Distributed Security Management solution for the Internet of Things

Kuber Kohli, Kelly Fallon, Priyank Thavai
CIS 657 : Principles of Operating System
Group 3: Project Report

***Abstract*—With continuous extensive research being done on the Internet of Things (IoT) to achieve the reality of smartcities, smarthomes, smartcars etc. which has been long anticipated by the researchers and developers, there are still some concerns that needs to be resolved before those anticipations can be actualized. One such big concern is *Security*. This Project is aimed at providing an easy, distributed, fault tolerant and resilient solution for managing Operating System security features for interconnected IoT devices using the *consensus algorithms* and a system level security management demon. Systems having these kind of solutions will be able to support long lived and unattended operation which is a basic need for IoT devices.[1]In many instances device software management is provided by application middleware.[3] By moving clustered security management to the OS we allow participation in distributed networks natively by providing a common set of security constraints to member hosts. This approach allows the local security constraints commonly provided by operating systems to be dynamically applied and managed across a common security domain of IoT devices.**

I. Introduction

In contemporary trends, the proprietary IoT software update solutions that exist from device vendors generally follow client server model. This model depends on the availability of control servers. As well the client server model in many instances may not scale either well or economically with many thousands of clients. In current practice it is not uncommon for IoT devices to not be updated or managed. Additionally, it is common that legacy applications are not fit for use on the internet, such as telnet, to be used by IoT devices [4]. While there are many application level frameworks for managing software distribution and configuration, the operating system's security functionality has a criticality that benefits from an isolated and OS managed approach as it provides a security configuration & management domain which is separate from application layer software distribution and management.

This project proposes the implementation of *Raft Consensus Algorithm* as a solution for security management in IoT. *Consensus algorithms allow a collection of machines to work as a coherent group that can survive the failures of some of its members.*[5] The Raft consensus algorithm essentially operates on a simple logic that a leader is chosen from among the devices in distributed system and that leader then dictates how the changes will be made, though its implementation is a lot more complex and requires much more details.

Furthermore, simply implementing the raft consensus algorithm may not prove to be fruitful. The algorithm should be implemented in a way that the following problems are also addressed :

1) To meet realistic system requirements that derive from long lived and unattended operation, IoT devices must be able to continue to operate satisfactorily in the presence of, and to recover effectively from, security attacks. Solutions may even require downloading new code. [1]
2) In addition to the security and protection aspects of the Internet with which we are all familiar (such as communications confidentiality, the authenticity and trustworthiness of communication partners, and message integrity), other requirements would also be important in an Internet of Things. We might want to give things only selective access to certain services, or prevent them from communicating with other things at certain times or in an uncontrolled manner; and business transactions involving smart objects would need to be protected from competitors prying eyes.[2]
3) An Internet of Things potentially has a larger overall scope than the conventional Internet of computers. But then again, things cooperate

mainly within a local environment. Basic functionality such as communication and service discovery therefore need to function equally efficiently in both small scale and large-scale environments. [2]

4) The operating systems of distributed systems and Internet of Things (IoT) need to be able to utilize security services in a manner that provides fault tolerance and resilience in an automated manner. There should not be a single point of failure. Minimum manual intervention must be required.

II. Security integrated within Operating System

The heterogeneity of the devices in the IoT has its own advantages but at the same time it raises a lot of problems and challenges for the developers. To develop a strong and efficient mechanism to ensure security and privacy becomes a complex task. All the devices in the Internet of Things will be smart devices and all the devices will have one thing in common - all these devices will be running on some kind of operating system. The operating systems will essentially be defining the functionality of the devices and how they communicate with each other. So it makes perfect sense to integrate the security mechanism deep within these operating systems so that security becomes a part of the backbone of the IoT. As discussed before the OS managed approach has its own advantages which benefits by lowering the costs for implementing and managing security solutions in IoT. Having a third party or an application level security solution even though feasible complicates managing security in IoT. It will also encourage different companies to develop their solution resulting in a chaos. Different devices in IoT running different security solutions will compromise the security and availability of the whole network. While on the other hand an OS managed will bring uniformity across all devices. But it has to be ensured that the same security solution has to be developed for different types of operating systems or at least the nature of the security solution should be same so it would make the security look like its machine independent.

III. Scope and Goals

The time constraint constrained the scope and implementation of the project. The project is aimed at proving that the proposed solution is a feasible option that can be implemented at low costs and that it will require less human interaction as the changes are automated by the consensus algorithm. Although some further research and development, discussed later in the paper, will be required before a commercial version of the solution can be developed and tested.

