Aiakos: *Updating critical informations without prior secure communication channel*

Hugo Genesse
Génie informatique
Polytechnique Montréal
Montréal, Canada
hugo.genesse@polymtl.ca

Alexis-Maurer Fortin
Génie informatique
Polytechnique Montréal
Montréal, Canada
alexis-maurer.fortin@polymtl.ca

Abstract— **Weak, reused passwords on a large number of IoT devices is a current major threat to the security of the internet. We propose a secure methodology and tool to update critical informations such as passwords without a prior secure communication channel like SSH. Our methodology uses already present technology on those devices to be able to patch already deployed systems. This exercise explains the various challenges the IoT faces and why usual security practices fall short with this new paradigm.** ***Keywords—Internet of Things, Authentication, scalable security methodologies.***

I. Introduction

As the growth of the number of computerized systems continues, the same paradigms that were used for some systems might not apply to the scale of the Internet of Things. By introducing a large quantity of those limited embedded devices into the internet, models like user-controlled passwords were challenged. For example, for administrative work on routers by internet service providers, having only one common password for every device is convenient. The support technician can easily manage the access point to his liking without the hassle of having the customer present to provide the custom password. This poses a security concern.

The malware research group MalwareMustDie identified default passwords as the attack vector used by the infamous Mirai botnet that attacked DNS servers resulting in internet service disruption in the US[1]. This attack proved the power of controlling IoT devices vulnerable because administration interfaces were protected only by default passwords. It used a list of only 61 default passwords used by manufacturers[3]. Furthermore, this list could easily be increased with passwords used by various ISPs and other manufacturers to have a higher number of controlled devices. Those password lists seem to be attractive for criminals as a list of more than 1700 devices IP and their password was published in August 2017[4]. It is then possible to understand the impact that deployed IoT devices that have a weak, reused password could have on the internet security in the near future.

Gartner forecasts 8.4 billion IoT devices to be operational in 2017 with a large portion of them, 5.3 billion, being consumer operated devices[5]. Predictions indicate that this number can only grow exponentially. As OWASP[14] states that insecure administration interfaces that have default passwords are the number 1 threat this type of devices faces, we can imagine a large number of devices presently deployed that are vulnerable to such attacks.

This is why default passwords on IoT devices is a major problem and should be addressed while attempting to explain the different technical specificities of the problem of changing critical informations on IoT devices without any prior secure administration interface.

Our research objective is to establish secure password changing methodologies for deployed IoT devices and establish and quantify the challenges this new paradigm faces. We think that those go beyond the usual suspects of lack of awareness by developers or simple laziness. Our hypotheses are that the scale, lack of prior secure administration interfaces, resources and time play a definitive role in the lack of security of the password management in the IoT world. Those are metrics that are important in the adoption of methodologies and are not well understood by the security community when put in the connected embedded devices context in our opinion. Our current use case for IoT devices will be consumer grade routers.

We then want to measure the impact of not reviewing our security practices when we face the current IoT security problems. The contribution brought by this work aims to help nuance the opinion about the methodology needed when tackling security issues with smart connected devices. We will present the state of the art on technologies that touch password management in the Internet of Things and past research regarding threats present in this field as well as possible solutions. The methodology of the experiment will then be presented on our solution and its testing. Afterwards, we will discuss the results of our experiment and then dress our conclusions from the results and our analysis.

II. State of the art

Prior work on this subject approaches it from different angles. The majority of work, evaluates the efficiency of default credentials used on devices. They all come to the same conclusion that default passwords are a security flaw for this type of device[7]. Those default credentials are vulnerable to trivial brute forcing because they consist of short passwords with low entropy, making it easily guessable by modern brute force tools. It was also found that very often manufacturers use the same default passwords for many of their devices. This makes it easy for attackers since as soon that they have the information about the manufacturer of a certain device they can easily find the default password and use it, therefore skipping the need for performing a brute force attack. Previous research agrees that default passwords must be changed for tougher passwords that cannot be brute forced or easily found on the internet[7]. Although we agree with those statements, those articles do not provide a solution on the problem of managing the currently deployed devices.

One aspect that is not approached especially when dealing with multiple devices is password management. Indeed the suggestion to change those default passwords is an interesting one but they have to be managed securely afterwards and that is also a challenge. Fortunately, good password management software exist but there is still a technical challenge to manage such a large number of heterogeneous devices that has to be solved. This problem is relevant to ISPs because of the large number of passwords they have to manage for their customers and the need to be able to provide support services to them.

