SentinelOne

2025 Cloud Verified Exploit Paths and Secrets Scanning Threat Report

White Paper

Introduction

In today's evolving digital landscape, understanding the threats and specific ways attackers infiltrate cloud environments is paramount. SentinelOne's Cloud Verified Exploit Paths and Secrets Scanning Threat Report sheds light on the most critical and dangerous cloud threats observed in 2024, backed by insights drawn from our extensive base of over 11,000 customers. This report doesn't just enumerate common vulnerabilities; it reveals the most prevalent exact paths attackers are using to gain entry and access, providing unparalleled visibility into real-world attack scenarios.

At the core of this insight is SentinelOne's Verified Exploit Paths™ technology, which highlights potential attack routes and prioritizes risks within cloud environments. This unique approach, powered by our Offensive Security Engine, simulates attacks against cloud infrastructure to uncover vulnerabilities and provide clarity on which issues are most pressing. This allows security teams to move beyond theoretical possibilities and focus their attention on genuine, verified threats.

We've then coupled these threats with our Secrets Scanning data taken from the same customer base. The exposure of secrets—such as API keys, credentials, and tokens—has become an often underprioritised risk for organizations. In this report SentinelOne sets about illuminating the most critical and prevalent types of secrets discovered within cloud environments.

This report presents an integrated view of in-depth analysis of each of the top verified exploit paths as well as exposed secrets we are observing across our customer base for each severity category—Critical, High, Medium and Low. As well as revealing the top 5 for each severity category. These insights will empower organizations to:

- Prioritize remediation efforts based on real exploit data.
- Enhance defensive strategies by understanding attacker behavior.
- Allocate resources more effectively to mitigate risks that matter most.

By leveraging the findings from this report, security teams can gain a clearer understanding of the practical threats facing their cloud infrastructure and take informed actions to bolster their security posture.

Top 5 CRITICAL Severity Verified Exploit Paths		
Vulnerability	**Severity**	**Prevalence**
CVE-2020-35489	CRITICAL	
CVE-2018-18925	CRITICAL	
CVE-2021-41277	CRITICAL	
CVE-2021-45046	CRITICAL	
CVE-2021-41277-AWS	CRITICAL	

Top 5 CRITICAL Severity Secrets		
Secret	**Severity**	**Prevalence**
AWS Keys	CRITICAL	
RazorPay API Keys	CRITICAL	
Twilio Master Credentials	CRITICAL	
Github Token	CRITICAL	
Github Old Token	CRITICAL	

The Top CRITICAL Severity Verified Exploit Path in 2024

The most frequent CRITICAL Verified Exploit Path is CVE-2020-35489.

CVE-2020-35489 is a vulnerability affecting the `Gurux.DLMS` library, which is used in the implementation of smart meter protocols. This specific flaw allows for **denial of service (DoS)** or potentially more severe impacts due to improper input validation within the protocol's parsing logic.

Exploitation in the Wild

The vulnerability primarily arises from the following conditions:

1. **Malformed Packet Handling**
   An attacker can exploit CVE-2020-35489 by sending specially crafted packets to a system that uses the `Gurux.DLMS` library. These packets can disrupt the system's operation or force it into an unstable state.

2. **Remote Trigger**
   The flaw can be triggered remotely, allowing attackers to send malformed data over the network to exploit the weakness, potentially leading to the exhaustion of system resources or crashes.

While there have been limited reports on widespread exploitation in the wild, any system utilizing this library and exposed to untrusted network traffic could be vulnerable.

Mitigation Techniques

Organizations can protect against CVE-2020-35489 with the following measures:

1. **Update the Affected Library**
   Ensure that the `Gurux.DLMS` library is updated to the latest version where this vulnerability is addressed. Always monitor the library maintainers' website or repository for security patches.

2. **Input Validation**
   Implement additional input validation to detect and block malformed or unexpected data packets before they reach critical parsing functions.

3. **Network Segmentation**
   Restrict access to systems running the `Gurux.DLMS` library by segmenting networks and only allowing trusted traffic.

