Published as : Hosnieh Rafiee, Christoph Mueller, Lukas Niemeier, Jannik Streek, Christoph Sterz and Christoph Meinel. "A Flexible Framework For Detecting IPv6 Vulnerabilities". The 6th International Conference on Security of Information and Networks (SIN2013), ACM, November 26 - 28, 2013 Aksaray, Turkey

A Flexible Framework For Detecting IPv6 Vulnerabilities

Hosnieh Rafiee
Rafiee{at}hpi.uni-potsdam.de

Christoph Mueller
Christoph.Mueller{at}
student.hpi.uni-potsdam.de

Lukas Niemeier
Lukas.Niemeier{at}
student.hpi.uni-potsdam.de

Jannik Streek
Jannik.Streek{at}
student.hpi.uni-potsdam.de

Christoph Sterz
Christoph.Sterz{at}
student.hpi.uni-potsdam.de

Christoph Meinel
Meinel{at}hpi.uni-potsdam.de

Hasso Plattner Institute, University of Potsdam
P.O. Box 900460, 14440 Potsdam, Germany

ABSTRACT

Security has recently become a very important concern for entities using IPv6 networks. This is especially true with the recent news reports where governments and companies have admitted to credible cyber attacks against them in which confidential information and the security of data have been compromised. In this paper we will introduce a flexible framework that can be used for penetration testing of IPv6 networks. Due to the large address space in each of the IPv6 subnets, the traditional scanning approaches do not work. Here we introduce our new scanning algorithm which will find the IPv6 nodes on the Internet which are using Domain Name System (DNS) servers. Our implementation results showed that the use of the DNS Security Extension (DNSSEC) with NSEC3 [5], which is a new and promising approach for the prevention of zone walking, was not able to prevent us from gathering information about nodes on different networks.

Categories and Subject Descriptors

H.4 [**IPv6 security**]: IPv6 Penetration Testing; D.2.8 [**DNS security**]: DNS vulnerabilities—*DNS Reconnaissance using DNSSEC*

Keywords

Zone walking, NSEC3, DNSSEC, Fuzzier approach, IPv6, Security, Privacy, Attacks, Penetration test

1. INTRODUCTION

Today, the use of the Internet for a myriad of different applications is increasing exponentially. This is because, for governments and businesses alike, it is an easier, faster and more economical way to reach and conduct business with target audiences. As a result of this wide use of the Internet, vast amounts of secret, proprietary, and personal data pass through Internet networks and are stored in network repositories. A major concern is the fact that the Internet has turned into a war zone where governments, businesses, and individuals work very hard to find ways to infiltrate the Internet networks and the network repositories in order to obtain this secret, proprietary, and personal data, which is then used in ways to further their goals to the detriment of those affected. The Internet is being assimilated into the cultures of countries around the world.

In the non Internet world, to send a letter to a specific place, one needs an exact address. To accomplish this task using the Internet, the Internet uses a standard protocol called Internet Protocol version 4 (IPv4). Every device on the Internet is assigned an IP address (a numeric format for IPv4 and a hexadecimal format for Internet Protocol version 6 (IPv6)) that is used to aid in the sending and receiving of information over the Internet. Unfortunately the address space in IPv4 can only support 2^{32} unique IP addresses and these have already been assigned to nodes on the Internet, which means that the supply of IPv4 addresses is exhausted. To rectify this situation,in 1998, the Internet Engineering Task Force (IETF) proposed that another version of this protocol be created to support a larger address space. This new Internet Protocol is called IPv6 [3]. But due to the unclear security flaws that existed with this protocol, its deployment was postponed for several years. There are too many variables involved and too many questions remain unanswered. In this paper we address this problem by proposing a flexible framework to be used for penetrating and securing IPv6 networks. Our purpose is to offer this system as a basic consulting system that will give companies and governments the ability to test their networks against the different types of attacks possible. These attacks are then categorized, which then makes it faster and easier to target them later. We also discuss the new vulnerabilities of DNSSEC when NSEC3 is used. This allows us to demonstrate that the use of DNSSEC will not prevent scanning attacks, but will decrease the chance of this type of attack succeeding. The result of this boils down to the fact that we use DNS as a tool to scan nodes that have DNS records in IPv6 networks. The remainder of this paper is organized as follows: Section 2 explains the definition of DNS and DNSSEC. Section 3 briefly summarizes possible attacking mechanisms. Section 4 explains the deficiencies that exist with the current test beds and implementations. Section 5 discusses our proposed framework. Section 6 evaluates our proposed frame-

SIN'13, November 26-28, 2013, Aksaray, Turkey. Copyright 2013 ACM 978-1-4503-2498-4/00/10 ...$15.00.

