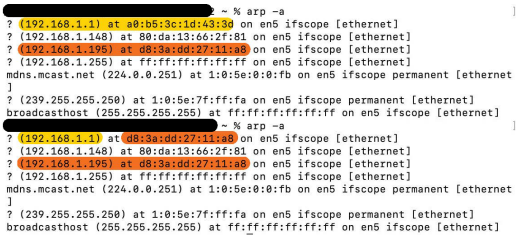

Figure 5: Attacker ARP Poisoning: Utilizes ARP protocol to update the ARP entry for the network router in the ARP table of the application service.

A MITM attack is a cyber-attack where an attacker intercepts and potentially alters the communication between two unsuspecting parties. In a multi-cloud environment, an attacker executes a MITM attack by inserting an attack server between two component services along their communication path (see Figure 3). The attacker utilizes ARP [10] as the initial attack vector. ARP is a network protocol that communicates Media Access Control (MAC) addresses of each device on a subnet to every other device on the subnet to allow proper delivery of Ethernet packets throughout that subnet. The MAC address is added to each Ethernet packet of network communication to ensure proper delivery of the packet. The ARP table on a network device, such as a server, contains the IP Address and associated MAC address for each device on the local subnet. ARP is a foundational protocol to Ethernet and has no inherent security to prevent attacks such as ARP Poisoning. ARP

Poisoning is the corruption of a service's ARP table. Using Ettercap on ParrotOS, the ARP Poisoning attack was executed on the application service by replacing the MAC address of the router with the MAC address of the attack server (see Figure 4). With the MAC address replaced, the attack server now receives all the traffic from the application service that is leaving the subnet, which includes communication to the database service (see Figure 5).

The attacker will do three things with the traffic from the application service: First, capture and store the traffic. Second, search the captured traffic for user credentials and other privileged information. Third, forward the traffic onto the router. Forwarding the traffic keeps the communication between the application service and database service active, and therefore prevents either service from realizing that there is a MITM attack occurring.

The execution of the MITM attack attack on the V_{webapp} follows these steps. The attacker compromises the V_{webapp} network by gaining control of a system on the same subnet. This is not covered in our research and is assumed that the attacker has compromised a system already on the network. The attacker installs the Ettercap application with the ARP Poisoning functionality. The attacker runs Ettercap and sets up the ARP Poisoning attack. The application service is set as Target 1 and the router for the subnet is set as Target 2. The ARP Poisoning attack is then initiated. The Ettercap application sends an ARP update to the application service with its own MAC address to be associated with the IP address of the router. In Figure 5, the "arp -a" command is run before and after the ARP Poisoning attack has begun. The first set shows the IP address of the router, 192.168.1.1, which has a different MAC address than the attack server, 192.168.1.195. The second set of outputs from the "arp -a" command shows that the router and the attack server have the same MAC address. With the ARP table poisoned, the application service traffic meant for the router is instead going to the attack server. Shown in Figure 3, the ARP Poisoning changes the communication path with the router from the “Original Path" to the “New Path”.

With the communication path now going through the MITM Attacker, they can capture and inspect/analyze the packets from the application service. In the analysis of the application's network traffic, we can find the admin credentials for the database service (see Figure 6). The admin credentials to the database service allow our attacker to read all data in the database. After successfully retrieving data from the database, the attacker can then use the extracted information to gain unauthorized access to the email accounts stored in the database. The compromise of the email accounts is beyond the scope of our MITM attack, thus we did not exploit any of the users' email accounts.

A successful MITM attack leads to the interception of privileged information by the attacker. The compromise of sensitive information enables the attacker to exploit it for the purpose of advancing the attack or to trade or disclose the pilfered data.

5 MITIGATION AGAINST MITM ATTACKS

Mitigating MITM attacks in multi-cloud environments centers around strong access control and network security. Attackers very often deploy an ARP attack as the initial step to a MITM attack since, as mentioned in Section 4, ARP has no inherent security, and is therefore susceptible to ARP Poisoning with limited direct mitigation available.

Preventing ARP Poisoning can be done through a couple of techniques. Static ARP tables would prevent the attacker from being able to change the ARP table of the application service. However, this limits the dynamic nature of the subnet, and in large environments would require a large amount of manual effort by network engineers. Dynamic ARP Inspection, found on higher-end switches verifies all ARP updates, and discards updates that look malicious. This advanced feature is becoming more common on small business switches but requires a higher level of network administration effort and expertise.

If the techniques to prevent an ARP attack cannot be implemented, then we can deploy mitigations that limit the impact of a successful attack. Isolating each service to a subnet would help mitigate an ARP attack. The network would have to be configured to have one device per subnet, therefore ARP messages and updates/changes would not be sent to other devices. Also, with the isolation of each device, it would be easy to detect if an attacker was able to add a device to a subnet which would be removed from the network.

