DNS Spoofing as a Medium for Anonymized Package Delivery

Rick Housley, Francesco Garruzzo, Michael McCarthy
Department of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, New Jersey 07030
Email: {rhousley, fgarruzz, mmccart1}@stevens.edu

Abstract—**The Internet consists of two principal namespaces: the domain namespace, and the Internet Protocol (IP) address space (http://tools.ietf.org/html/rfc1034). Traditionally Domain Name System (DNS) webservers are utilized as a means to traverse between these namespaces by translating domain name requests into IP addresses. As the Internet largely consists of end-users and services in need of domain name translation, DNS traffic is highly prevalent. Google's Public DNS server alone manages 70 billion requests a day (http://googleblog.blogspot.com/2012/02/google-public-dns-70-billion-requests.html).**

We present a novel method for anonymized, elusive, unidirectional messaging and package delivery using existing DNS infrastructure. Due to the pervasiveness of DNS traffic we believe that malicious content encoded within DNS packets will go unnoticed by traditional Intrusion Detection Systems (IDS) and networking analysts. The proof-of-concept presented utilizes DNS spoofing to direct DNS responses to a listening machine. The listening machine decodes messages from the DNS packet's 16-bit transaction ID.

A virtualized network was used as the test-bench for our proof-of-concept implementation. Wireshark was used for debugging purposes as well as for ensuring the chaffy quality of the DNS requests. The final implementation consisted of a server and client. The server is capable of sending messages and files to the client while maintaining what would appear to be traditional network traffic. To ensure that the packets did not appear malicious Snort in conjunction with Snorby, a popular open-source IDS and its partner interface, were utilized. The results met our initial objectives with one shortcoming; the server-client pair only works on internal networks or with external-facing IP pairs. This is a result of Network Address Translation (NAT) acting as a firewall for non-requested packets.

I. Introduction

A. Motivation

Anonomized messaging has a number of use cases. In recent years social media has played a large role in geopolitical developments in the middle east. As a result governments have tightened their control over Internet infrastructure to quell widespread disobedience. Messaging clients that are discrete and anonymous provide security to dissenters and allow for the free dissemination of information.

Ignoring the social consequences of anonomized messaging services, malware has utilized non-traditional message exchanging techniques for quite some time to prevent detection. For example, simple malware has been known to communicate with Command and Control (C&C) servers via Internet Relay Chat (IRC) (http://searchsecurity.techtarget.com/feature/Command-and-control-servers-The-puppet-masters-that-govern-malware). A recent news article even showed that the popular website Reddit was used as a Command and Control Center by the well known iWorm botnet (http://www.intego.com/mac-security-blog/iworm-botnet-uses-reddit-as-command-and-control-center/). As HTTP, HTTPS, FTP, and SSH are highly prevalent protocols, malware is also known to tunnel traffic through such protocols.

B. DNS Background

A Domain Name System (DNS) is a layered hierarchical system used for naming any resource connected to the internet. DNS associates information with domain names that are assigned to these resources. Most notably, DNS translates a domain name into its IP address which is needed to actually navigate to the resource via the internet. The other key piece of information that DNS associates to a domain is its Time to Live (TTL). The TTL of a domain is a number indicating how many seconds to cache the resulting information of a domain on the DNS server.

When a request is sent to a DNS server, the server first checks its cache to see if the record already exists and has already been resolved. If the record exists in the cache, it responds to the request with the information from the cache. If the record does not exist, it reaches out to a DNS server above it, which repeats the same process. This chain goes until either a DNS server has the record cached and serves it to the requesting system, or until the request reaches a root name server. A root name server is at the top level of the DNS chain. When no DNS server in the chain of requests has the result cached, the root server is asked for the information, and the root server responds with the authoritative name servers for the domain.

C. DNS Spoofing

DNS spoofing is a method for diverting DNS responses from a requesting entity to a target IP address. Traditionally computers make requests from a DNS server by sending it a DNS request. The source IP, transaction ID, request name, and several other options are packaged into an IP datagram and are sent off to the destination DNS server. When a DNS message is spoofed a target IP is set as the source IP address.

