IBM Security

是什麼讓Slack長了蟲？

What makes Slack vulnerable to Blind SSRF attack

Luke Chen

2021/5

IBM

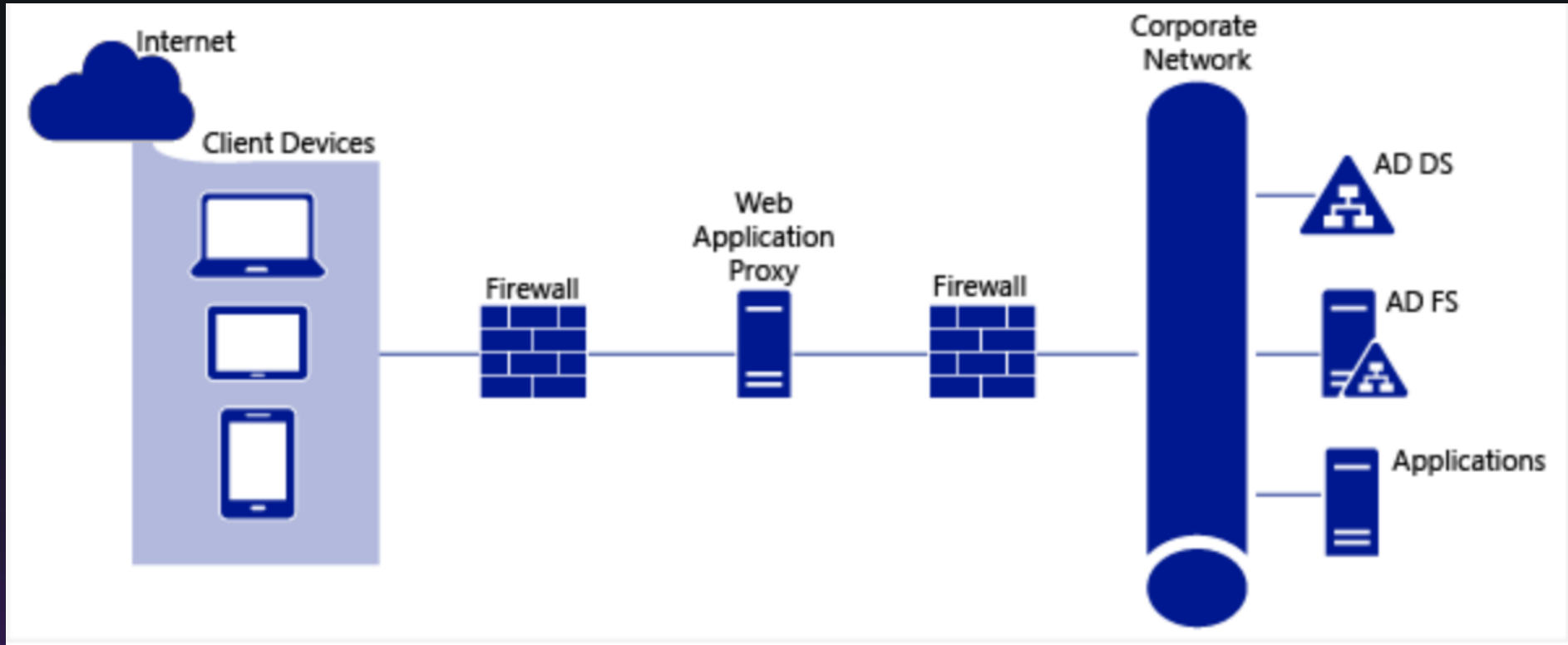
Luke Chen

- dad of 2 kids
- software engineer in IBM
- white-hat hacker
- open-source contributor

Before we started

Before we started

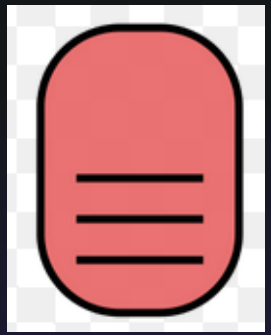
Let’s test a check IP service



Source: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/dn383650(v=ws.11)

- X-Forwarded-For – **IP address** the client connected to on the proxy (e.g. **1.2.3.4**)