Published as : Hosnieh Rafiee, Christoph Mueller, Lukas Niemeier, Jannik Streek, Christoph Sterz and Christoph Meinel. "A Flexible Framework For Detecting IPv6 Vulnerabilities". The 6th International Conference on Security of Information and Networks (SIN2013), ACM, November 26 - 28, 2013 Aksaray, Turkey

work. Section 7 explains our plans for future work. Section 8 presents a summary of our conclusions.

2. WHAT IS DNS?

The DNS [7] is a hierarchical database that allows for easy translation of names into IP addresses. The data is stored on DNS servers in a specific format in what are called Resource Records (RR). Today, with the increased number of websites on the Internet, it is not an easy task to remember the IP addresses of all these websites. This is especially true of websites using IPv6 addresses. A good example would be in trying to remember and use the IPv4 address of the google website, "173.194.113.178", instead of remembering and using the user friendly address, www.google.com". For IPv6 it becomes even more difficult to remember the actual address for the google website, "2a00:1450:4005:809::101f", because of the hexadecimal complexity of the address. It is clear that one cannot remember thousands of websites by their IP addresses. This is why the DNS was proposed and is used to assist in the expansion of the use of the Internet by making it more user friendly. However useful this protocol is, its basic protection mechanisms do make it vulnerable to several types of attacks, such as cache poisoning, reflector attacks, spoofing attacks [9], etc.

2.1 What is DNSSEC?

The DNS Security Extension (DNSSEC) [1] was introduced to protect the DNS from security vulnerabilities. It provides the nodes on DNS servers with data integrity and secure authentication. DNSSEC protects DNS servers by using public/private key pairs to encrypt the RRs in a zone. A zone is a portion of domain space that is authorized and administered by a primary name server and one or more secondary name servers. For each zone a different key pair is used. For security reasons, DNSSEC uses a pre-signing phase for the zones. The private keys are removed from the DNS servers after signing the zone to avoid key exposure. This fact makes it impossible to sign any domains on the fly. It should thus be possible to verify that a DNS server is not only signing all the entries correctly, but also that the correct DNS severs are responding. To implement this, the server of the parent zone will be asked for its public key. This public key will then be used to verify the so called DS Key (Delegation Signer) of the initially requested server. Using this method it is possible to continue up the chain until the root server is reached, which can then verify that the initial server can be trusted. In order to accomplish this, the DNSSEC introduces the following new types of RRs:

- RRSIG: Signature for A, AAAA, MX, NS RRs
- NSEC/3 [5]: Used for NXDOMAIN responses. A NSEC RR includes the names of the successor and predecessor names in order to prevent replay attacks.
- DNSKEY: Public key of the zone. Clients can use this key for signature validation.
- DS (Delegation Signer): Used to create a chain of trust. This is done to ensure that the zone that a node is talking to is in fact the correct zone. The server of the parent zone is queried and then its own DNSKEY is returned in order to check the signature of the DS key.

3. EXISTING TOOLS USED FOR PENETRATION TESTING

There are currently several research groups and individual researchers actively involved in testing IPv6 networks for vulnerabilities in a variety of IPv6 protocol suites. These include addressing mechanisms, extension headers, fragmentations, tunneling or the dual stack networks (using IPv4 and IPv6 at the same time). One popular tool for this endeavor is The Hacker Choice (THC)[1] Attack Suite, which was created by and is supported by some individual researchers.
In spite of being considered one of the most complete tools available on the Internet with which to initiate a large variety of attacks in IPv6 networks, one needs to call each attack separately, one by one, using the required parameters in order to execute these attacks. Another problem with this tool is that no reports are prepared indicating whether or not the attacks were effective on the network and what hosts were vulnerable to what attacks. A third problem is that there is no good documentation for developers to use in writing additional code for their testing purposes. There are other available tools, but they have single functionality and thus they can be used only for a specific purpose. For example, nmap [6], halfscan6, etc. are used for scanning, or for locating, live nodes on IPv6 networks. SendIP [2], scapy [3], isic6 [4], etc. are used to generate and manipulate IPv6 packets. Some other tools are used just for testing a particular service. Web applications are one example of important target services within IPv6 networks. [8] explains the modifications required in order to be considered a valid current tool used for penetration testing of web applications based on checklists containing the most prevalent security issues. One example of a modification is related to a node scanning mechanism that should take into account the IPv6 large address size and modify its scanning mechanism accordingly. Another problem is that none of these tools can be used to scan a network when DNSSEC and NSEC3 are being used.