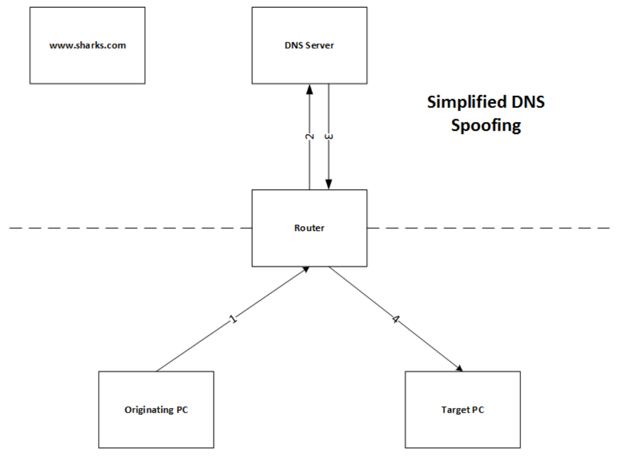

Fig. 1. The figure above depicts a simplified diagram of how the path of a DNS spoofing attack would operate.

One analogy of DNS spoofing is using an improper return address without a stamp. When a letter is sent without a stamp it is returned to the return address. If an improper return address is placed on the piece of mail it will be returned to a location different than the actual originating IP address.

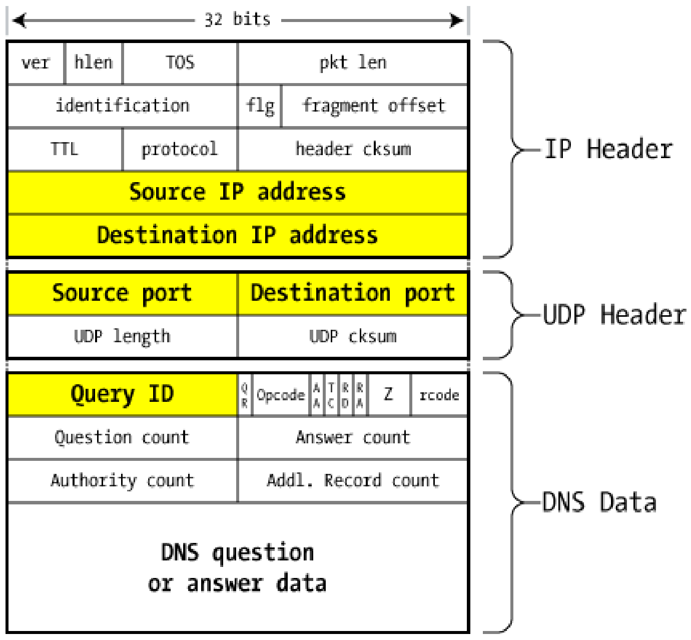

Fig. 2. The figure above depicts the format of an IP datagram. When spoofing a DNS message the source address is altered to an address different than that of its origin

II. Development Process

The development process largely consisted of two stages: verification of DNS spoofing as a valid means of discrete information transfer and final prototyping. Initial verification consisted of using the python 'scapy' package for datagram manipulation. Using the scapy package it is very easy to quickly form and send a packet (see below).

```
from scapy.all import *

mypacket = IP(dst="8.8.8.8")/
           UDP(dport=53)/
           DNS(qd=DNSQR(/
           qname="sharks.com"))
sr1(mypacket)
```

Fig. 3. Example DNS request packet creation. Requests the IP for 'www.sharks.com' from Google's DNS server (8.8.8.8)

After sending test packets, such as the one in Listing 1, the responses were observed using wireshark. Once it was verified that the packets were succesfuly formed and sent, we began to verify that it was possible to spoof DNS messages using similar methods.

```
from scapy.all import *

mypacket = IP(dst="8.8.8.8",
            src="155.246.5.38")/
           UDP(dport=53)/
           DNS(id=0x1337,/
                qd=DNSQR(qname="sharks.com"))
sr1(mypacket)
```

Fig. 4. Example DNS spoofing request packet creation. Requests the IP for 'www.sharks.com' from Google's DNS server (8.8.8.8). In this scenario the source IP has been modified tp an address differ

Once again wireshark was setup to confirm that the spoofed messages went through. After the transaction ID was confirmed to be a viable medium for encoding data, development of the final prototype began. The final prototype was developed as two individual applications: the server and the client.

III. Experimental Setup

Experimentation was performed off campus to prevent any issues with IT, especially for when malicious data was sent. As this project relies on the communication between a server and client a virtualized system was necessary. The host system ran the final server prototype and the virtualized system ran the client. Communication between the server and client were monitored through wireshark to assess their performance. Snort and snorby were setup on the virtual system to monitor and asses the maliciousness of visible network traffic.

