**LOCAL DNS HEALTH CHECKER**

**Proposal for CSE508  Network Security Course Design Project**

**SAMPATHKUMAR KILAPARTHI | 112079198**

In the recent years, interest in the security and resilience of DNS has grown significantly. Even in the areas that are far more remotely involved with DNS itself. This is because of the active role DNS plays in the functioning of the internet and various internet-based attacks. A DNS lookup is therefore a prerequisite for most Internet transactions. This means that the DNS is not only crucial to the operation of the Internet, but DNS is in fact a single point of failure for most Internet transactions. Unfortunately, the DNS ecosystem has slowly evolved from a simple system at its inception to a vastly distributed, complex and brittle system today. The current state of DNS is not befitting of a service that is a single point of failure for most Internet transactions.

The DNS is a critical infrastructure that everyone use and the key component on the internet. And yet it is one of the most insecure, vulnerable components in this infrastructure. Due to this importance, there is a strong demand for securing communication within the DNS system. When compromised it may lead to catastrophes. A simple attack like DNS spoofing allows an attacker to manipulate DNS responses and propagate malicious DNS tables all over the Internet. This is because of an incorrect design assumption that DNS resolution’s malignity. There exists a need to prevent attacks on DNS namely DNS hijacking, DNS exploits, DNS DDoS, Data Exfiltration via DNS etc. And hypothetically, if an attacker is able to impersonate a DNS server nothing can be done to reverse any malicious changes that come under the purview of any authoritative DNS sub-system.

**Prototype**

The primary idea of this approach is to check if there is any DNS poisoning. So whenever a new request is fired by the browser, We evaluate the DNS lookups by cross checking them with the previous IP results. If they differ, we then alert the user regarding the possibility of DNS poisoning with a dialog. The extension will remember all the previously resolved domain name IPs. So if for any reason, these DNS servers are not reachable or if DNS server goes down or if our IP is blocked, this extension will provide us a backup which contains the list of all resolved IP addresses. Additionally it provides us with statistics of all visited sites. We can extend this plugin to monitor network traffic using Chrome APIs.

In this chrome extension, with every DNS hit the browser remembers the resolved IP. It stores these IP addresses returned by the DNS server into the machine in a LOFI( Last one First Input) order . These lists are then stored locally usually in C:/Windows/System32/drivers/etc/hosts on windows and in /etc/hosts on Linux machines based on the OSes the browser runs on. So everytime the browser fires a request, a corresponding DNS lookup is performed. This IP value is then cross checked with the previous DNS resolution. If it doesn’t match with the existing IP address present, we find the longest common prefix of all the previous DNS lookups. If it doesn’t meet the threshold, a warning dialog will be displayed. On the other hand, if it meets the threshold, we then add it to the Cache List.

To avoid clogging the user with alerts, the plugin remembers the alert state and won’t display the same alert twice. However, it resets once the DNS cache is cleared.

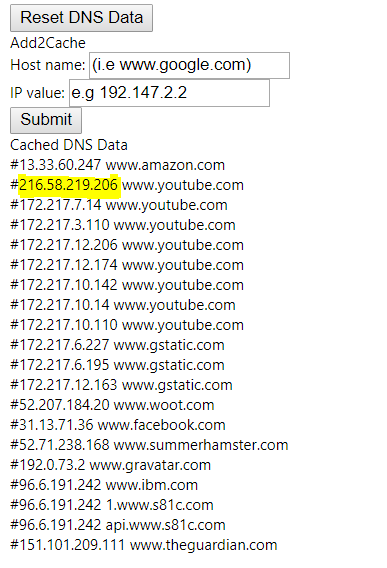
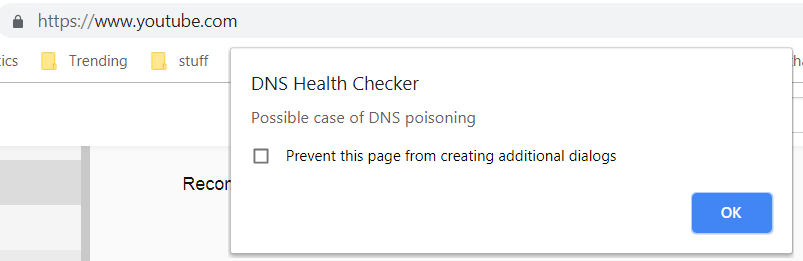


Figure showing possible case of DNS poisoning for youtube.



**Observations:**

The plugin is found to be quite effective in identifying changes in DNS loopup responses. To start with, I have added an incorrect DNS entry for [www.flipkart.com](http://www.flipkart.com) into the browser cache with the help of plugins user interface. The idea behind this test is that **DNSHealthChecker** would detect changes in the DNS loopup as compared to the previous entries if any. While this plugin works perfectly fine for hostnames with single DNS resolution, the same cannot be maintained for servers with global presence. In this case, a DNS name may have multiple A records, clients may then choose any record and use that address. For example Big players like Google, Amazon return different IP values for the same DNS query. This is because of CDN’s and (Round Robin DNS) load balancing, load distribution techniques they use. They also maintain a backup DNS entry as a fail-over. Additionally, the DNS responses are also found to vary with the location of the request. These cases are identified as possibilities of DNS poisoning and resulted in increased false positive rate.

To reduce the number of false positives, I have implemented a different metric for DNS cache matching in the plugin. Instead of relying on non-forgiving string matching algorithm we based our matching on the metric longest common prefix. The lcp of the DNS values list is calculated and if it doesn’t meet the threshold, an alert would be displayed accordingly. Even this approach won’t hold true for DNS responses that fall below the threshold. For this Design project, this would be the future work for this plugin.

**Shortcomings**

Since this plugin depends on the differences that present between the previous DNS lookup and the present one. It cannot detect abnormal DNS entries that have been changed previously in the server. This add-on gathers information about all the entries that are being resolved when we enter a website name in the search bar. So, if someone has access to the local storage of the browser extension, they can manipulate the entries to suit their advantage. For example, with such a case like Mirai, the attackers can spoof popular sites and redirect the user landing into an exact looking replica. And this plugin won’t be able to detect it accurately.

Additionally, it also contains the list of all websites that the user visits. This might pose a slight privacy risk as other users might peek into the entries that have been made so far. Nonetheless, the scope of this project is to check if DNS lookup values changed or not.

Submitted to Prof. Amir Rahmati by Sampath Kumar Kilaparthi (Graduate Student at SUNY Stony Brook University Department of Computer Science with ID : 112079198) for CSE508 course project.