Shelby Hagemann
Sareh Assiri
CYB 410: Name of Class
4 December 2022

Assignment 3, Part 3: The Dangers of Domain Name System Poisoning

One very important aspect of web programming and web servers is the Domain Name System, or DNS. The Domain Name System is a "hierarchical naming system built on a distributed database for computers, services, or any resource connected to the internet or a private network" [1]. This feature is vastly important in that it allows a web server to connect to a website, associating its domain name with its IP address. The way that DNS operates is it searches for its stored records for a given, requested domain name. Then, it can retrieve the domain name's corresponding IP address. When someone enters a URL into their web browser, their computer contacts a Domain Name System to retrieve the website's data. As mentioned before, the Domain Name System then searches through its database to find the necessary information for the computer to reach the desired website. Once the computer reaches this website, the Domain Name System will store data pertaining to the computer in its own server.

Unfortunately, there are several types of attacks specifically designed to steal data through the Domain Name System. One of these attacks, which will be discussed in this essay, is Domain Name System Poisoning, or DNS Poisoning for short. Domain Name System Poisoning entails manipulating "known vulnerabilities within the domain name system (DNS). When [the attack] is completed, a hacker can reroute traffic from one site to a fake version" [2]. This type of attack involves caching. A cache is "a high-speed data storage layer which stores a subset of data, typically transient in nature, so that future requests for that data are served up faster than is possible by accessing the data's primary storage location" [3]. While caching can be incredibly useful in web development and accessing websites as a general, it can also be quite dangerous in regards to having important data and information be leaked.

There are several ways in which a Domain Name System Poisoning attack can occur. For example, this attack can occur by "impersonating a server…Tying up the server…[or[Exploiting open ports" [2]. If there are not security measures in place, this form of a Domain Name System

attack can occur quite easily. A common form of a Domain Name System Poisoning attack is phishing, where a hacker creates a fake website designed to look identical to a real website. The page that is generally used on the fake website for an attack typically involves some form of a login system that prompts the user to input their credentials, which in turn get stolen by the hacker.

This heinous form of a Domain Name System attack can be held against a multitude of programs, servers, and websites. For example, as of May 2022 a very commonly used C library possessed security vulnerabilities that made it a prime example of what is at risk of a Domain Name System Poisoning attack. This issue was found in the uClibc library for the C programming language. This is a standard library that "focuses specifically on embedded systems" [4] and is used often in C programming. This library in particular is "intended for Linux kernel-based operating systems for embedded systems and mobile devices" [4]. It is clear that this library is very broadly used and can potentially work with sensitive information. uClibc also "provides an extensive DNS client interface that allows programs to readily perform lookups and other DNS-related requests" [4], meaning it makes a perfect candidate for falling victim to a Domain Name System attack.

Transaction IDs are often used by Domain Name Systems and can actually aid in preventing attacks and vulnerabilities. When transaction identification numbers are kept as random as possible, it is significantly more difficult for hackers to break through security systems and steal data. Several researches discovered the uClibc vulnerability for DNS poisoning when they realized that the transaction ID for this library is, in fact, never random. Attached below is a screenshot of the code that contains this vulnerability [4]. As described my Malwarebytes Labs, "at first the transaction ID is incremental, then it resets to the value 0x2, then it is incremental again" [4].

```
1307                    /* first time? pick starting server etc */
1308                    if (local_ns_num < 0) {
1309                            local_id = last_id;
1310    /*TODO: implement /etc/resolv.conf's "options rotate"
1311     (a.k.a. RES_ROTATE bit in _res.options)
1312                            local_ns_num = 0;
1313                            if (_res.options & RES_ROTATE) */
1314                                    local_ns_num = last_ns_num;
1315                            retries_left = __nameservers * __resolv_attempts;
1316                    }
1317                    retries_left--;
1318                    if (local_ns_num >= __nameservers)
1319                            local_ns_num = 0;
1320                    local_id++;
1321                    local_id &= 0xffff;
1322                    /* write new values back while still under lock */
1323                    last_id = local_id;
1324                    last_ns_num = local_ns_num;
1325                    /* struct copy */
1326                    /* can't just take a pointer, __nameserver[x]
1327                     * is not safe to use outside of locks */
1328                    sa = __nameserver[local_ns_num];
1329                    __UCLIBC_MUTEX_UNLOCK(__resolv_lock);
1330
1331                    memset(packet, 0, PACKETSZ);
1332                    memset(&h, 0, sizeof(h));
1333
1334                    /* encode header */
1335                    h.id = local_id;
1336                    h.qdcount = 1;
1337                    h.rd = 1;
```

As a result of this issue in the code, the transaction ID will always be the same, allowing hackers who figure this trick out to take advantage of the vulnerability and gain unauthorized access to the system. In order to utilize this vulnerability, the attacker would simply need to custom tailor their DNS poisoning attack to focus on the transaction ID.

This vulnerability is quite dangerous and surely violates each of the CIA components that have been discussed in this class. Because a hacker could easily gain unauthorized access to the system once they discover the simplicity of predicting what the transaction ID is, this security issue violates confidentiality. The hacker could work their way into the system and gain access to sensitive information and data. Because of how easily a hacker could thwart this kind of system, they could grant themselves administrative access, which would then give them the permissions to make changes to it, then in turn violating both integrity and availability. This would violate

integrity in that the information within a system and its functions could be changed. This would also violate availability in that any changes made could cause the system to not operate the way it is expected, potentially even going offline.

Part of my difficulty in doing this assignment was writing a harness that fits the needs of this specific vulnerability. It is important that this type of harness searches for duplicated lines of code and variable assignments within the program. A large issue with this vulnerability is the fact that the transaction identification number is being reassigned multiple times, the subsequent instances being what set this imperative number to a value of 0x2. A fuzzing tool would need to be able to recognize that this variable for the transaction ID is being reset on three separate occasions among seven lines of code.

Sources

[1] Infoblox. 2022. What is Domain Name System (DNS)? (2022). Retrieved December 4, 2022 from https://www.infoblox.com/glossary/domain-name-system-dns/#:~:text=Domain%20Name%20System%20(DNS)%20is,Internet%20or%20a%20private%20network.

[2] OKTA. 2022. DNS Poisoning (DNS Spoofing): Definition, Technique, and Defense. (2022). Retrieved December 3, 2022 from https://www.okta.com/identity-101/dns-poisoning/#:~:text=DNS%20poisoning%20is%20a%20hacker,the%20way%20the%20DNS%20works.

[3] Amazon AWS. 2022. Caching Overview: What is Caching? (2022). Retrieved December 3, 2022 from https://aws.amazon.com/caching/#:~:text=In%20computing%2C%20a%20cache%20is,the%20data's%20primary%20storage%20location.