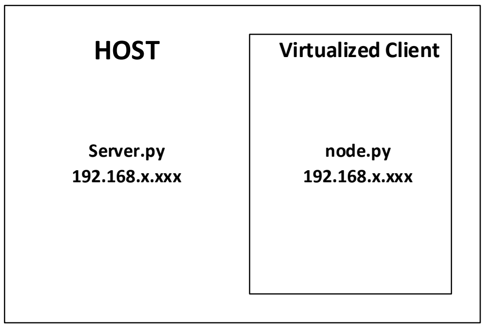

Fig. 5. The figure above depicts a diagram of how the test system was setup.

IV. Proof Of Concept

```
usage: server.py
[-h] [--rchaff c] [--fsend f] [--dmessage m] [--targetIP t]
                 [--randomDelay d]

optional arguments:
  -h, --help       show this help message and exit
  --rchaff c       File name of DNS request url chaff
  --fsend f        File to send
  --dmessage m     Message to send
  --targetIP t     Target IP in format
  --randomDelay d  Delay between DNS requests
```

A. Features

- **DNS Chaff**
  In order for all of the DNS requests/responses to seem like ”normal” internet traffic it is key for the traffic to mimic normal traffic. Therefore, each message request should **not** have the same DNS question. If a DNS Chaff file is provided the program with random cycle through different DNS questions. In testing the top 50 most popular websites were used.

- **Random Delay**
  Consistently timed DNS requests may look suspicious to a network analyst. For this reason DNS requests can be made at randomly specified delays.

- **Send File**
  An entire file can be sent using the methedology described within this report.

- **Message**
  A small single string message can be sent using the methedology within this report.

B. Limitations

Unfortunately we found that this process only works for external facing IP addresses or internal networks. This limits the ultimate functionality of the project. However, it is still viable for things like controlling malware on routers, etc.

V. Installation / Execution

Code must be cloned from the following git repo:
git clone https://github.com/rickhousley/cpe490-dns.git

Scapy must also be installed for packet manipulation. This can be easily accomplished using the following commands on a linux based system:

```
wget scapy.net
unzip scapy-latest.zip
cd scapy-2.*
sudo python setup.py install
```

Note: Given the rules of your local network administrator it may not be advisable to utilize this program without permission.

VI. Summary / Conclusions

DNS spoofing can be used to provide security in transferring messages and data anonymously. With that in mind, the team set out to successfully transfer data with no trace back to the source in a way that was most efficient and presented the best funtionality. After determining the plausibility of package delivery, the group decided that using python instead of C++ allowed for the group to create more advanced functionality with less complex code. In conclusion, the group determined that the method was in fact proven to work, untraceable, but the accuracy of the transferred data could be compromised. It was discovered that the server-client pair only works on an internal network or with external-facing IP pairs. We were able to successfully transfer data with minimal data loss internally. For future endeavors, we believe monitoring the entropy of these encoded data packets and comparing them to the entropy of typical DNS packets could be a viable way of preventing this attack.

VII. Acknowledgements

Contributors:

- **Rick Housley (Project Leader)** - Developed original concept for DNS spoofing as a medium for anonymized package delivery. Developed and debugged both main server source code as well as client code with oversight from partners. Is responsible for developing the encoding scheme used for each message.

- **Francesco Garruzzo** - Designed experimental setup and acted as our in house network analyst. Watched network traffic to verify spoofing as well as to ensure that generated traffic appeared indistinguishable from typical DNS response/request traffic. Also connected with our local network administrator to gain permission for out local tests.

- **Michael McCarthy** - Performed initial verification of DNS spoofing's plausability as a medium for anonymized package delivery. Headed the initial attempt to use C++ as the base server/client language. This was eventually

abandoned so more features could be implemented using Python.

We would like to thank Professor Hong Man for his thorough class CPE-490:Information Systems and Technology. Without his class this project would not have been possible.

VIII. References

- Na. ”Root DNSSEC.” Root DNSSEC RSS. DNSSEC, 1 Jan. 1998. Web. 11 Dec. 2014.

- N/a. ”DNSCurve: Usable Security for DNS.” DNS Forgery. DNS Curve, 1 Jan. 1998. Web. 11 Dec. 2014.

- N/a. ”Google Public DNS: 70 Billion Requests a Day and Counting.” Official Google Blog. Google, 2 Feb. 2012. Web. 11 Dec. 2014.