4. OUR PROPOSED FRAMEWORK

The initial phase used in attacking any network is to scan the nodes in order to gather their IP addresses in order to find what available services are running on them. The main assumption made about the existing scanning tools that are discussed in section 4 is that the nodes of the attacker reside within the same network or are able to make use of ICMPv6 messages in order to find all IPv6 nodes residing in the network(s). This assumption is based on the fact that the large IPv6 address space makes it unfeasible for an attacker to scan 2^{64} nodes residing in each subnet. However, this assumption is not always true. This is the reason why there is a need to design new scanning algorithms. Some attackers tried to misuse DNS, as a tool, in an attempt to gain more information about the nodes. This helps them obtain the IP addresses of nodes inside the target networks who have associated DNS records. However, this attack will fail in cases where DNSSEC is used.
Another problem with the current tools is their inability to generate reports and the fact that they do not focus on all

[1]http://www.thc.org/thc-ipv6/
[2]http://snad.ncsl.nist.gov/ipv6/sendip.html
[3]http://www.secdev.org/projects/scapy/
[4]http://isic.sourceforge.net/

Published as : Hosnieh Rafiee, Christoph Mueller, Lukas Niemeier, Jannik Streek, Christoph Sterz and Christoph Meinel. "A Flexible Framework For Detecting IPv6 Vulnerabilities". The 6th International Conference on Security of Information and Networks (SIN2013), ACM, November 26 - 28, 2013 Aksaray, Turkey

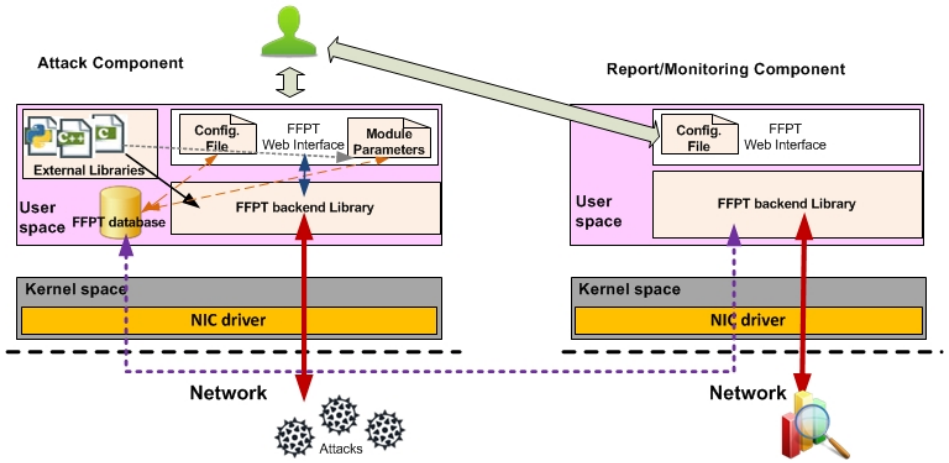

Figure 1: Components of Our Proposed Framework (FFPT)

available protocols used in IPv6 networks. They are also incapable of checking to see whether or not a combination of two or more protocols can be used to generate new attacks. One example relates to the vulnerability of the DNS server. To address these problems we propose the use of a flexible framework that we call a Flexible Framework for Penetration Testing (FFPT). We also discuss our new scanning algorithm that can be used in combination with existing scanning approaches in order to obtain the IP addresses of nodes residing on the target network. Our framework will also generate vulnerability reports which administrators can use to help in the resolution of detected security flaws in their network. This framework provides for the use of extensions from three different perspectives: service discovery, attacks, and reports. Here we will focus on the Attack components and we will discuss our scanning algorithm.