User



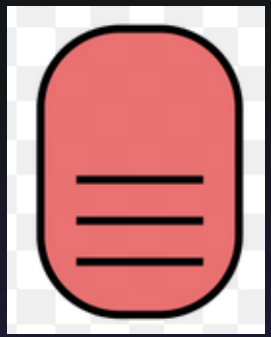
Proxy Server

Application Server

Host: app.server.com

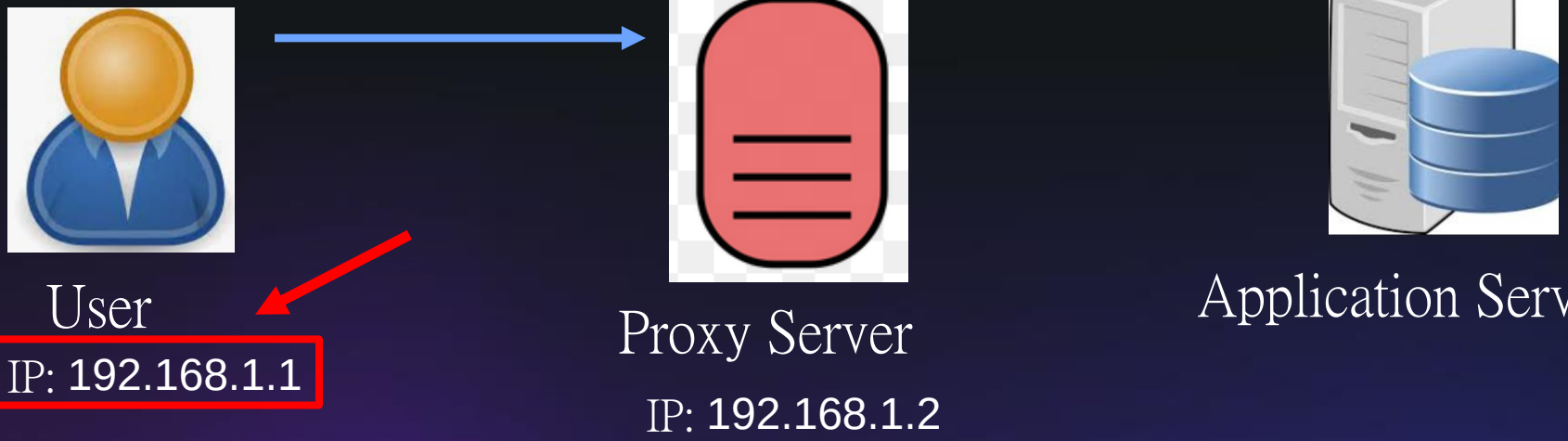
User

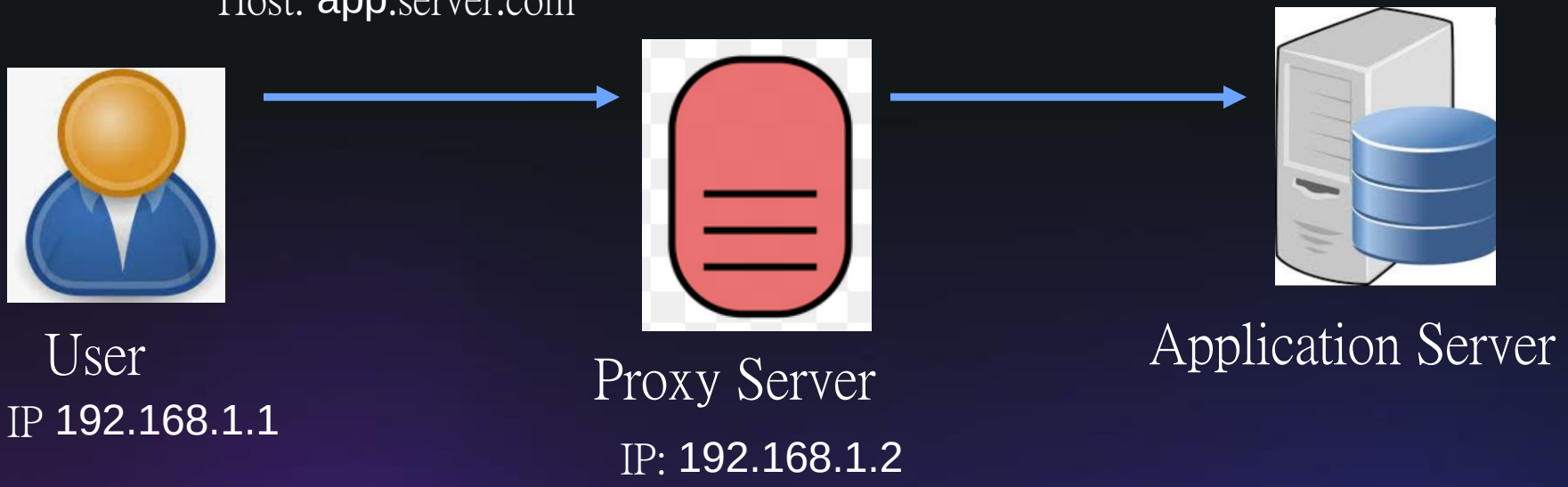
IP: 192.168.1.1



Proxy Server

Application Server


Host: app.server.com
User
IP: 192.168.1.1
Proxy Server
IP: 192.168.1.2
Application Server


Host: app.server.com
Host: internal.app.server.com
X-Forwarded-For: 192.168.1.1
User
IP 192.168.1.1
Proxy Server
IP: 192.168.1.2
Application Server

U

IP: 1

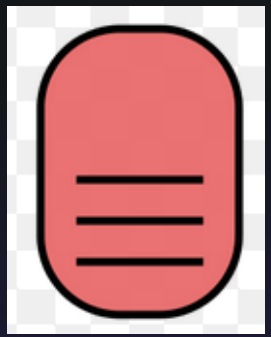
rver

```
192.168.1.1 - - [26/Apr/2021:15:08:27 +0800] "GET / H
192.168.1.1 - - [26/Apr/2021:15:08:27 +0800] "GET /cs
192.168.1.1 - - [26/Apr/2021:15:08:28 +0800] "GET /js
192.168.1.1 - - [26/Apr/2021:15:08:28 +0800] "GET /as
192.168.1.1 - - [26/Apr/2021:15:08:29 +0800] "GET /js
192.168.1.1 - - [26/Apr/2021:15:08:30 +0800] "GET /fa
192.168.1.1 - - [26/Apr/2021:15:08:35 +0800] "POST /
192.168.1.1 - - [26/Apr/2021:15:08:41 +0800] "POST /
192.168.1.1 - - [26/Apr/2021:15:08:49 +0800] "POST /
192.168.1.1 - - [26/Apr/2021:15:08:57 +0800] "POST /
192.168.1.1 - - [26/Apr/2021:15:08:57 +0800] "GET / H
192.168.1.1 - - [26/Apr/2021:15:08:58 +0800] "GET /js
192.168.1.1 - - [26/Apr/2021:15:08:58 +0800] "GET /cs
192.168.1.1 - - [26/Apr/2021:15:08:58 +0800] "GET /cs
192.168.1.1 - - [26/Apr/2021:15:08:59 +0800] "GET /js
192.168.1.1 - - [26/Apr/2021:15:09:01 +0800] "GET /js
192.168.1.1 - - [26/Apr/2021:15:09:04 +0800] "GET /js
192.168.1.1 - - [26/Apr/2021:15:09:06 +0800] "GET /re
192.168.1.1 - - [26/Apr/2021:15:09:06 +0800] "GET /as
192.168.1.1 - - [26/Apr/2021:15:09:06 +0800] "GET /re
```