4. **Monitoring and Alerts**
   Deploy intrusion detection systems (IDS) or intrusion prevention systems (IPS) to monitor for unusual network traffic patterns that could indicate exploitation attempts.

The Most Frequent CRITICAL Severity Secret

The most frequently exposed **CRITICAL** secret is **AWS Keys**, with the highest prevalence across all customer environments. These keys provide programmatic access to AWS services, including compute, storage, and database resources. If exposed, they represent a significant risk to cloud security, as attackers can use them to manipulate an organization's AWS environment.

Key Risks

1. **Unauthorized Infrastructure Access**
   Exposed AWS Keys enable attackers to directly access AWS resources, potentially leading to unauthorized data retrieval, infrastructure changes, or service disruptions.

2. **Privilege Escalation**
   If the exposed keys have administrative privileges, attackers can escalate their access, gaining control over the entire AWS account.

3. **Service Abuse and Financial Impact**
   - Attackers may use the keys to launch expensive resources, such as large-scale compute instances, leading to substantial financial loss.
   - Malicious actors might also use AWS resources for illicit activities, such as cryptomining.

4. **Data Breach**
   Attackers can use the keys to access sensitive data stored in AWS services, including S3 buckets, RDS databases, or other critical systems.

5. **Lateral Movement**
   Exposed keys can provide attackers with an entry point to explore the broader cloud environment, identifying and exploiting additional vulnerabilities.

Mitigation Techniques

1. **Implement Key Management Best Practices**
   - Store AWS Keys securely using tools such as **AWS Secrets Manager**, **AWS Systems Manager Parameter Store**, or third-party secret management solutions
   - Never hardcode keys into source code or configuration files, and use environment variables or secret injection instead.

2. **Use IAM Roles**
   - Replace long-term access keys with **IAM roles**, which provide temporary credentials and eliminate the need for hardcoded keys
   - Use **instance profiles** for EC2 instances or **IAM roles for AWS Lambda** to grant access dynamically.

3. **Enable Multi-Factor Authentication (MFA)**
   Require MFA for all sensitive operations, especially those involving root or administrative keys.

4. **Restrict Access**
   - Apply the **principle of least privilege** to ensure that access keys only have permissions required for their specific tasks
   - Use **IAM policies** to restrict API calls or regions that the keys can access.

5. **Monitor and Detect Exposures**
   - Enable **AWS CloudTrail** to monitor API calls and detect anomalous behavior.
   - Use **Amazon GuardDuty** to identify potential unauthorized use of exposed keys.
   - Regularly scan for exposed keys in code repositories, logs, or cloud storage.

6. **Rotate Keys Regularly**
   - Establish a policy for regular key rotation to minimize the window of exposure.
   - Immediately revoke and replace any keys suspected of being compromised.

7. **Enable AWS Billing Alerts**
   Set up billing alarms to detect unexpected usage that could indicate unauthorized activity.

8. **Encrypt Data and Secure Resources**
   - Use **S3 bucket policies** and **encryption** to protect data even if access keys are exposed.
   - Restrict access to critical services using **VPC endpoints** or **security groups**.

9. **Educate Developers**
   Train developers on the importance of key security and the dangers of sharing or exposing keys in repositories, logs, or configuration files.

Top 5 HIGH Severity Verified Exploit Paths		
Vulnerability	**Severity**	**Prevalence**
CVE-2021-40822	HIGH	
CVE-2019-11248	HIGH	
mysql-native-password	HIGH	
dirsearch-php-my-admin	HIGH	
CVE-2021-44228	HIGH	

Top 5 HIGH Severity Secrets		
Secret	**Severity**	**Prevalence**
MongoDB Credentials	HIGH	
Google Cloud Credentials	HIGH	
Github Token	HIGH	
Google Keys Oauth2	HIGH	
Opsgenie Api Key	HIGH	

The Most Impactful HIGH Severity Verified Exploit Path in 2024

The most impactful HIGH Verified Exploit Path is CVE-2021-40822.