4.1 FFPT Architecture

FFPT is a user space consisting of an easy to use framework. It is made up of two main components: attack and report/monitoring. They can be installed for use in two different linux-based nodes in IPv6 networks. Each of these components makes use of a web interface and a FFPT backend library. Figure 1 depicts the architecture of these components. The flexibility of the framework gives users the ability of dragging and dropping their own scripts into the framework or to extending the framework through the use of other external scripts or codes that are available from the Internet by use of the web interface. They can add and save their commands, which contain the required parameters, using this interface so that, later, the FFPT backend library can compile the external scripts and codes and save them in a directory, and then, save the path to the compiled code in its database. FFPT will consider and run these scripts and codes during the next user-triggered network service execution. During the execution phase, the commands will pass through to the FFPT backend library in order to trigger a standard console command.
The web interface is written in PHP. Different credentials are considered by this framework in order to increase the transparency to the end user and to make logging activities easier in cases where a problem occurs. The credentials are:

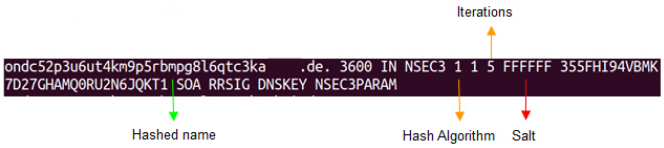

Figure 2: DNSSEC and NSEC3

administrator, developer, and general user. A developer is only allowed to add and modify his own scripts and code, which he added to the framework. Administrators are allowed to modify all scripts added by any developers to the framework. He can also designate people as developers or general users. A general user is only able to execute scripts that are currently available in the framework and he does not have the ability to add new ones or to modify existing ones.

Attack Component

This component performs two different functions:

Service/Protocol Discovery.
FFPT uses ping sweeps or ICMPv6 echo messages to scan the network in case multicast ICMPv6 messages were used. In cases where this approach cannot be used (Blocked ICMPv6 echo messages by a firewall or IDS), it should use our new algorithm. It then finds all available nodes and compiles a list of their running or supported protocols and services. This occurs immediately after FFPT user login. At the completion of this task, a list of all available services is displayed and under each service the IPv4/IPv6 address of all nodes running these services will be listed. By clicking on any part of this list, available attacks are shown for that particular service or protocol. The user can then choose the attack(s) that he wants to run, from the list of available attacks, and then he will initiate the attack(s).

DNS Reconnaissance Algorithm. When available, DNS Zone Transfers offer the easiest opportunity for reconnais-

Published as : Hosnieh Rafiee, Christoph Mueller, Lukas Niemeier, Jannik Streek, Christoph Sterz and Christoph Meinel. "A Flexible Framework For Detecting IPv6 Vulnerabilities". The 6th International Conference on Security of Information and Networks (SIN2013), ACM, November 26 - 28, 2013 Aksaray, Turkey

sance. There is usually a minimum of two DNS servers storing a copy of a zone file: a master and a slave. A single master DNS server keeps the main copy of the zone file, which contains the essential zone data. Slave nodes receive copies of that file using the Zone Transfer Process. Synchronizing zone files via zone transfers can be executed either centrally, by notifying slaves when changes are stored in the master, or by allowing slaves to query at a configurable time interval. By sending a DNS query with the special type, AXFR, a client may receive the DNS server's zone file. This is a full zone transfer request. A recent alternative approach is the Incremental Zone Transfer (IXFR).
By not securing DNS servers, all available hosts on those servers and, in fact the entire zone, will be exposed. Once DNS Zone Transfers are sufficiently secured, and DNSSEC is enabled, a request for a non-existing record will be answered with the nearest existing enclosing records. For example, a request for the non-existent foo.bar.com record will be answered with the signed records for fabulous.bar.com and great.bar.com. If DNSSEC does not sign a response for all non-existing records that are requested dynamically, then a potential DoS attack vector could be performed. A generic response claiming that the entry does not exist can be used in a replay DoS attack. This is why, even the message that proves the non-existence of a host, has to be signed and verified. But signing new entries, while the DNS server is online, is not possible for security and performance reasons. All signing is done offline and security does not permit key availability during runtime. DNSSEC's solution is to send "real" responses of the existing entries that have already been signed. When a zone is signed using DNSSEC, a linked list is created that connects all RRs in a ring, starting with the zone name, itself, in alphabetical order. The enclosing NSEC records, that are returned when no RR was found, contain the host names of the previous and next available host. The FFPT scanning algorithm can now start asking for non-existing records which will result in the return of the two closest existing host names. It will modify the request to find the next set of names and will continue to "walk" through the linked list until it returns to the beginning. This is also known as zone walking.
To cope with these flaws a new NSEC3 [5] standard was developed. Host names contained in NSEC3 RRs are no longer written in plain text. Instead, a salted and iterated, e.g. SHA-1, hash function is used to obscure the names. The hash function is only one way hashing, making zone walking, as described earlier, much more difficult. However, it does not totally prevent zone walking. The salt used for hashing is publicly available as shown in figure 2 and the function used for hashing is well known. FFPT can thus calculate Rainbow Tables for a list of possible or likely host names. By comparing the hashes from the Rainbow Tables and the hashes in the NSEC3 RR records, the clear text value from which the hash was computed can be retrieved. Calculating the Rainbow Tables in a smart way (considering often used domain names, which is also called dictionary based attack) can make host discovery a very feasible proposition. Here then, briefly, are the steps of our algorithm:

1. If AXFR is possible, ask for the whole copy of the zone file. Otherwise go to step 2.
2. If DNSSEC is not enabled, report a failure and stop scanning. Otherwise go to step 3.
3. If the DNS server uses NSEC, then start zone walking. Otherwise go to step 4.
4. If the DNS server uses NSEC3, start zone walking, gather hashed, execute the dictionary attack offline and then go to step 5.
5. If any records are left, do brute force attacks and repeat this step until no unknown record is available.

Attacks Execution.
This phase triggers all available attacking scripts for selected services over a certain period of time. This time was chosen by the user when this attack script was added to the FFPT. It also activates the report/monitoring component so that it is ready to monitor the network.

Report/Monitoring Component

The task of the report server is to act as a basic consulting system that advises the user as to what security and privacy flaws exist in his network. When the report/monitoring component receives a wake up message from the attack component, it then retrieves the list of attacks from the database of the attack component, and then starts monitoring the network in order to find out whether or not the attack was effective. This is done by sending probe messages or silently sniffing the network. For example, to see whether or not a node set its IP address based on a fake router advertisement message, the monitoring service sniffs the network to obtain the unsolicited neighbor advertisement message.
To improve the reports, the user can easily add new scripts in order to determine the effectiveness of the attacks. For some types of attacks this phase may have problems in generating the report. This is because FFPT does not install any external components nor does it activate SNMP on any node in the target network. This does not mean that the ability to use this protocol does not exist. If the user wants to manually enable SNMP on any nodes in his network, then he can configure FFPT to use the SNMP protocol.

4.2 Mathematic Analysis of the FFPT Scanning Algorithm

We are considering naive brute forcing. All possible combinations (including duplicate symbols) are calculated from formula 1. where l is the length of word, i is the number of iterations, A represents the 38 symbols in the alphabet which includes 10 digits (0-9), 26 letters (a-z) (only lowercase), dash(-), and underscore (_). $|A|$ is the size of alphabet.

$$\begin{aligned} &Cartesian\,Product(A \times A \times A...) = \\ &\{(a_1, ..., a_n)\ |a_i \in\ A\ \rightarrow \forall i,\ 1 \leq i \leq n\} \end{aligned} \tag{1}$$

So the size of the resulting set is $R = |A|^l$. If the length of word, i.e., $l = 5 \implies R = 38^5 = 79235168 \sim 80\ Million.$

4.3 FFPT Vulnerability Model

To make it easier to find new vulnerabilities in a targeted IPv6 network, N, we use the following mathematical concepts to define our vulnerability model.

- U: The sets of all standard protocols
- V(x): The sets of all vulnerabilities for protocol x.

Published as : Hosnieh Rafiee, Christoph Mueller, Lukas Niemeier, Jannik Streek, Christoph Sterz and Christoph Meinel. "A Flexible Framework For Detecting IPv6 Vulnerabilities". The 6th International Conference on Security of Information and Networks (SIN2013), ACM, November 26 - 28, 2013 Aksaray, Turkey

- V(x|y): The sets of all vulnerabilities for protocol x made possible by protocol y
- Pr(x): The probability for the occurrence of x.
- P: The sets of all running protocols in network N where:

$$P = \{p_i | p_i \subset U, i > 0\} \forall p_i \in P, \exists p_j \in P \rightarrow V(p_i) = V(p_i) + V(p_i|p_j), i \neq j > 0 \quad (2)$$

Formula 2 explains how combining some protocols in the network might create new vulnerabilities for other protocols. One example of this is p_j for DNS. From this statement we can define C as follows:

- C: The sets of different combinations (combination of 2 protocols, combination of 3 protocols, ..., combination of n protocols) of running protocols in network N where:

$$C = \{(p_i, p_j, ..., p_n) | p_j, p_i, ..., p_n \in P \text{ and } i, j, ..., n > 0 \text{ and } i \neq j \neq ... \neq n\}, C \subset P \quad (3)$$