But, what if user sends XFF…

Host: app.server.com

X-Forwarded-For: 192.168.1.10



User

IP: 192.168.1.1

Proxy Server

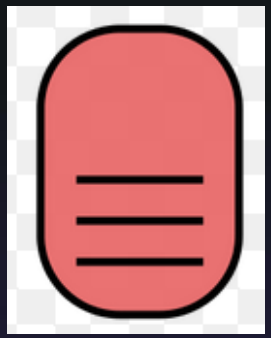
Application Server

Host: app.server.com

X-Forwarded-For: 192.168.1.10

User

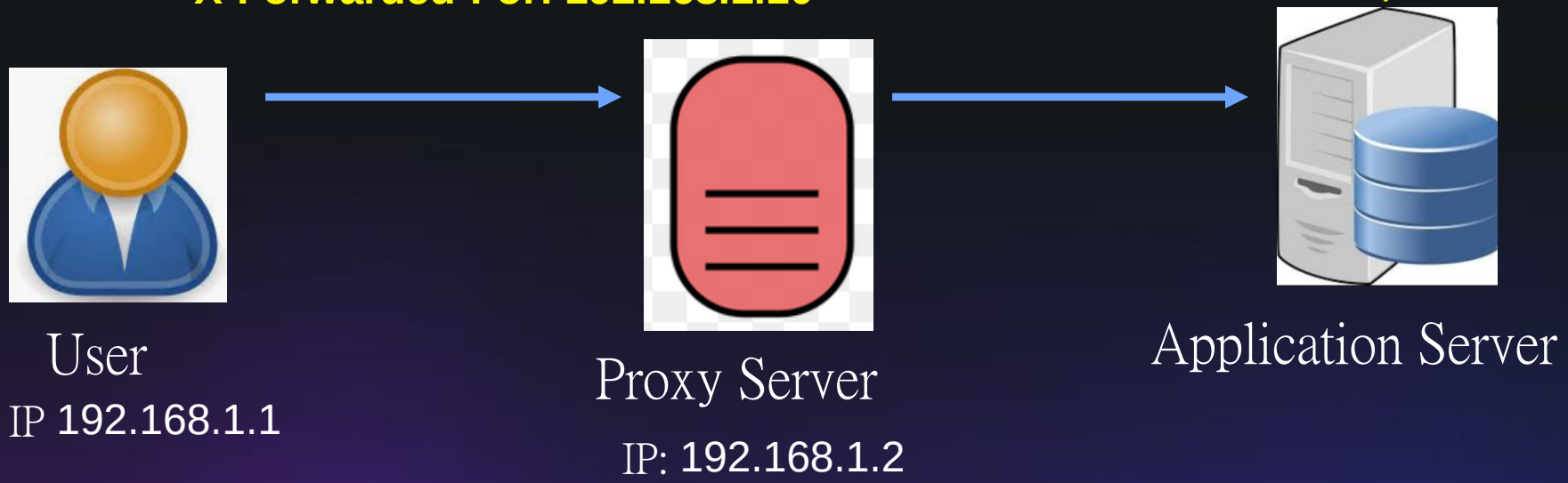
IP: 192.168.1.1



Proxy Server

IP: 192.168.1.2

Application Server


Host: app.server.com
X-Forwarded-For: 192.168.1.10
Host: internal.app.server.com
X-Forwarded-For:
192.168.1.10, 192.168.1.1
User
IP 192.168.1.1
Proxy Server
IP: 192.168.1.2
Application Server

U

IP:

```
192.168.1.10 - - [26/Apr/2021:15:08:27 +0800] "GET / HT
192.168.1.10 - - [26/Apr/2021:15:08:27 +0800] "GET /css
192.168.1.10 - - [26/Apr/2021:15:08:28 +0800] "GET /js/
192.168.1.10 - - [26/Apr/2021:15:08:28 +0800] "GET /ass
192.168.1.10 - - [26/Apr/2021:15:08:29 +0800] "GET /js/
192.168.1.10 - - [26/Apr/2021:15:08:30 +0800] "GET /fav
192.168.1.10 - - [26/Apr/2021:15:08:35 +0800] "POST / H
192.168.1.10 - - [26/Apr/2021:15:08:41 +0800] "POST / H
192.168.1.10 - - [26/Apr/2021:15:08:49 +0800] "POST / H
192.168.1.10 - - [26/Apr/2021:15:08:57 +0800] "POST / H
192.168.1.10 - - [26/Apr/2021:15:08:57 +0800] "GET / HT
192.168.1.10 - - [26/Apr/2021:15:08:58 +0800] "GET /js/ver
192.168.1.10 - - [26/Apr/2021:15:08:58 +0800] "GET /css
192.168.1.10 - - [26/Apr/2021:15:08:58 +0800] "GET /css
192.168.1.10 - - [26/Apr/2021:15:08:59 +0800] "GET /js/
192.168.1.10 - - [26/Apr/2021:15:09:01 +0800] "GET /js/
192.168.1.10 - - [26/Apr/2021:15:09:04 +0800] "GET /js/
192.168.1.10 - - [26/Apr/2021:15:09:06 +0800] "GET /res
```