The mitigation against the MITM attack deployed in our V_{webapp} is encryption, where we encrypt all the network traffic to and from the application service. Several challenges come with deploying encryption in a multi-cloud environment. APP_{MC} are deployed across multiple CSPs. Each CSP has its own method of implementing and managing encryption. Strict coordination efforts must be put in place to ensure that the communication path between services is secure. The best practice noted in Scott et al. [23], is to always use TLS. Our V_{webapp} mitigation implemented TLSv1.2. TLS must be implemented on both sides of the communication path and should use the highest common version that each service can support. Our database service supported TLSv1.3, the latest TLS version. However, our application service only supported up to TLSv1.2, therefore, TLSv1.2 was implemented. To encrypt the path from the application service to the database service we enabled encryption AWS for our database instance. The counterpart in the application service is to include the SSL/TLS certificate bundle (us-east-2-bundle.pem) from AWS. The two sides of this configuration set up TLS encryption of the communication path between the application service and the database service. With encryption enabled, when our attacker

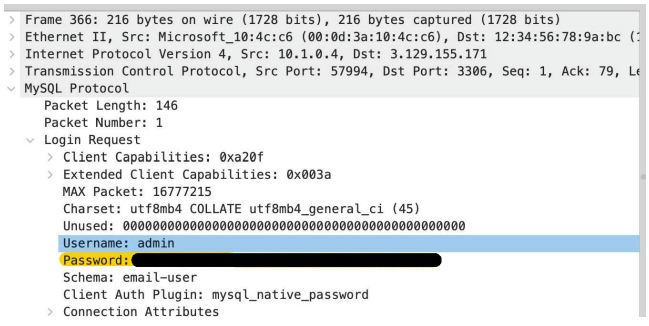

Figure 6: Attacker stolen credentials: Using Wireshark, the attacker is able to capture and inspect the network traffic to steal the user/administrator credentials.

captures the network traffic, they are unable to decipher the data that is being transferred between the component services.

Other mitigations noted in Scott et al. [23] target access control, such as implementing SSO and secure API key management. Utilization of a Single Sign-On (SSO) technology enables strong access control. SSO offloads the authentication of a user to a third-party identity provider that has implemented advanced authentication methods like Multi-Factor Authentication (MFA). Access control and IAM management policy differences between the CSPs create vulnerabilities that can be exploited by attackers, which were explained in Section 2.3. Without a third-party identity provider, we were not able to implement SSO. Implementation of API keys and their secure management is another mitigation that supports strong access control in the APP_{MC}. Because our V_{webapp} is integrating one application service and one database service, the increased security gained through the use of an API key over username/password combination is limited and therefore out of scope for our experimentation.

A significant development in access control is Self Sovereign Identity (SSI) [4, 9]. Authentication through SSI is dependent on the trustworthiness of a third-party verifier and the verification of the user's identity. Integrating SSI in the multi-cloud environment would enhance supervision of authentication across all component services and empower users with greater control over privileged information [22].

6 CONCLUSION & FUTURE WORK

The nature of a multi-cloud application being spread across multiple cloud providers opens it up to attacks. The challenge in multi-cloud architectures lies in securely integrating and accessing the distributed component services of the victim web application (V_{webapp}) discussed in our research. We conducted experiments that involved launching targeted attacks against V_{webapp}. These attacks were specifically focused on exploiting the limited or weak access controls in place during the inter-service communication process. The executed MITM attack showcased the attacker's capability to intercept and acquire the user's information in the absence of sufficient mitigations. The presence of limited mitigations increases the vulnerability of the V_{webapp}, potentially leading to more severe and damaging attacks. We also discussed the benefits of incorporating robust access controls, such as sophisticated user authentication and network security technology, to minimize the vulnerability to unauthorized access by potential attackers. Furthermore, we have described the efficacy of implementing mitigations in reducing the vulnerabilities inherent in such an environment. Additional research is necessary in this domain as a result of the emergence of new attacks carried out by the attackers. There is a need to investigate and understand how these new attacks are being deployed and leveraged by attackers; and for the development of targeted security strategies specific to these newly developed attacks.

ACKNOWLEDGMENTS

This research was supported by NSA H98230-21-1-0317, and National Science Foundation (NSF) grant #1565484.