The total members of $C = \sum_{i=1}^{(n-1)} Pe \begin{pmatrix} n \\ i \end{pmatrix}$ *where* $Pe \begin{pmatrix} n \\ i \end{pmatrix}$ is the number of all possible permutations of n protocol from P set by considering $C \subset P$.

$$(2), (3) \rightarrow V(N) \approx \sum_{i=1}^{n} (V(p_i) \times Pr(V(p_i))) + \sum_{i=1}^{n} \sum_{j=1}^{n} (V(p_i|p_j) \times Pr(V(p_i|p_j))) + \cdots + \sum_{i=1}^{n} \sum_{j=1}^{n} \cdots \sum_{l}^{n} (V(p_i|p_j p_l) \times Pr(V(p_i|p_j \ldots p_l))), i \neq j \neq \cdots \neq l \quad (4)$$

Formula 4 depicts the total vulnerability in network N with n protocols running. It is a general formula that shows the vulnerability model. It is possible to derive different formulas with regard to the different conditions that might exist in the network.

$$(4) \rightarrow \lim_{Pr(V(p_i|p_j \ldots p_l)) \rightarrow 0} V(N) \approx \sum_{i=1}^{n} V(p_i) \quad (5)$$

If for each protocol we assume there is at least one vulnerability and the probability of a vulnerability resulting from a combination of protocols is heading toward zero, then the total number of vulnerabilities in the network can be estimated using formula 5. It is one of the possible cases where a derivation of formula 4 is used. For example, if there are 10 protocols running in the network, and each protocol has at least two vulnerabilities, then $V(N) = \sum_{i=1}^{10} 2 = 20$. Another example is when the combination of only one protocol (p_x) with other protocols is important, and there is no permutation, then formula 4 changes to:

$$(4) \rightarrow V(N) \approx \sum_{i=1}^{n} (V(p_i) \times Pr(V(p_i))) + \sum_{i=1}^{n} V(p_i|p_x) \times Pr(V(p_i|p_x)), p_i \neq p_x \quad (6)$$

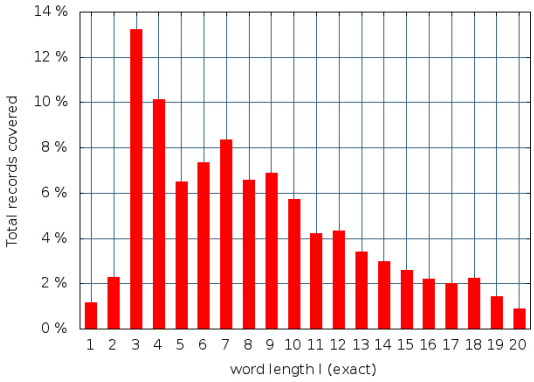

Figure 3: Distribution of Third Level Domain Length

5. EVALUATION OF OUR FFPT

We evaluated our framework by considering some factors such as the time it takes to find the nodes' IP information from the DNS server along with the time needed to generate the report describing said services and attacks.

5.1 Node Detection

Dictionary based attacks make use of a predefined set of commonly used names so as to compare the hashes of these names with the hashes from the server. However, a list of the most commonly used names is needed first. Additionally, intelligent brute-forcing needs some data itself. To gather these information and fill our database with useful domain names, we implemented our DNS scanning algorithm using python. We did not optimize the implementation, so it uses a single core thread, which means that there is a potential for optimization to enable it to run faster. We ran our code on a computer with a 2.60 GHz CPU. We also implemented the case where we have a database full of common names. To gather the names for this database we found the list of most popular web sites from Alexa.com, but it was unable to provide us with much information about the third level domains behind these domains. To get the information about third level domains we needed to make the request using AXFR. We scanned one million domains. Among them, 55160 domains (5.5 %) were not secure, which enabled us to obtain a copy of the zone file by using an AXFR query. This helped us to gather 1.43×10^6 RRs which we stored in our database for later use in a dictionary attack against NSEC3. According to our results, the popularity of a 3 length word (3 character word), third level domain (Example: www.google.com, www in this combination is the third level domain) is about 14% and the next popular third level domain is a 4 length word. Figure 3 shows the popularity of an n length word, third level domain.
Figure 4 shows how many third-level-domains can be col-