Source: https://media.giphy.com/media/BtX1KVvkHPp7i/giphy.gif

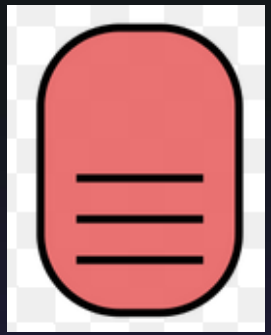
And the other X-Forwarded headers

- X-Forwarded-Port -- **port** the client connected to on the proxy (e.g. **80**, **443**)

- X-Forwarded-Proto -- **protocol** the client used to connect to the proxy (**http, https**)

- X-Forwarded-Host -- content of the **Host header** the client sent to the proxy.

User

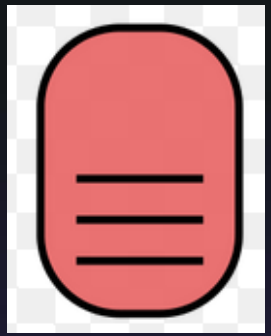


Proxy Server

Application Server

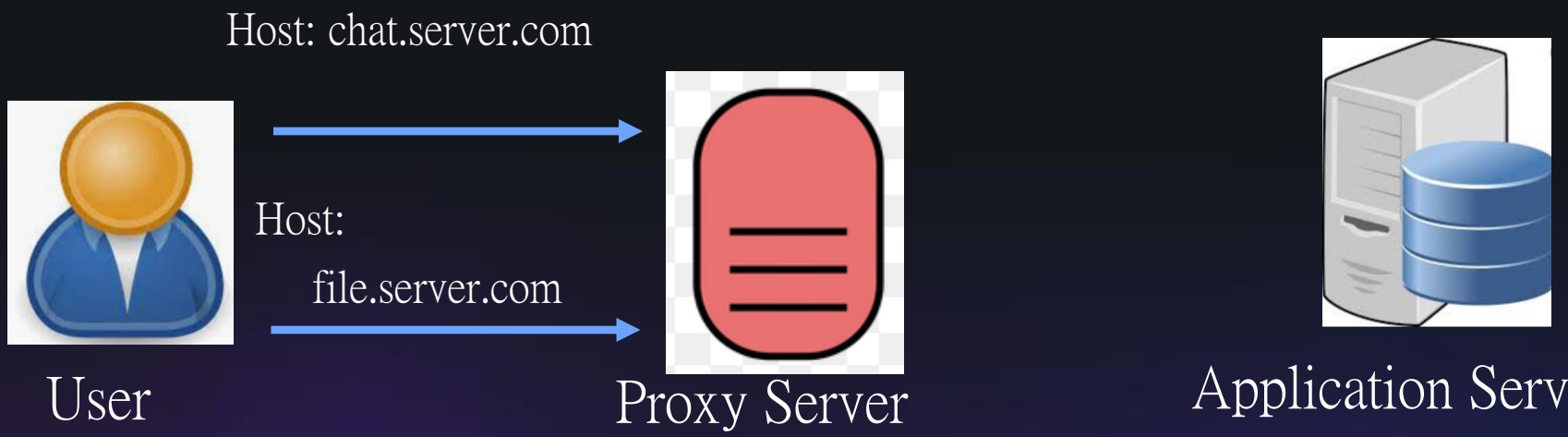
Host: chat.server.com

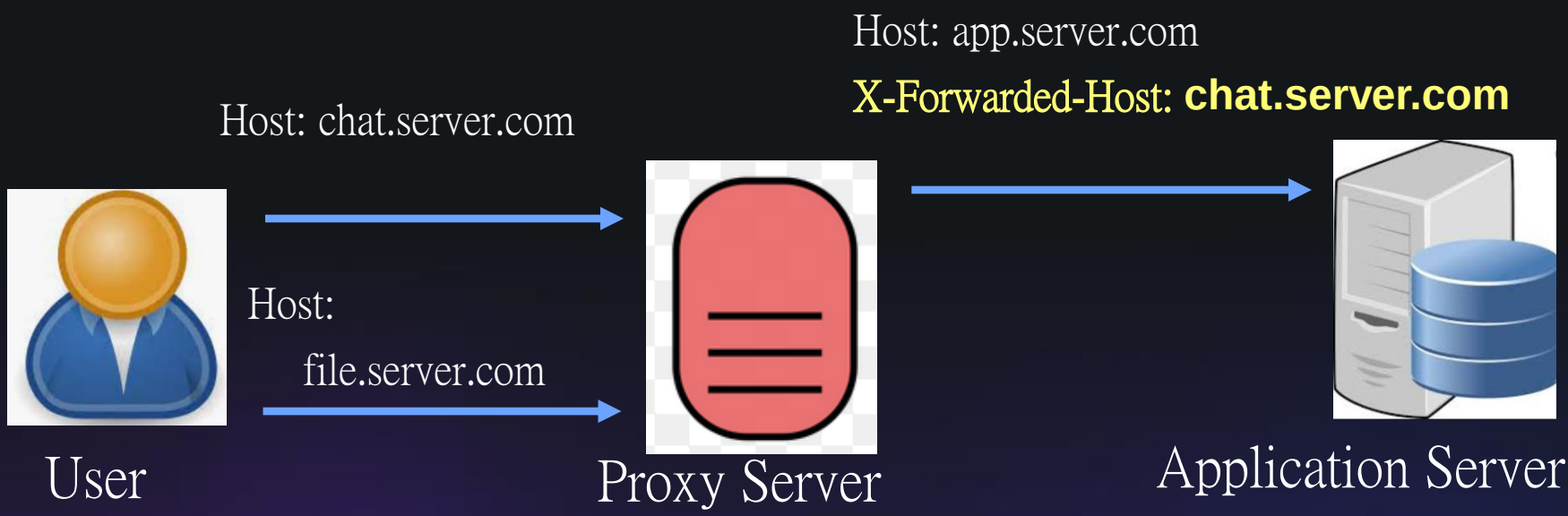
User

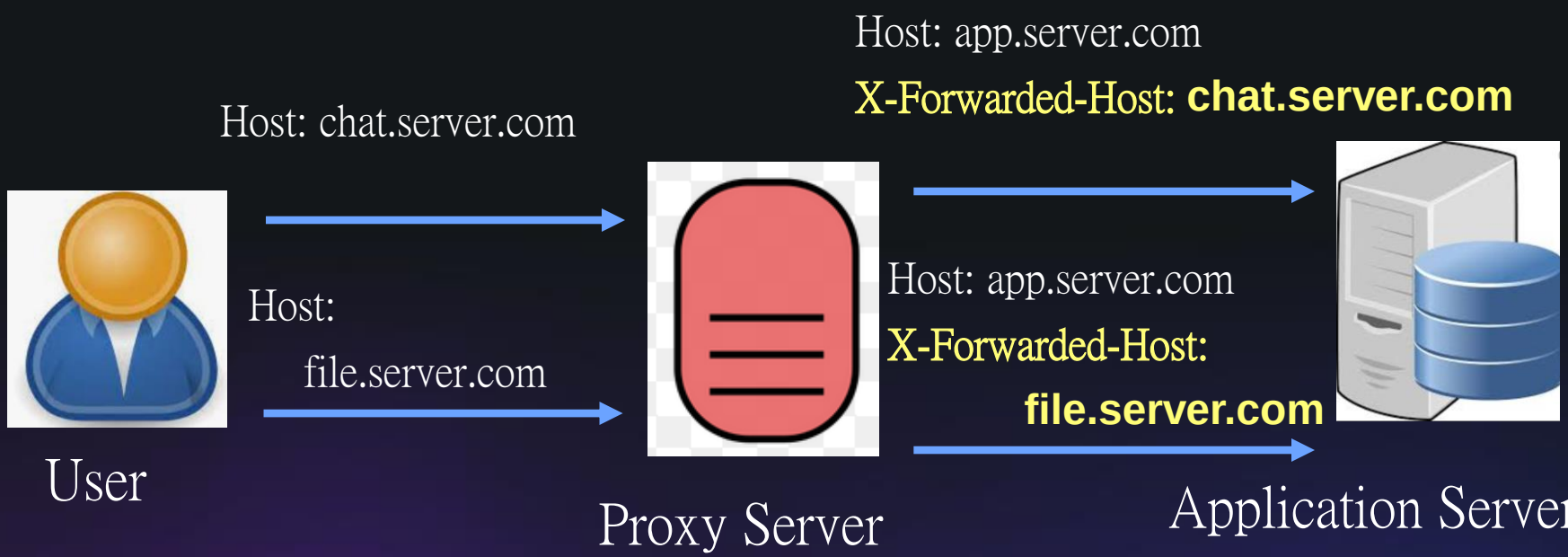


Proxy Server

Application Server


Host: chat.server.com
Host:
file.server.com
User
Proxy Server
Application Server


Host: app.server.com
X-Forwarded-Host: chat.server.com
Host: chat.server.com
Host:
file.server.com
User
Proxy Server
Application Server


Host: chat.server.com
Host:
file.server.com
User
Proxy Server
Host: app.server.com
X-Forwarded-Host: chat.server.com
Host: app.server.com
X-Forwarded-Host:
file.server.com
Application Server