Previous research fails to come with a better solution other than saying passwords must be hardened. Indeed, this can be easily done by security conscious home users, but those are rarely preoccupied by the security of their devices[4]. It then falls to the ISPs or the manufacturers to secure the devices they distribute. The number of devices ISPs distribute to their clients brings in a scale problem for password changes which does affect the home user. Also, the fact that some of those devices don't have secure communication channels and users prefer unencrypted but simpler ones [8] which makes it harder for internet service providers and vendors to use the software present on theses devices to remotely change those passwords because of a lack of encryption to transmit them. We can also conclude that it may not only be an awareness issue but also a usability problem that leads to those practices. The computer security company Rapid7 created a tool written in Perl to verify IoT devices on a network. It scans the network to find connected devices and then according to its characteristics, it tests if it is using its default password and reports it[9]. This tool is useful but users still have to change those passwords afterwards.

Some work also focused on authentication between the devices. For example, Pereira and al. proposed a challenge-response authentication mechanism that maintains interoperability and low power consumption[10]. Power consumption and overhead are definitely a challenge that we rarely face with non-embedded devices and solutions to problems regarding IoT have to take it into consideration. This research study doesn't take humans into account because the goal was to verify devices, and not users, which doesn't apply in the current use case. The authentication mechanism should be usable by humans as well for their administration needs without clashing with their usual practices.

We also see papers[6] confirming that default passwords are in use for insecure communication channels. Those devices are already deployed and pose a real threat. Shodan states that 22 255 devices run with default passwords on a telnet service with most of them identified because the service banner explicitly states the default credentials[12]. The paper from Patton and al. doesn't provide any scalable solution to the problem but evaluates it as of critical severity.

III. Methodology

1. *Solution*

The solution we present solves the problem of changing default passwords securely and at scale and limits the technologies needed. Our goal is also to explain the unique technical difficulties encountered while creating a tool that can be used by administrators of a large number of devices to manage their passwords easily and securely while limiting the prerequisites for the device. To do so, we need to create our own solution. The program connects to the devices specified by the user. After connecting, on the device it calls back using HTTPS to get a new password. The password is created using a random password generator and sent to the device. All the passwords for devices are stored in a csv that can be imported in any well known password manager. We think that this choice improves security as we don't have to code our own encrypted database for passwords that could be vulnerable because of a lack of review of the cryptographic implementation. An administrator using this tool can easily change the passwords of multiple devices at once and using his password manager of choice reconnect to those devices when needed.

Our program is going to be written in Python3 and make use of the language standard modules which will make our code portable, easier to maintain and easier to expand as Python is a popular language with a large community.

The first step is to connect to the devices. Telnet is one of the main protocol used by IoT devices for remote connection and management and is not encrypted. This provides our prior insecure communication channel. Using the provided default credentials, the program will connect to the device using the Python Telnet client. Using an insecure communication channel doesn't diminish security at this step because we use the already known default password that will be changed in the next seconds.

The second step, when connected through Telnet, is to establish a secure communication channel that will be used to request the password and change it. To do so, we install a SSH server on the device. The device is then accessible with the same weak credentials but now through a secure communication channel. We can then close the Telnet connection as this was the only action we needed that connection for and the password changing process can continue through a more secure channel.

The third step is to connect to the device via the secure channel that is the previously created SSH server. We can then disable Telnet as it shouldn't be used anymore because the new password would be sent in plaintext on the next Telnet session and could be intercepted at that time. We then request the new password through an HTTPS request to the Aiakos server. This adds security as a by-product of changing the password securely by eliminating the weak administration interface to discourage the user from using vulnerable technologies.

On the fourth step, the device requests its new password to the Aiakos server. To make the request in our experiment, we used the wget GNU utility as a large number of devices are susceptible to have that program installed. We use the “-O-” switch to output the new password to the standard output to be able to parse with the client.