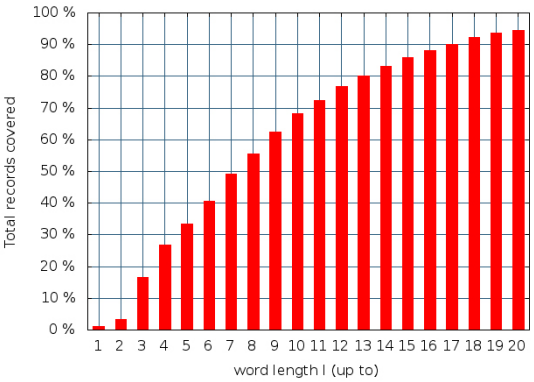

Figure 4: Name Length Distribution of Our Collected Data

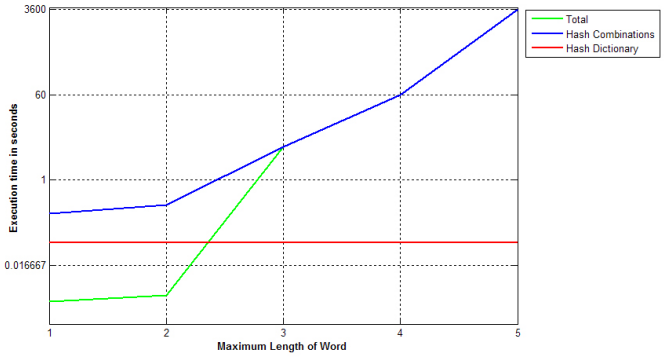

Figure 5: Time Required to Execute Brute Force Attacks On Words of different Length

lected, by percentage, with a given number of characters (length of word). We could cover 50% of all third level domains by the use of brute-forcing with up to 20 characters. Figure 5 shows the different combination of alphabets as was explained in section 4.2 and the time required to do brute force attacks against this combination. Our result reflects an hours time using a normal CPU. As explained earlier, we did brute force attacks offline, and did not ask the DNS server for each record after gathering hashes from the DNS server. This is because we did not want to do a DoS attack against a DNS server when our goal was to obtain more records about nodes. We also did not want to slow down the I/O.

5.2 Attack Execution

The time duration of attacks is a variable time determined by the user and input using the FFPT web interface. If the user chose a value of 40 seconds, then the duration of the attack would be 40 seconds.
To enhance the framework with the addition of more attacks, we implemented new attacks with which to evaluate the IPv6 protocols. These attacks entailed the use of fuzz mechanisms, evaluating DNS protocols such as multicast DNS (mDNS) [2] and the evaluation of Mobile IPv6. Some instances of the new attacks are as follows:

Attacks Against Multicast DNS (mDNS)

DNS is one of the application layer services that uses an IP address. This is why the existing attack tools on DNS needs to be modified in order to support IPv6. mDNS is one of the new operations of DNS used in a local link in the absence of a unicast DNS server. The domain names using multicast DNS end with *.local*. mDNS is used for finding printers or other shared folders of different Operating Systems (OS), or for checking the uniqueness of names used in local links. When a host joins a network that supports multicast DNS, it tries to set its local hostname, like *mydomain.local*, and then it sends a multicast DNS message to all nodes on that local link to see whether or not the chosen name is unique. We implemented a Man In The Middle (MITM) attack using C++ by extending the packet-manipulation-library [5].Using a *Sniffer* object, provided by the packet-manipulation-library, the component analyzed all traffic on UDP port 5353. Each host name query (AAAA, i.e. IPv6 RRs) was stored in a map laid out to remember the questioner and the host name in question. Every time a DNS response was received by our framework, the map was checked for a matching answer. If found, the attack is carried out against the original questioner. For example, if node A and B want to communicate together then node B will ask the name of node A using mDNS in order to connect via its name. Node A picks up a name and tries to check the uniqueness of this name by sending a mDNS message. Our MITM components (in FFPT framework) receive this message as well as Node B. Our component can then spoof that message and claim to have that name. There is no security proposed for mDNS, so this can easily happen. Our component could wait for node A and B to start their connection. First the MITM component will send out unicast goodbye packets to B indicating that the original host name holder gave up his authority over the name (node A will not hold onto A's name anymore). Then the MITM component sends another spoof unicast mDNS message to node B claiming to be allowed to use name A and continues the communication and then redirects the communication to the real A. In this case it plays a MITM attack. According to the mDNS RFC, the node should not accept unicast messages. However, we discovered that all current mDNS implementations accept unicast messages so that we were able to successfully execute our attacks.