Host: app.server.com

X-Forwarded-Host: **file.server.com**

Application Server

Host: app.server.com

X-Forwarded-Host: **file.server.com**

Application Server

Host: file.server.com

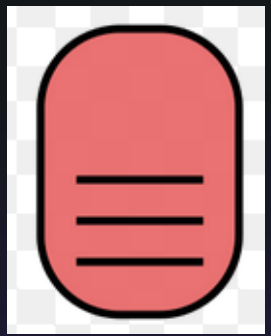
File Server

But, what if user sends XFH…

Host: file.server.com

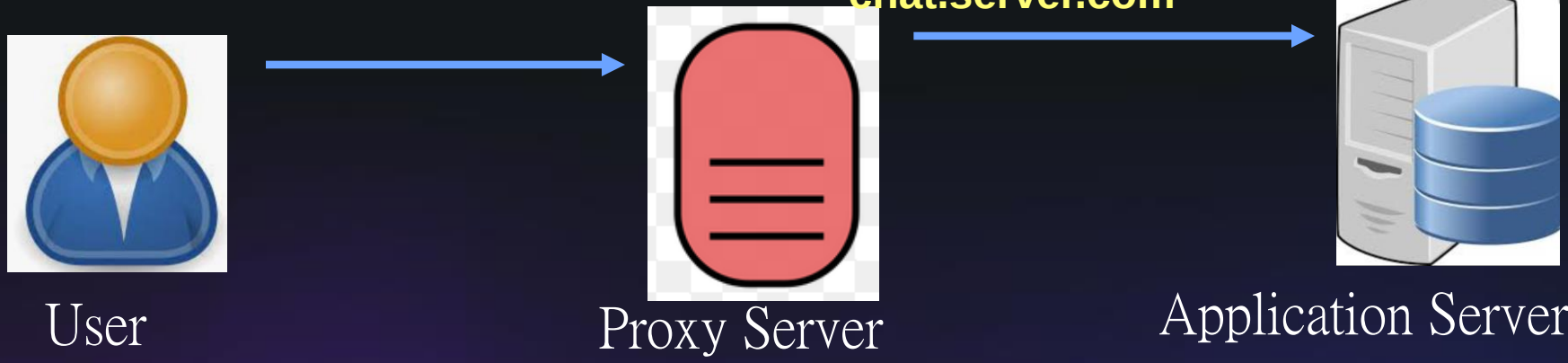
X-Forwarded-Host: evil.com

User



Proxy Server

Application Server


Host: file.server.com
X-Forwarded-Host: evil.com
Host: app.server.com
X-Forwarded-Host:
evil.com,
chat.server.com
User
Proxy Server
Application Server

Host: app.server.com

X-Forwarded-Host:

evil.com, file.server.com

Application Server

Host: evil.com

Let’s add RegEx to protect ourselves…

Host: app.server.com

X-Forwarded-Host:

evil.com, file.server.com

Application Server

Regex: /^file\.server\.com/

Host: app.server.com

X-Forwarded-Host:

