**Threat Intelligence Report**

### **1. Overview**

This report provides an analysis of a recent security breach at a financial company. The attack was caused by an unpatched vulnerability in a web application, allowing attackers to gain unauthorized access to the network. The report outlines the types of attacks that could have been used, explains how the vulnerability was exploited, and provides recommendations for preventing similar incidents in the future. Additionally, it lists the AWS and Azure resources that could be affected by such an attack.

### **2. Types of Attacks**

Several attack techniques could have been used to exploit the unpatched vulnerability in the web application:

#### **2.1 SQL Injection (SQLi)**

* Attackers inject **malicious SQL queries** into an input field to manipulate the database.
* Can be used to **bypass authentication, extract sensitive data, modify records, or delete data**.

#### **2.2 Remote Code Execution (RCE)**

* A vulnerability allows attackers to **execute arbitrary code** on the server remotely.
* This can lead to **complete system compromise**, allowing attackers to install malware or take control of the network.

#### **2.3 Cross-Site Scripting (XSS)**

* Attackers inject **malicious scripts** into web pages viewed by users.
* This can be used to **steal session cookies, impersonate users, or spread malware**.

#### **2.4 Privilege Escalation**

* Attackers exploit system misconfigurations or vulnerabilities to **gain higher-level privileges**.
* This can allow them to **access restricted resources** or move laterally across the network.

### **3. How the Vulnerability was Exploited**

The attackers exploited an **unpatched SQL Injection vulnerability** in the web application. Here’s how the attack unfolded:

1. **Initial Exploitation:** The attacker identified a vulnerable input field that did not properly sanitize user input.

**SQL Injection Attack:** The attacker inserted a specially crafted SQL query:  
 ' OR '1'='1'; --

1. This allowed them to **bypass authentication and gain access to the database**.
2. **Privilege Escalation:** Using credentials obtained from the database, the attacker **gained admin-level access**.
3. **Lateral Movement:** The attacker moved across **AWS/Azure resources**, accessing additional systems and sensitive data.
4. **Data Exfiltration:** The attacker **exfiltrated financial records and personally identifiable information (PII)**.

### **4. Preventive Measures**

To prevent similar attacks in the future, the following security measures should be implemented:

#### **4.1 Apply Software Updates & Patches**

* Regularly update web applications, databases, and underlying servers.
* Enable **automatic security updates** in AWS/Azure.

#### **4.2 Implement Web Application Firewall (WAF)**

* Deploy **AWS WAF** or **Azure Web Application Firewall (WAF)** to filter and block malicious traffic.
* Configure **rules to detect SQL Injection, XSS, and RCE attacks**.

#### **4.3 Enforce Input Validation & Prepared Statements**

* Validate and sanitize all **user input** to prevent SQL Injection.
* Use **prepared statements with parameterized queries** to eliminate injection risks.

#### **4.4 Implement Multi-Factor Authentication (MFA)**

* Require **MFA for all user logins**, especially privileged accounts.
* Use **AWS IAM MFA** or **Azure Active Directory MFA**.

#### **4.5 Restrict Network Access**

* Implement **least privilege access** for all users and services.
* Use **AWS Security Groups** and **Azure Network Security Groups** to enforce network segmentation.

### **5. Affected Cloud Resources**

The following **AWS and Azure services** may be impacted by such attacks:

| **Cloud Provider** | **Impacted Resources** |
| --- | --- |
| **AWS** | EC2 (Virtual Machines), RDS (Databases), S3 (Storage), Lambda (Serverless Computing) |
| **Azure** | Virtual Machines, Azure SQL, Blob Storage, Azure Functions |

### **6. Conclusion**

The security breach was caused by an **unpatched SQL Injection vulnerability** in the web application. Attackers exploited this weakness to gain **unauthorized access** and move laterally across the network. To prevent future incidents, organizations should implement **patch management, input validation, WAF protection, and least privilege access policies**. Additionally, securing cloud resources with **AWS/Azure security tools** is critical to preventing future cyber threats.

By applying these security best practices, financial organizations can **enhance their cyber defense posture and reduce the risk of future breaches**.

### **7. References**

* [MITRE ATT&CK Framework](https://attack.mitre.org/)
* [AWS Security Hub](https://aws.amazon.com/security-hub/)
* [Azure Security Center](https://azure.microsoft.com/en-us/services/defender-for-cloud/)