CVE-2021-40822 is a high-severity Server-Side Request Forgery (SSRF) vulnerability affecting GeoServer versions up to 2.18.5 and versions 2.19.x up to 2.19.2. This flaw can be exploited through the configuration option that allows setting a proxy host, potentially enabling attackers to manipulate server requests to access internal resources or sensitive data. The main risk involves unauthorized data exposure as attackers can leverage SSRF to send crafted requests via the vulnerable server.

Details of the Exploit Path

1. **Initial Access**
   Attackers exploit CVE-2021-40822 by leveraging exposed endpoints that do not properly sanitize or restrict input data. This creates an opportunity for injecting malicious payloads or bypassing standard authentication measures.

2. **Privilege Escalation**
   Once initial access is achieved, attackers may elevate their privileges using additional scripts or vulnerabilities present in the environment, gaining administrative control.

3. **Payload Execution**
   With elevated privileges, the threat actor can execute commands remotely, install backdoors, exfiltrate data, or pivot to other parts of the network, significantly amplifying the threat.

4. **Persistence**
   Attackers may establish persistence through scheduled tasks, malware, or other tactics that maintain their foothold even after defensive actions are attempted.

Impacted Systems

This CVE mainly targets cloud-based applications, especially those that handle data processing or have open APIs susceptible to exploitation. The exposure often depends on outdated software versions or configurations lacking recent patches.

Mitigation Strategies

1. **Patch and Update**
   Ensure all affected software is updated to the latest version where the vulnerability is addressed.

2. **Input Validation**
   Implement strict input validation and sanitation mechanisms across all API endpoints.

3. **Access Controls**
   Restrict permissions and enforce least privilege principles to minimize the impact of successful exploitation.

4. **Monitoring and Alerts**
   Utilize monitoring tools to detect abnormal activities or attempts to access privileged areas within the cloud environment.

Top HIGH Severity Exposed Secret

MongoDB Credentials were found to be the most frequently exposed **HIGH** severity secret. These credentials provide access to MongoDB databases, which often store sensitive information such as user data, transaction records, and application configurations. Exposure of these credentials presents a significant risk to cloud environments.

Key Risks

1. **Unauthorized Data Access**
   Attackers can use exposed MongoDB credentials to gain direct access to databases, enabling them to view, modify, or delete sensitive information.

2. **Data Exfiltration**
   Once access is gained, attackers can extract valuable data, potentially leading to regulatory penalties, reputational damage, or financial losses.

3. **Data Manipulation**
   Attackers may insert, update, or delete records, disrupting operations and eroding data integrity.

4. **Service Abuse**
   Exposed credentials could be used to exploit database resources, leading to increased operational costs or service disruptions.

5. **Secondary Exploits**
   With database access, attackers could identify and exploit additional vulnerabilities, such as injecting malicious scripts or escalating privileges.

Mitigation Techniques

1. **Implement Access Controls**
   - Enforce **role-based access control (RBAC)** to ensure users and services have the minimum permissions required.
   - Restrict database access to specific IP addresses or networks using whitelisting.

2. **Secure Credential Management**
   - Store MongoDB credentials securely using secret management solutions like HashiCorp Vault, AWS Secrets Manager, or Azure Key Vault.
   - Avoid hardcoding credentials in source code or configuration files.

3. **Enable Strong Authentication**
   - Use **complex, unique passwords** for MongoDB credentials and rotate them regularly.
   - Implement **multi-factor authentication (MFA)** where possible for database access.

4. **Encrypt Data**
   Ensure all data is encrypted in transit (using TLS) and at rest (using built-in MongoDB encryption features).

5. **Monitor and Audit**
   - Regularly audit database access logs for unauthorized or suspicious activity.
   - Set up alerts for abnormal access patterns, such as connections from unexpected locations.

6. **Patch and Update**
   Keep MongoDB versions up to date with the latest security patches to address known vulnerabilities.