On the server side, the next step is then to generate a new password using the Python secrets module[12] to generate good random passwords that are hard to brute force. We preferred using a standard implementation of random choice instead of making our own to benefit off the code review that has been done for this module. Having the new password generated, it then sends it directly to the device and the device changes it. Using a classic HTTP server would allow the attacker to intercept and see the clear text value of the password because it was not encrypted before sending. Even if we used SSH to connect at this point, we want the password to be transmitted securely to the device from the Aiakos server. To solve this, we have an HTTPS server using the SimpleHTTPServer and SSL modules from Python standard modules, that permits secure connections with the device. We made that choice because those devices have a higher chance of being able to make HTTPS requests because of their need for an encrypted web interface or to communicate with secured APIs and web services to update their firmware and limit the risks of intercepting the update. This minimizes the risk of getting the new passwords stolen by listening to the traffic on the network.

The server also keeps the changed password, IP address and username in a standardized CSV file that can later be imported into a password manager. Users should not keep this file and is then not included in the Aiakos threat model.

The final step is done by the client after receiving the password. It changes the password with the chpasswd command and then closes the SSH connection. The procedure is then finished.

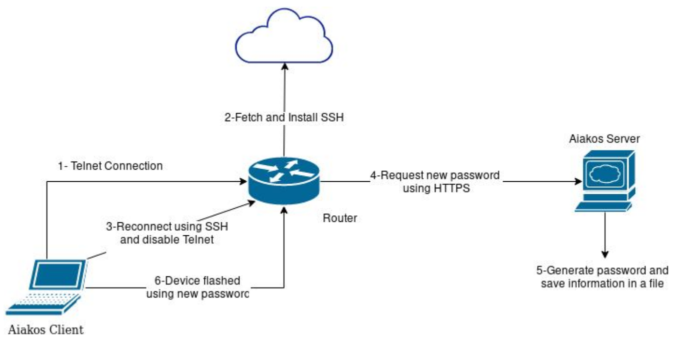
2-Fetch and Install SSH
Aiakos Server
1- Telnet Connection
4-Request new password using HTTPS
Router
3-Reconnect using SSH and disable Telnet
5-Generate password and save information in a file
6-Device flashed using new passworc
Aiakos Client

Fig. 1: Aiakos password change process.

2. *Testing*

To test our new tool to manage passwords on IoT devices without secure administration interface, we identified 2 characteristics that are critical to the success of this type of technology.

The first metric is the time to change the password on a fixed number of devices. This serves the purpose of highlighting the difference of scale between IoT and usual practices. We fix that number to 5 devices which makes for easier testing. The time will be calculated manually when passwords are changed by operators and with the time UNIX utility when passwords are changed by our tool. This provides more precise results when possible. The results will then be compared to extrapolate the time needed for a higher number of devices.

The second metric is security of the password change process. To measure the security level of the method used, a number of characteristics will be verified. The first will be the use of a secure communication channel to transmit the password or that the password is transmitted on the wire without encryption. The second will be the storage of the passwords. The ability to steal the passwords will be judged on whether passwords are stored encrypted, hashed, unencrypted or not stored at all.

To make the experience as sound as possible, multiple operators will do the same operations to minimize the chance that the skill level of the operators influences the results. The average will then be calculated to estimate what it would look like in the real world. The fastest times of each operators will then be compared to our automated approach.

We think that these tests evaluate the solution on a more realistic scale for the Internet of Things and will demonstrate that the common security practices need to be reviewed and evolve to face the challenges that this new computing paradigm brings to the internet.

Results

We ran the whole password changing procedure with the manual and automated approaches with the exact same steps. For each test, we changed the password of 5 devices. Tests were all ran in a network with a 117.72 Mbps downloading speed and a 37.12 Mbps uploading speed[15] on a wired interface to reduce the impact of fluctuations in network speed and reliability. The host machine is a Dell XPS P31F with 16GB of RAM and an Intel i7 2.3 GHz processor. The operating system is Arch Linux 4.14.3-1-ARCH with Docker 17.10.0 and Python 3.6.3.

1. *Automated approach*

Aiakos changed the password in an average of 169.4 seconds in our lab tests using the Docker containers as target devices. The server, client and containers were all on the same physical machine. All containers were in the same subnet that was reachable by the host computer. Each device had a different IP to mimic the use of multiple physical devices.

Test number	Time (s)
1	171
2	170
3	171
4	167
5	168

Table 1: Automated tests results with Aiakos.

2. Manual approach

For the manual operators, we had two experienced computer students. Steps were given to the operators and commands were copy-pasted to reduce time as much as possible. To generate the passwords, the Aiakos server was used with the same criterias that Aiakos used to build the passwords automatically.

Test number	Time (s)	
	Operator 1	Operator 2
1	423	927
2	338	677
3	287	321
4	257	261
5	246	253

Table 2: Manual operators results.

