IoT security enhanced by removing the human factor

A secure mechanism for updating the defaults in an IoT (Internet of Things) device without human intervention makes the device entirely resilient to present-day dictionary-based attacks without a discernible impact on performance.

The Internet of Things domain has moved considerably from being limited to company infrastructure in technologies like the RFID in 1999 to everyday objects like refrigerators, ACs, television today, and we expect it to cross the 10 billion mark in 2020. With the introduction of an intelligent 'default value updating algorithm', we would like to eliminate the possibility of hackers remotely accessing home IoT devices. This algorithm will inform the device manufacturers of a new installation and facilitates in the automatic update of the default password and the default IP address of one's IoT device eliminating the dependence on the human user.

The manufacturer currently urges the users to change the default password on their IoT device upon installation and initial configuration. However, for most users, accessing the management console of the device is a tedious task. The Mirai Botnet, in 2016, infected around 2 million IoT devices with the help of just 68 default username and password pairs. Unlike traditional cybersecurity devices like the laptop computer, applying security schemes on an IoT device is exceptionally challenging due to the limited computational and storage capabilities. The device is also required to be extremely energy-efficient since most are cell-powered.

There have been many approaches to securing an IoT device, from beefing its encryption by incorporating modified low-power algorithms like Adiantum by google or using the moving target defence concept where the device footprint continually changes. However, these mechanisms are useful only when the attacker wants to brute-force. To circumvent the above techniques, if one uses them individually, one can find the internet-facing IoT devices using a search engine such as Shodan, use default credentials and easily log into the interface, update the firmware and turn it into one of the million bots under one's control, remotely. To tackle this attack mechanism, a manufacturer can incorporate our algorithm into their IoT device which, along with secure encryption like Adiantum and Moving target defence will sufficiently protect any IoT device from becoming a zombie.

We assume that the user of the device has an account on the manufacturer's website and has linked the device ID to the account. Now, the device contacts the one-time password reset server setup on the manufacturer's end using SSH, and upon verification, the server provides a randomly generated alphanumeric string 'r'. We assume the length of this string as 'l'. On receipt of 'r', the device computes an HMAC 'l – 1' times on 'r'. We have then programmed the algorithm to use our Key extraction algorithm, similar to HKDF, but using less power and computation, to extract the human-readable key from this hashed value which is the new password for the console. The server mails the password to the registered user's email. A similar approach is followed to change the default IP address for the management console.

We evaluate the effectiveness of our algorithm in safely changing the default password and IP of the console by setting up a python server that acts as the manufacturer's website and OTP reset server and three Raspberry pi as representations of IoT devices. We attach two wattmeters to measure power consumption. One pi will act as a present-day IoT device (our baseline) and will contact the python server using RTSP which is used by most CCTV cameras to stream video over the internet. The second pi has our algorithm flashed into memory and thus upon configuration connects to the python server requesting for 'r'. The python server provides the r-value, hashes it, extracts the key and mails the new password to the user's email registered to the account. Simultaneously, the

device changes the password on the management console without user intervention. We measure the power consumption and calculate the overheads caused by our algorithm. The server calculates the time required by the device to reset password to gauge the speed of the algorithm. The third pi will be used to test how resistant this algorithm is against an external attacker who tries to snoop into the network using Wireshark and SSLstrip. We expect to find that SSL does an excellent job at masking the 'r' value and since the attacker needs both r value and device ID (hardcoded in the firmware), he is unable to compute the HMAC to find the password of the console. He is also unaware of the algorithm used for the extraction of the key from the hash. Also, since the password is not a simple word, a known-dictionary attack cannot be employed on the device.

We conclude that future research needs to be conducted to compress the flash size of the algorithm (to take up less storage) and more efficiency in power consumption. Moving forward, we expect that this trend of not relying on users to completely secure their device and manufacturers taking full responsibility of securing the device will provide a much secure future for the Internet of Everything.