7. **Enable Security Features**
   Use MongoDB's native security features like **IP binding** and **SCRAM-SHA authentication** to further secure access.

Top 5 MEDIUM Severity Verified Exploit Paths		
Vulnerability	**Severity**	**Prevalence**
BlindSSRF-12	MEDIUM	
BlindSSRF-11	MEDIUM	
BlindSSRF-2	MEDIUM	
BlindSSRF-1	MEDIUM	
CVE-2019-1943	MEDIUM	

Top 5 MEDIUM Severity Secrets		
Secret	**Severity**	**Prevalence**
Google API Key	MEDIUM	
Slack Webhook URL	MEDIUM	
AWS RDS	MEDIUM	
Fresdesk Api Key	MEDIUM	
Slack Bot Token	MEDIUM	

The Most Impactful MEDIUM Severity Verified Exploit Path

The most frequent MEDIUM Verified Exploit Path is BlindSSRF-12. This also claims the title of the most commonly found Verified Exploit Path of 2024, narrowly taking it from sister vulnerabilities BlindSSRF-11 and BlindSSRF-2 as well as CVE-2021-40822.

Blind Server-Side Request Forgery (SSRF) vulnerabilities occur when an application can be induced to make back-end HTTP requests to a supplied URL without returning the response to the attacker. This lack of direct feedback makes exploitation more challenging but still potentially harmful.

Exploitation in the Wild

Attackers exploit blind SSRF vulnerabilities by inducing the server to make requests to internal or external systems. Common exploitation techniques include:

1. **Internal Network Scanning**
   Attackers can map internal networks by sending requests to internal IP ranges and observing response behaviors.
2. **Accessing Internal Services**
   Exploiting blind SSRF can allow unauthorized access to internal services not exposed to the internet.
3. **Metadata Service Exploitation**
   In cloud environments, attackers may access metadata services to retrieve sensitive information like AWS credentials.

Mitigation Techniques

Organizations can implement the following measures to mitigate blind SSRF vulnerabilities:

1. **Input Validation**
   Strictly validate and sanitize user inputs that are used in server-side requests.
2. **Allowlist External Domains**
   Restrict server-side requests to a predefined list of trusted external domains.
3. **Disable Unnecessary Protocols**
   Disable protocols that are not required, such as `gopher://`, to reduce the attack surface.
4. **Network Segmentation**
   Isolate internal services to prevent unauthorized access from compromised applications.
5. **Monitor Outbound Traffic**
   Implement monitoring to detect unusual outbound requests that may indicate SSRF attempts.

A Special Word on BlindSSRFs in Relation to GitHub

Given the prevalence of these Verified Exploit Paths and the popularity of the platform for developers, the data highlights attackers aggressively targeting misconfigured webhook endpoints, exposed secrets, and improperly validated user inputs in 2024, a trend that will no doubt continue into 2025. By exploiting BlindSSRF vulnerabilities, attackers can manipulate server-side requests to access internal resources, exfiltrate sensitive data, or escalate privileges within cloud-connected environments. This highlights the critical need for secure webhook configurations, strict input validation, secure secret management and regular audits of repository settings to mitigate these risks in GitHub-based workflows effectively.

Top MEDIUM Severity Exposed Secret

Google API Keys were identified as one of the most frequently exposed **MEDIUM** severity secrets. These keys provide access to various Google Cloud services, such as Maps, YouTube, and Compute Engine, making their exposure a significant security risk.

Key Risks

1. **Service Abuse**
   - Exposed keys can allow attackers to exploit Google Cloud services at the organization's expense, resulting in unexpected charges or service disruptions.
   - Attackers could use APIs like Maps for large-scale queries, leading to quota exhaustion and denial of service for legitimate users.
2. **Data Leakage**
   If the key has permissions for sensitive APIs (e.g., access to storage buckets or databases), attackers could exfiltrate confidential data.

3. **Credential Harvesting**
   An exposed key could provide attackers with additional information about the organization's Google Cloud setup, enabling further targeted attacks.