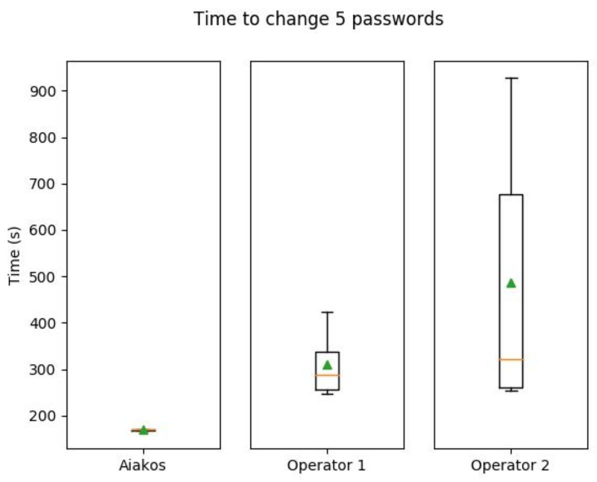

Figure 2: Time to change 5 passwords with the different approaches and operators.

Analysis

From the results we gathered, we can see a difference in time between manual and automated approaches. Indeed, the automated method gave results which were faster than the manual approach for each test case. Also, automated times were more consistent than the manual approach which had a significant variance between operators and between each runs of the same operators. This can be explained by a difference in the operator's abilities and familiarity with the procedure. We observed that the operators both had consistently better times each try. We did not extrapolate those results because operators thought they developed optimal processes to change the passwords without scripting involved. Better evaluation of the soundness of this conclusion is touched in the future work section of this article. When taking this into account, the last two runs were more consistent and should not improve greatly if we were to make more runs.

The fastest times were 246 seconds for the first operator which means an average of 49.2 seconds per device and 253 for the second operator for an average of 50.6 seconds per device. By extrapolating these results and increasing the number of devices to change the passwords on to a thousand devices, we obtain a total time of 13 hours and 40 minutes for the first operator at best without breaks. Furthermore, the second operator would total 14 hours and 3 minutes for the same number of devices. As the scale of the IoT is closer to that range, our experiment helps to illustrate the fact that automated solutions closer to this paradigm reality need to be developed. Without it, the security problem of having weak and default passwords could remain a problem for the years to come because of the cost to fix those solutions by hand without proper tools is simply too big without incentives. With the automated approach using Aiakos, no technician has to do this dreaded work which we feel would be an possible incentive for large companies and ISPs that maintain a large number of IoT devices to adopt a better security posture.

We can see that the second operator has a much higher standard deviation that could be explained by the lack of experience on the testing machine and its tools. With better experience, his results were much closer to those of the first operator and would indicate that those results would be close to the best results a human operator could have on a small number of devices with the goal to change the passwords as quickly as possible.

As for the security comparison of both approaches, we decided that testing with the same procedure for both the automated and manual approach would be more realistic as the time variable is more easily evaluated. We felt that a human operator would also feel the need to have a secure process to change the password as much as we wanted to for the automated approach with Aiakos.

From our analysis, the main bottleneck of the automatic procedure was the download and installation of SSH on the devices. On devices that would not require us to install SSH the time would be greatly reduced. Having significantly faster results and easy ways to add devices, we can extrapolate the scale of our test and say that our solution, compared to the manual procedure, is more effective therefore scalable.

We also observed some other facts that help to justify an automated approach. First, the operators made mistakes in some cases that impacted their times and some runs had to be redone completely because they did not succeed to change the passwords successfully. With a larger scale of devices, operators could experience fatigue that could increase the number of errors and ultimately reduce the chance of having the incentives to change the passwords and such a large number of devices. We could also explain some mistakes by the stress that the operators felt because they tried to make their time as fast as they could which would not necessarily transfer into the real world in a normal technician workplace.

It also capital to discuss technical challenges we faced during the establishment phase of our secure password changing process. The first problem was that, even on a full Linux user-space that was provided by the Alpine OS, some utilities behaved differently as they would do on a normal Linux install. Some command line arguments were not available or had to be placed differently as it was the case with the wget utility. This shows that a certain level of changes have been made for common utilities on embedded platforms and could partially explain the difficulty to assess and fix security issues on those platforms.