Fuzz Attacks

Fuzz testing is one of the popular testing approaches used by industries. To better test an IPv6 protocol stack, we implemented fuzz attacks. The code is called by the FFPT framework, which is then responsible for the creation and sending of fuzzy IPv6 packets as well as the monitoring the target machines. Our first approach falls into the category of random input generation. We used scapy to generate our own packets as it allows for the generation of invalid packets, which is useful for the fuzz approach. Our experimental results show that many packets are being rejected in the early stages because of invalid formats. In some cases the victim node's crashed and we had to reboot the system. To improve our results we used grammar based fuzz mechanisms. Grammar based fuzz mechanisms are more precise. To fa-

[5]https://code.google.com/p/packet-manipulation-library/

1	www	50868
2	mail	40234
3	ftp	31948
4	localhost	21254
5	webmail	13830
6	smtp	13581
7	pop	11915
8	*	10974
9	webdisk	9264
10	ns1	8924
11	ns2	8693
12	cpanel	8105
13	whm	7955
14	autodiscover	5918
15	_autodiscover._tcp	4703
16	autoconfig	4568
17	ns	3126
18	test	3030
19	m	3016
20	default._domainkey	2846
21	blog	2437
22	imap	2424
23	dev	2159
24	pop3	2031
25	www2	1794

Figure 6: Some of The Third Level Domains Records Collected in Our Database

cilitate the process of grammar based fuzzing we used the Peach Fuzzer framework [6]. During the day that we ran our code, we could not find any malfunction in nodes that accepted our IPv6 plain packets. However, as we did not have access to the IPv6 implementation stack, it was not easy to completely evaluate the target nodes.

5.3 Report/Monitoring

The last step, and the most important step for FFPT, is the generation of reports. Based on our experiments, it takes an average of 9.77 seconds to generate a report on one protocol, like the Neighbor Discovery Protocol (NDP). This report varies from one protocol to another as it is dependent on many factors such as how fast the victim nodes responds to attacks, the duration of the attacks and whether the report component needs to send a probe message to gather data from the nodes in the network or whether just sniffing is sufficient.

6. FUTURE WORK

This framework is still in the deployment phase, which makes it a work in progress. We plan to enhance it with new attacks based on our assumptions from the vulnerability model. We are currently focused on some protocols in IPv6 networks, such as DNS, NDP, Dynamic Host Configuration Protocol (DHCP), Mobile IPv6, etc. We also plan on improving the report in order to generate more effective reports.

7. CONCLUSIONS

In this paper we proposed a flexible framework that gives users the ability to enhance it through the use of their own scripts or through the use of external code and scripts. We also introduced our scanning algorithm that will improve the scanning results in order to check for the possibility of all attacks against target networks. Unlike other tools, that do not produce a report of the vulnerabilities or that have a single functionality, our framework plays the role of a basic consulting system. The probability of new vulnerabilities being introduced in the network is also considered. The framework uses a learning approach to find the vulnerabilities that result from the combinations of different protocols in the network. Based on our evaluation, we could find a model for vulnerabilities and produce a time estimate for scanning a target network using our scanning algorithm.

[6]http://peachfuzzer.com/

8. REFERENCES

[1] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Protocol modifications for the dns security extensions. IETF, Mar. 2005. http://tools.ietf.org/html/rfc4035.

[2] S. Cheshire and M. Krochmal. Multicast dns. IETF, Feb. 2013. http://tools.ietf.org/html/rfc6762.

[3] S. Deering and R. Hinden. Internet protocol, version 6 (ipv6) specification. IETF, Dec. 1998. http://tools.ietf.org/html/rfc2460.

[4] S. Hogg and E. Vyncke. *IPv6 Security*. Cisco Press, ISBN-13: 978-1-58705-594-2, 2009.

[5] B. Laurie, G. Sisson, R. Arends, and D. Blacka. Dns security (dnssec) hashed authenticated denial of existence. IETF, Mar. 2008. http://tools.ietf.org/html/rfc5155.

[6] G. F. Lyon. Nmap network scanning: The official nmap project guide to network discovery and security scanning. ISBN-10 0-9799587-1-7, Jan. 2009.

[7] P. Mockapetris. Domain names - implementation and specification. IETF, Nov. 1987. http://tools.ietf.org/html/rfc1035.

[8] C. Ottow, F. van Vliet, P. de Boer, and A. Pras. The impact of ipv6 on penetration testing. *Springer*, 7479:88–99, Auguest 2012.

[9] H. Rafiee, M. v. Loewis, and C. Meinel. *Challenges and Solutions for DNS Security in IPv6*. IGI, http://www.igi-global.com/chapter/challenges-and-solutions-for-dns-security-in-ipv6/78870, 2013.