4. **Reputation Damage**
   Abuse of APIs (e.g., sending spam through YouTube APIs) could harm the organization's reputation and trust with customers.

Mitigation Techniques

1. **Restrict API Key Usage**
   - Set **API key restrictions** by limiting access to specific IP addresses, referring URLs, or applications.
   - Use **service account keys** instead of API keys for accessing critical Google Cloud services.

2. **Implement Quotas**
   Define **usage quotas and limits** for API keys to prevent excessive usage and minimize the impact of abuse.

3. **Secure Key Management**
   - Store API keys securely using secret management tools like Google Secret Manager or third-party solutions like HashiCorp Vault.
   - Avoid embedding API keys in client-side applications or public repositories.

4. **Rotate Keys Regularly**
   - Establish a process for **key rotation** to reduce the window of opportunity for exploitation.
   - Revoke exposed keys immediately and generate new ones with updated restrictions.

5. **Monitor and Audit**
   - Enable Cloud Monitoring to track API usage and detect unusual patterns, such as access from unexpected geographies or excessive usage spikes.
   - Set up alerts for suspicious API activity.

6. **Use OAuth 2.0**
   Replace API keys with **OAuth 2.0 authentication** for services requiring user-level permissions or high-value transactions.

7. **Educate Developers**
   - Train developers on secure coding practices to avoid hardcoding keys in source code.
   - Encourage frequent code reviews to catch accidental exposures early.

Top 5 LOW Severity Verified Exploit Paths		
Vulnerability	**Severity**	**Prevalence**
csp-missing	LOW	
hsts-missing	LOW	
xframe-missing	LOW	
csp-misconfigured	LOW	
nginx-version	LOW	

Top 5 LOW Severity Secrets		
Secret	**Severity**	**Prevalence**
AWS RDS	LOW	
RazorPay	LOW	
RSA private key	LOW	
JWT Token	LOW	
Bitly Access Token	LOW	

The Most Impactful LOW Severity Verified Exploit Path

The most impactful LOW Verified Exploit Path is “CSP-Missing”. The “CSP-Missing” threat typically refers to a lack of a properly implemented Content Security Policy (CSP) on a web application. A CSP is a security measure designed to mitigate a range of attacks such as Cross-Site Scripting (XSS) by controlling the sources from which resources (e.g., scripts, images) can be loaded. Without a CSP, attackers might more easily inject malicious scripts into a website, compromising user data or leading to other exploits.

Key Points on CSP

1. **CSP Implementation**
   A CSP is best implemented via HTTP response headers, allowing site administrators to define permitted content sources. The `Content-Security-Policy` header is the standard method for this.

2. **Enforcement and Reporting**
   The `Content-Security-Policy-Report-Only` header is also available, allowing the policy to run in a non-blocking mode for testing. This helps identify potential issues without disrupting site functionality.

3. **Security Advantages**
   Deploying a strong CSP offers defense-in-depth, making it significantly harder for attackers to exploit vulnerabilities like XSS.

4. **Policy Types**
   Modern best practices recommend “strict” CSPs that use mechanisms like nonces or hashes for safer, dynamic script handling.

Top LOW Severity Exposed Secret

AWS RDS Credentials are another critical secret that, if exposed, poses significant security risks. These credentials provide access to Amazon Relational Database Service (RDS) instances, which often store sensitive organizational and customer data. Their exposure can lead to severe consequences for both security and operations.

Key Risks

1. **Unauthorized Database Access**
   - Exposed credentials allow attackers to directly access the RDS database, potentially gaining read, write, or administrative privileges.
   - This could lead to data theft, deletion, or unauthorized modifications.
2. **Data Exfiltration**
   Attackers can export sensitive information, such as personally identifiable information (PII), intellectual property, or financial records, leading to compliance violations and reputational damage.
3. **Data Corruption or Manipulation**
   Malicious actors may modify or delete critical data, causing operational disruptions or data integrity issues.
4. **Service Abuse and Cost Implications**
   Attackers could exploit the database resources for their purposes, driving up AWS costs for the organization.
5. **Lateral Movement**
   Once inside the database, attackers might uncover additional information to move laterally within the AWS environment, escalating privileges and compromising more assets.