Future work

A first improvement to our design would be performance related. Currently, we process one device at a time. In order to drastically improve the time, we could process multiple devices in parallel. This would greatly improve efficiency since our procedure is highly parallelizable.

Our implementation is also currently tightly coupled to the specifications of those routers and lacks flexibility. One way to improve that would be to decouple from the main application the implementations specific to devices. That would improve our software design and help us to eventually support many devices.

The next improvement would be to support more devices. Indeed, right now we only support a certain portion of IoT devices that have a Telnet connection and have everything needed to install SSH and do HTTPS requests. Of course, that is not the case for all devices. We would have to adapt our implementation for the specifics of those devices in order to support them. For example, we had access to a package manager which we used in our procedure to install the SSH server and that would not be perfectly transferable to smaller and more limited devices. Implementing the process to change passwords securely to other devices would help to identify what are the limitations of those devices in terms of security capabilities and the possibility of fixing the problems they have. We think that some devices could be so limited that our approach could not be implementable and we would need to change to another approach that could be network-based with proxies acting as gatekeepers to the devices with weak credentials. This device would be less limited and could authenticate the user to the vulnerable devices with proper authentication.

Lastly, since IoT devices can be used by many people, professionals and non-professionals alike, it would be a good idea to improve the ease of use of our solution. It would also be a good idea to build either a desktop or a web app with a GUI that is easier to approach.

On the testing side of the experiment, we would want to test and compare with the parallelized automated approach to hopefully further the gap between the automated and manual approach.

We would also like to test with a larger number of operators to establish a better statistical soundness of the results of our experiment. That would involve more than 30 operators with varying degrees of experience but all of a minimum level that would be acceptable to qualify as a technician.

The number of devices should also greatly increase to study operator's fatigue and its effect on password changing time. This would help to build a better model to extrapolate the effect of the scale of the number of devices in the IoT world on the realism of security methodologies.

Conclusion

Our experiment helped to highlight some possible solutions to the problem of already-deployed IoT devices with weak credentials. We developed a tool to change automatically those devices securely without the need of human interaction which would help incentivize the ISPs and other companies maintaining a large number of this class of devices to fix a current problem that affects the security of the internet as a whole and opens the door to low skill actors to have a big negative impact. We also highlighted some problems we feel should be studied by the academic community without abstracting them like technical challenges faced when developing and implementing security technologies for the IoT world.

References

[1] MalwareMustDie, ”MMD-0056-2016 - Linux/Mirai, how an old ELF malcode is recycled..”

http://blog.malwaremustdie.org/2016/08/mmd-0056-2016-linuxmirai-just.html, 2016

[2] OWASP, “Internet of Things Top 10 infographic”, https://www.owasp.org/images/8/8e/Infographic-v1.jpg, 2015

[3] JGamblin, “scanner.c”, https://github.com/jgamblin/Mirai-Source-Code/blob/master/mirai/bot/scanner.c,

[4] https://arstechnica.com/information-technology/2017/08/leak-of-1700-valid-passwords-could-make-the-iot-mess-much-worse/

[5] Gartner, “Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016”, https://www.gartner.com/newsroom/id/3598917, 2017

[6] M. Patton, E. Gross, R. Chinn, S. Forbis, L. Walker, and H. Chen, "Uninvited Connections: A Study of Vulnerable Devices on the Internet of Things (IoT)," in Joint Intelligence and Security Informatics Conference (JISIC), 232-235, 2014

[7] https://internetinitiative.ieee.org/images/files/resources/white_papers/internet_of_things_feb2017.pdf

[8] https://blog.shodan.io/telnet-is-dead-long-live-telnet/

[9] P. P. Pereira, J. Eliasson, and J. Delsing, “An authentication and access control framework for CoAP-based Internet of Things,” in Proc. 40th Annu. Conf. IEEE Ind. Electron. Soc. (IECON), Dallas, TX, USA, Oct. 2014, pp. 5293–5299

[10] https://information.rapid7.com/iotseeker.html

[11] https://docs.python.org/3/library/secrets.html

[12] https://www.shodan.io/search?query=%22default+password%22+port%3A%2223%2

[13] Intel, “Guide to IoT”, https://www.intel.com/content/www/us/en/internet-of-things/infographics/guide-to-iot.html, 2017

[14] https://www.owasp.org/index.php/About_The_Open_Web_Application_Security_Project

[15] SPEED OF ME, “SpeedOf.me”, http://speedof.me/, 2017