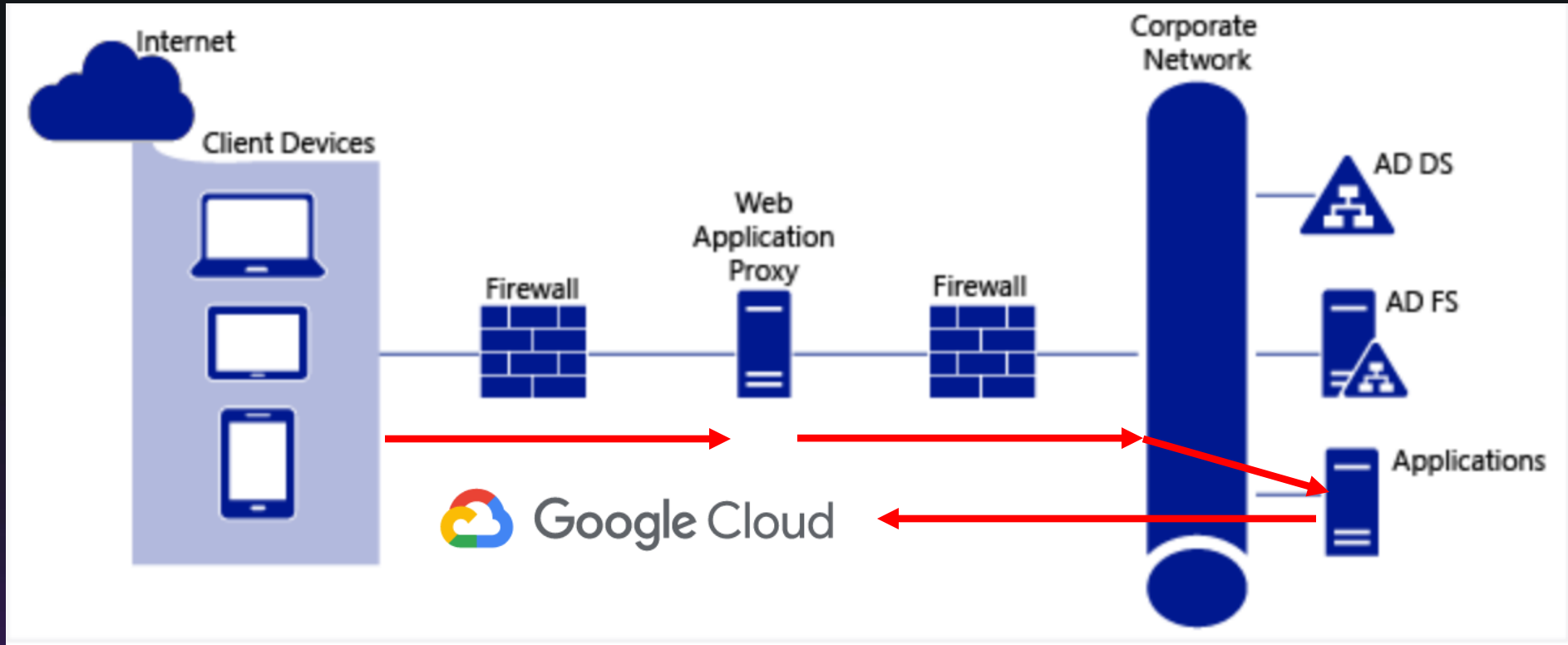
evil.com, file.server.com

Application Server

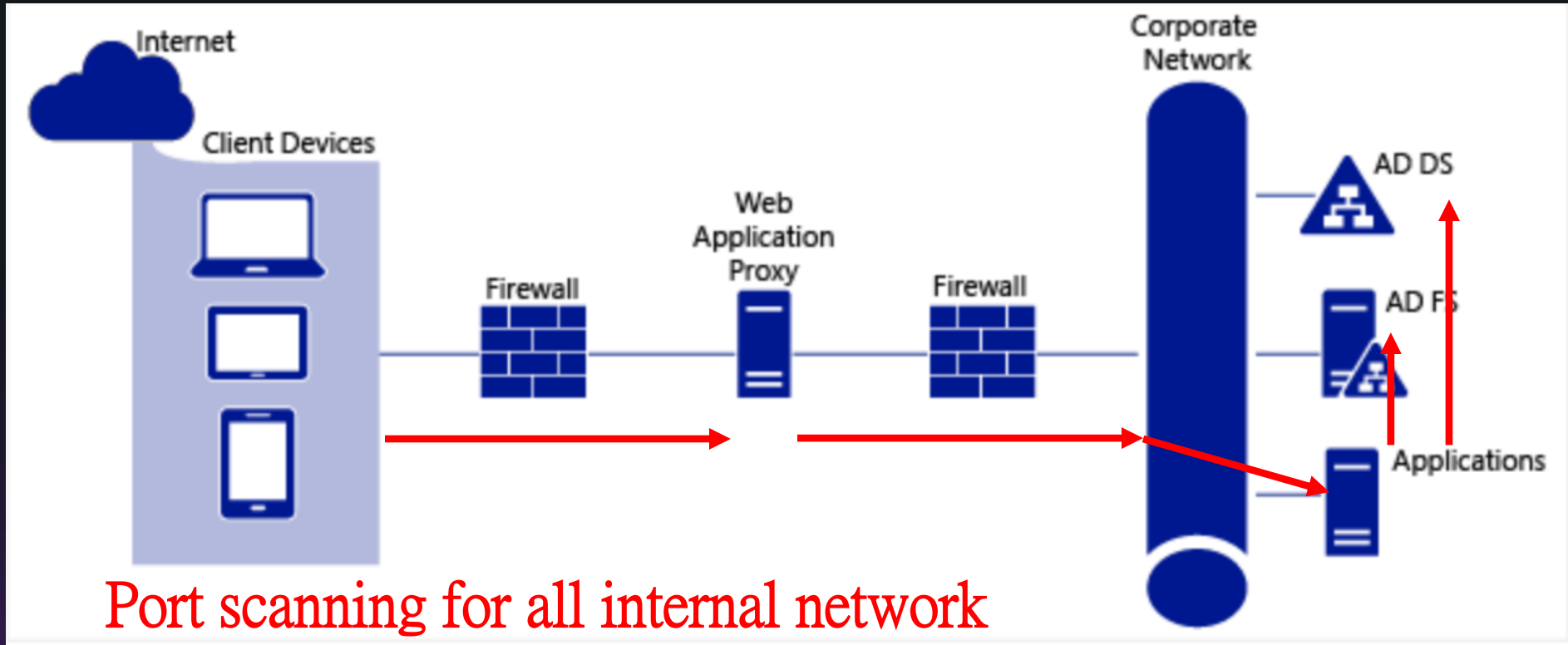
