Chapter 3

Related Work

3.1 Relevant Literature

Henniger et al. [11] perform Internet-wide scans of the entire IPv4 address space for hosts that listen on port 22 (SSH) or port 443 (HTTPS) using the NMap tool. NMap is an open-source network scanner that is used for network audits. They perform TLS or SSH handshakes on hosts that listen on port 443 or port 22, respectively and gather most of the data that might help discover weak keys. For TLS, they capture most of the metadata like certificate information and other X.509 certificate fields. For SSH, they collect the host keys using a simple client developed in C. All of the scans and data processing were carried out using Amazon's EC2 service, which provided them with an infrastructure to collect and analyse data efficiently. After processing the information, they identify some patterns between a host that shares keys and try to pinpoint some of the causes behind these. In the TLS protocol, the server sends its public key in a TLS certificate during the handshake stage. This key is used to provide a signature during the handshake stage or can be used to encrypt session details depending on whether RSA or DSA encryption algorithm is negotiated between the client and the server. Similarly, in the SSH protocol, the host key allows the server to authenticate itself to the client by providing a signature during the handshake stage. In both these protocols, if the private keys are known by an attacker and depending on the type of the encryption scheme (RSA or DSA), an attacker can perform different attacks like man-in-the-middle or decrypt the message containing the session key and, in turn, use that to decrypt the entire session.

[11] further computes the private keys using the weak public keys by exploiting the weakness of RSA and DSA algorithms when used with insufficient randomness. As a result, they can compute private keys for 0.50% of the total TLS hosts and 1.06% of the SSH hosts found from their respective public keys since the key pair is related mathematically. They scanned 12.8 million TLS hosts

and only obtained 5.8 million unique certificates for them. For SSH, they scanned 10.2 million hosts and obtained 6.2 million unique keys. Indicating that about 61% and 65% of the TLS and SSH hosts shared the same key as another host in the scans. Investigating the causes of this key sharing between hosts, they found that some of the key sharing was not due to vulnerabilities in the infrastructure but large hosting providers that used a single key for multiple IP addresses. Another significant reason was TLS certificates that belonged to the same organisation. Therefore, they excluded exceptions from their data analysis process and clustered the remaining host that shared keys.

Dr Farrell's work on surveying long term cryptographic keys for SSH, mail and web protocols [8] is relevant literature for this project. The work surveys keys for ten countries and [8] describes scans that involved hosts in Ireland and Estonia. The program used to carry out these scans made use of Maxmind Geolocation databases and Censys databases to perform these scans. Then ZMap and ZGrab are used to collect data on hosts that listen on port 25, and further, the data is processed and analysed. From one of the scans in Ireland carried out in 2018 that involved 18,268 hosts that offer at least one SSH or TLS service, approximately 53% hosts shared keys. Out of the 54,447 hostport combinations, only 36% unique keys were seen through that scan. The research also reports key sharing among countries and shows how widespread this resue is. The clustering of hosts is based on the fingerprint information collected from SSH hosts and hosts that use TLS on top of the standard protocols. "If any two IPs share a fingerprint irrespective of the port, they are clustered together" to check for key reuse among hosts. Other metadata during TLS and SSH handshakes are collected for extensive data analysis. After the data analysis, some scripts are used to visualise the cluster information to communicate with local asset holders to understand better the causes of this key reuse from an asset holder's point of view and how the network security infrastructure can be improved. He introduces a metric called HARK %: "Host that is resuing keys" for quantifying this key reuse. One of the main hypotheses of the work was to see if there is a correlation between the HARK% and improvement in the security posture.

Since most of the work carried out in [8] was done using a combination of Python2 and Bash Scripts is quite memory intensive and Python2 has depriciated since. As outlined in section 1.2, the goal of this project is to migrate the existing code to Python3+ and refactor the code to increase readability and hopefully reduce the extensive memory usage using new functionality available in Python3. The proejct also checks how key reuse has evolved over the years since 2018.

3.2 Summary

This section summaries some of the causes and dangers of this key reuse as found in [8, 11].

3.2.1 Causes

- **Hardcoded Keys:** Many devices are shipped with default keys by manufacturers in the firmware of the device. Devices that belonged to the same model shared the same keys unless changed by the user.
- **Lack of Randomness:** Another significant reason for key reuse was the lack of entropy during key generation. They concluded this by checking hosts in clusters that shared keys and cross-checking that with other models sharing the same keys. The frequency distribution indicated a flaw during key generation.
- **Multihomed Hosts:** If a host has more than one network interface with different IPv4 addresses, it will show as individual hosts in the scans.
- **Virtual Machines:** If a few hosts that share a key and all those hosts are VMs, then key reuse it can be passed as acceptable as all hosts are all under one physical entity.

3.2.2 Dangers

- **Cross Protocol Attacks:** Even though new versions of TLS are available, the migration from the older version is still slow. Key reuse can increase the probability of cross-protocol attacks, and old versions of TLS/SSL may be vulnerable to these.
- **Masquerading:** If asymmetric encryption is used for authentication, then key reuse may enable an attacker to pose as a host if a breach of a host occurs in a cluster.
- **Credential Exposure:** If sensitive information like passwords are sent using plain text protocols like SMTP, and if an active attacker can masquerade as a host in a cluster, he/she can capture those credentials.
- **Cost of Leaks:** Since private keys are usually stored in some physical disk, there is always some probability of leaks. If there is a leak from a host in a cluster, it will affect all cluster members. Moreover, as the cluster size increases, the cost of these would also increase.