REFERENCES

[1] B. Leiba A. Melnikov. 2021. Internet Message Access Protocol, RFC 9051. https://www.ietf.org/rfc/rfc9051.html
[2] Sandesh Achar. 2022. Cloud Computing Security for Multi-Cloud Service Providers: Controls and Techniques in our Modern Threat Landscape. *International Journal of Computer and Systems Engineering* 16, 9 (2022), 379–384.
[3] Samuel Olaiya Afolaranmi, Borja Ramis Ferrer, and Jose Luis Martinez Lastra. 2018. A Framework for Evaluating Security in Multi-Cloud Environments. , 3059–3066 pages. https://doi.org/10.1109/IECON.2018.8591454 ISSN: 2577-1647.
[4] Md. Rayhan Ahmed, A. K. M. Muzahidul Islam, Swakkhar Shatabda, and Salekul Islam. 2022. Blockchain-Based Identity Management System and Self-Sovereign Identity Ecosystem: A Comprehensive Survey. *IEEE Access* 10 (2022), 113436–113481. https://doi.org/10.1109/ACCESS.2022.3216643
[5] Mohammed A. AlZain, Eric Pardede, Ben Soh, and James A. Thom. 2012. Cloud Computing Security: From Single to Multi-clouds. In *2012 45th Hawaii International Conference on System Sciences*. IEEE, New York, NY, USA, 5490–5499. https://doi.org/10.1109/HICSS.2012.153
[6] Amazon. 2024. Amazon Relational Database Service. https://aws.amazon.com/rds/.
[7] M. G. Avram. 2014. Advantages and Challenges of Adopting Cloud Computing from an Enterprise Perspective. *Procedia Technology* 12 (2014). https://doi.org/10.1016/j.protcy.2013.12.525
[8] Ataollah Fatahi Baarzi, George Kesidis, Carlee Joe-Wong, and Mohammad Shahrad. 2021. On Merits and Viability of Multi-Cloud Serverless. In *Proceedings of the ACM Symposium on Cloud Computing* (Seattle, WA, USA) *(SoCC '21)*. Association for Computing Machinery, New York, NY, USA, 600–608. https://doi.org/10.1145/3472883.3487002
[9] Md Sadek Ferdous, Farida Chowdhury, and Madini O. Alassafi. 2019. In Search of Self-Sovereign Identity Leveraging Blockchain Technology. *IEEE Access* 7 (2019), 103059–103079. https://doi.org/10.1109/ACCESS.2019.2931173
[10] Internet Engineering Task Force. 1982. *An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware*. Request for Comments RFC 826. Internet Engineering Task Force. https://doi.org/10.17487/RFC0826 Num Pages: 10.
[11] J. Klensin. 2008. Simple Mail Transfer Protocol, RFC 5321. https://www.rfc-editor.org/rfc/rfc5321.html
[12] P. Ravi Kumar, P. Herbert Raj, and P. Jelciana. 2018. Exploring Data Security Issues and Solutions in Cloud Computing. *Procedia Computer Science* 125 (2018), 691–697. https://doi.org/10.1016/j.procs.2017.12.089
[13] Jason Lingle, Kevin Dickens, Iram Bakhtiar, and James Herford. 2019. Security-as-a-Service in a Multi-Cloud Environment. https://doi.org/10.13140/RG.2.2.27812.12166
[14] Parrot Linux. 2024. Parrot Security. https://parrotlinux.org/. Accessed: 2023-10-05.
[15] Mohammad Masdari and Marzie Jalali. 2016. A survey and taxonomy of DoS attacks in cloud computing. *Security and Communication Networks* 9, 16 (2016).
[16] Lawrence E. Meyer and E. Billionniere. 2021. Upskilling to Meet Cloud Talent Needs. *2021 ASEE Virtual Annual Conference* null (2021), null. https://par.nsf.gov/biblio/10288742
[17] Microsoft. 2024. Azure Cloud. https://azure.microsoft.com/. Accessed: 202-01-12.
[18] NR Paul and D Paul Raj. 2021. Enhanced Trust Based Access Control for Multi-Cloud Environment. *Computers, Materials & Continua* 69, 3 (2021), 3079–3093.
[19] Ettercap Project. 2024. Ettercap. https://www.ettercap-project.org/. Accessed: 2023-11-12.
[20] Rackspace. 2024. Rackspace. https://www.rackspace.com/. Accessed: 2023-12-01.
[21] Morgan Reece, Theodore Edward Lander, Matthew Stoffolano, Andy Sampson, Josiah Dykstra, Sudip Mittal, and Nidhi Rastogi. 2023. Systemic Risk and Vulnerability Analysis of Multi-cloud Environments. arXiv:2306.01862 [cs.CR]
[22] Morgan Reece and Sudip Mittal. 2022. Self-Sovereign Identity in a World of Authentication: Architecture and Domain Usecases. arXiv:2209.11647 [cs.CR]
[23] Sam Scott and Graham Neray. 2021. Best practices for REST API security: Authentication and authorization - Stack Overflow. https://stackoverflow.blog/2021/10/06/best-practices-for-authentication-and-authorization-for-rest-apis/