Host: evil.com

Regex: /^file\.server\.com/

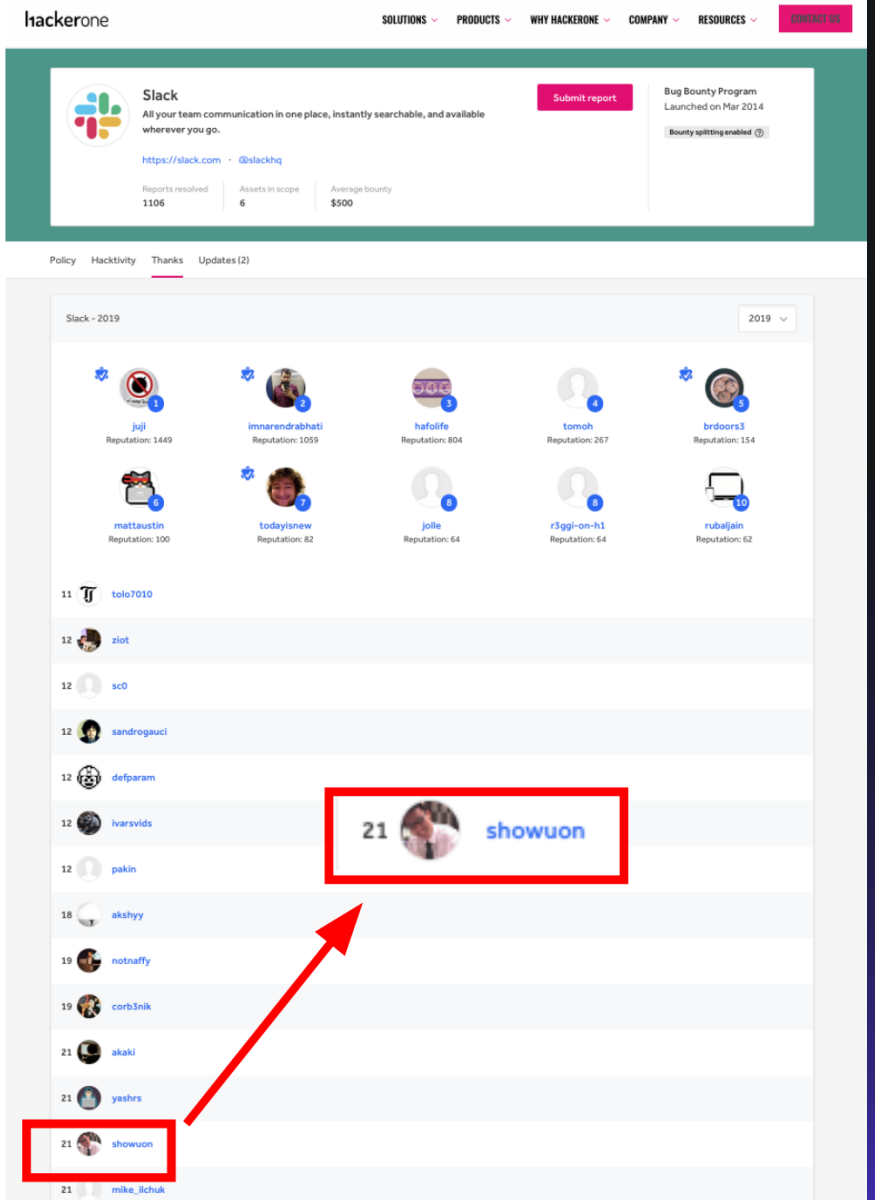
Does it really work?