Mitigation Techniques

1. **Secure Credential Management**
   - Use AWS Secrets Manager or Systems Manager Parameter Store to securely store and manage RDS credentials.
   - Avoid hardcoding credentials in application code or configuration files.
2. **Enable IAM Database Authentication**
   Replace traditional database credentials with AWS Identity and Access Management (IAM) authentication for connecting to RDS instances.
3. **Restrict Access**
   - Limit database access to specific trusted IP addresses or subnets using RDS security groups.
   - Enforce role-based access control (RBAC) to ensure users and applications only have the minimum required permissions.
4. **Encrypt Data**
   - Use RDS encryption at rest to secure stored data.
   - Ensure TLS encryption is enabled for all data transmitted to and from the database.

5. **Enable Monitoring and Alerts**
    - Use Amazon CloudWatch to monitor database activity and set up alerts for unusual access patterns or spikes in usage.
    - Enable RDS Enhanced Monitoring to track system-level metrics in real-time.
6. **Audit and Rotate Credentials**
    - Regularly audit RDS credentials and rotate them periodically to minimize the risk of misuse.
    - Immediately revoke any credentials suspected of being exposed.
7. **Implement Least Privilege Principles**
    - Avoid using root or master database credentials for application access.
    - Create separate credentials with limited permissions for different applications or services.
8. **Database Backup and Recovery**
    - Enable automatic backups in RDS to ensure quick recovery in case of data loss or compromise.
    - Periodically test the recovery process to ensure data integrity.

Conclusion

The 2025 Cloud Verified Exploit Paths and Secrets Scanning Threat Report underscores the evolving complexity and interconnectedness of modern cloud security challenges. By combining insights from verified exploit paths and secrets scanning, this report provides details and understanding amongst some of the most pressing risks in cloud environments.

The findings reveal that attackers continue to exploit both technical vulnerabilities and exposed credentials to compromise cloud infrastructures. From high-severity vulnerabilities like CVE-2021-40822 to critical secrets exposures such as AWS Keys, these threats highlight the importance of proactive and layered security strategies.

Key Takeaways From this Report Include:

- **The Prevalence of High-Risk Threats**
  AWS Keys and MongoDB Credentials were among the most exposed secrets, representing significant risks to cloud environments. Similarly, critical vulnerabilities like CVE-2020-35489 enable attackers to exploit improper input validation in widely-used systems.
- **Severity Spanning All Categories**
  Threats across all severity levels, including low-severity issues such as CSP misconfigurations, can serve as entry points for attackers and should not be ignored.
- **Integrated Risk Management**
  Cloud security is complex so having a unified platform that gives visibility and real-time response addressing all aspects of multi-cloud environments is essential for a robust cloud security posture.

Recommended Next Steps

1. **Prioritize Remediation**
   Focus on addressing the most prevalent and impactful vulnerabilities and secret exposures identified in this report.

2. **Adopt Best Practices**
   Implement key management, patching, and access control strategies to mitigate risks.

3. **Monitor and Adapt**
   Continuously monitor for new threats and update security measures to address evolving attack techniques.

4. **Invest in Automation**
   Leverage tools like SentinelOne's Verified Exploit Paths™ and secrets scanning solutions to prioritize, detect and mitigate threats in real-time.

5. **Educate Teams**
   Ensure that all stakeholders, from developers to your security peers, understand their role in securing the cloud environment.

SentinelOne

Contact Us

sales@sentinelone.com
+1-855-868-3733

sentinelone.com

About SentinelOne

SentinelOne (NYSE:S) is pioneering autonomous cybersecurity to prevent, detect, and respond to cyber attacks faster and with higher accuracy than ever before. Our Singularity XDR platform protects and empowers leading global enterprises with real-time visibility into attack surfaces, cross-platform correlation, and AI-powered response. Achieve more capability with less complexity.