Source: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/dn383650(v=ws.11)



Source: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/dn383650(v=ws.11)


hackerone
SOLUTIONS
PRODUCTS
WHY HACKERONE
COMPANY
RESOURCES
CONTACT US
Slack
All your team communication in one place, instantly searchable, and available wherever you go.
https://slack.com · @slackhq
Reports resolved 1106
Assets in scope 6
Average bounty $500
Submit report
Bug Bounty Program
Launched on Mar 2014
Bounty splitting enabled
Policy
Hacktivity
Thanks
Updates (2)
Slack - 2019
2019
juji Reputation: 1449
imnarendrabhati Reputation: 1059
hafolife Reputation: 804
tomoh Reputation: 267
brdoors3 Reputation: 154
mattaustin Reputation: 100
todayisnew Reputation: 82
jolle Reputation: 64
r3ggi-on-h1 Reputation: 64
rubaljain Reputation: 62
11 tolo7010
12 ziot
12 sc0
12 sandrogauci
12 defparam
12 ivarsvids
12 pakin
18 akshyy
19 notnaffy
19 corb3nik
21 akaki
21 yashrs
21 showuon
21 mike_ilchuk
21 showuon

REF: https://github.com/rails/rails/issues/29893

5. To simulate a host header attack, paste the curl command into a terminal and add `-H 'X-Forwarded-Host: evil.com'`. For example, for one of my requests:

```
Referrer: http://forwarded-host-demo.herokuapp.com/' -H 'X-Forwarded-Host: evil.com' --compres
```

Expected Result

User is redirected to the home page:

```
<html><body>You are being <a href="http://forwarded-host-demo.herokuapp.com/">redirected</a>.</
```

Observed Result

User is redirected to the home page on evil.com:

```
<html><body>You are being <a href="http://evil.com/">redirected</a>.</body></html>
```

Merged

Guard against DNS rebinding attacks by permitting hosts #33145

eileencodes merged 2 commits into `rails:master` from `gsamokovarov:host-authorization` on Dec 18, 2018

```
Rails.application.config.hosts = [
  IPAddr.new("0.0.0.0/0"), # All IPv4 addresses.
  IPAddr.new("::/0"),      # All IPv6 addresses.
  "localhost"              # The localhost reserved domain.
]
```

In other environments `Rails.application.config.hosts` is empty and no `Host` header checks will be done. If you want to guard against header attacks on production, you have to manually whitelist the allowed hosts with:

```
Rails.application.config.hosts << "product.com"
```

REF: https://github.com/rails/rails/pull/33145

So… how to mitigate this attack for ourselves?

- Protect on proxy server
- Protect on service application server

- Protect on proxy server
  - Replace with real client IP/Host

– Ref: https://techdocs.f5.com/en-us/bigip-15-0-0/big-ip-local-traffic-management-getting-started-with-policies/example-preventing-a-spoof-of-an-x-forwarded-for-request.html

Preventing a spoof of an x-forwarded-for request: iRules example

This topic provides an example of iRules code that is equivalent to a policy that prevents a spoof of an x-forwarded-for request. This is a situation where attackers might attempt to thwart security by falsifying the IP address in a header, and pass it through the BIG-IP system. This example replaces a request that includes an x-forwarded-for header with the actual client IP address.

```
when HTTP_REQUEST {
    set xff 0
    foreach x [HTTP::header names] {
        if { [string tolower $x] equals "x-forwarded-for" } {
            set xff 1
            HTTP::header remove $x
            HTTP::header insert X-FORWARDED-FOR [IP::client_addr]
        }
    }
    if { $xff == 0 } {
        HTTP::header insert X-FORWARDED-FOR [IP::client_addr]
    }
}
```

- Protect on service application server
  - Better RegEx protection for XFH
  - Don’t just get the 1st IP from the list

Application Server

Regex: /^(file|chat)\.server\.com$/

```
X-Forwarded-For: <fake>, <client>, <proxy1>, <proxy2>
```

- Don’ t just get the 1st IP from the list
- Find the latest unknown IP (assume proxy IP are known)

Takeaways

- X-Forwarded-For – e.g. **1.2.3.4**
- X-Forwarded-Port – e.g. **80**, **443**
- X-Forwarded-Proto – e.g. **http, https**
- X-Forwarded-Host – e.g. **example.com**

Thank you!

 /in/showuon/

 showuon@gmail.com