The scope of the project involves configuring an implementation of raft consensus algorithm in a distributed environment. The consensus algorithm should work efficiently and should synchronize the changes in one machine across all the devices in the distributed system. Each machine in the system will be comprised of a meta-data repository that will contain the key value pairs (KVPs). A custom service will continuously monitor the KVP meta-data repository and apply to the local node any new configuration directives detected.

A. Goals :

1) Demonstrate distributing security meta-data store via consensus algorithm as an OS system service & modify an existing service[APT] to use this updated security meta-data store. As well demonstrate that the cluster supports dynamic node membership where a nodes may be added to the cluster as well as leave the cluster on an ad-hoc basis.
2) Enable multiple security domains within the interconnected devices which will provide more granularity and control. This will allow one set of clustered security meta-data to contain information for multiple distinct sets of IoT devices that require separate security configuration.
3) Provide authentication and integrity assurances to each KVP in the meta-data repository by having the security configuration service require each KVP to be signed by an authorized security configuration key prior to its application to the OS. This accommodates the fact that any cluster member node may write meta-data for replication by assuring the

only KVP that are applied are those that are submitted by an authorized source. As well we would like to look at the application of using block chains and hashing to verify our meta-data integrity.

IV. Resources & Technology to be used

The *CentOS Linux* operating system was extended for use in this project. The virtual machines hosted in the Amazon EC2 Web Services Cloud Infrastructure were used to simulate an Internet of Things (IoT). There are three such machines that act as low cost substitute for IoT devices. Due to limited time frame an existing implementation of Raft in the *CopyCat*[7], an open source software, was used. CopyCat providing consensus and distribution of a meta-data store demonstrates the possibility of having a more optimized implementation of this functionality as a core OS service. Java programming was used to develop a custom security configuration service that uses system level access to receive and validate configuration data from our distributed store and apply it to our operating system service configuration files.

V. Implementation & configuration

Each of the three virtual machines was provisioned as a ”t2.small” instance type within the Amazon EC2 cloud. This allowed each host one virtual processor and two gigabytes of RAM. A persistent filesystem on each host was used to install the CentOS7 operating system, as well as our service, and as a backing store of our distributed state machine on each host. Firewalling was configured to allow TCP/IP networking between the hosts and our custom SysVinit init script was install on each host so that our service starts with the OS on boot. This configuration simulates the Internet of Things at a small scale. Each host is a simulated IoT device allowing us to test the security management configuration service's ability to update the hosts.

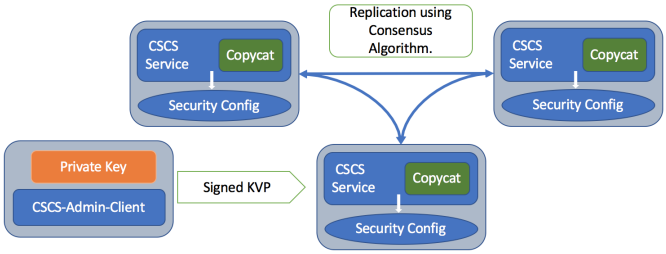
Once the hosts were built and configured we installed our service. The Clustered Security Configuration Service (CSCS) is a Linux system service authored in Java using the Spring Boot framework[9]. The source code for our service is available in Github in a repository titled ”ClusteredSecurityConfigService”.[10] Our service implements both the Copycat server and client processes providing both a fault-tolerant state machine replication using the Raft consensus algorithm as well as a way to query values from this clustered datastore. Once our service is started on the hosts the CopyCat server process negotiates a leader host in the distributed system and initiates synchronization of the distributed state machine comprised of KVPs. CSCS logs output to files within the /var/log/ directory structure. We monitored these log files to validate the success of cluster communications and data synchronization.

We designed the CSCS to contain modular services that are launched through an internal scheduler. We discussed three potential services for our demonstration. The first which we fully implemented in a Java class named YumConfigService. This class is executed every five minutes by our scheduler. The YumConfigService uses a Copycat client instances to query the distributed key store and retrieve any keys for its configuration domain that are labeled as containing data for the Yum configuration. The value of each KVP is a tuple containing both a plain text data string as a well as a SHA256withRSA signature of that data. It then uses the RSA public key that was distributed with the service to validate the signature. If the signature is valid the data is assumed to have been provided by a trusted source. If valid keys were found defining one or more Yum software repositories then the YumConfigService updates the Linux operating system's Yum configuration files located in the /etc/yum.repos.d/ directory. In this manner our service controls the source of software updates for the simulated IoT devices. Two other modular services that we created stubs for but did not implement are KernelConfigService and IptablesConfigService which can be implemented in the same manner to control kernel parameters and firewalling.

The management of keys within our state machine is performed by a command line Java utility that we authored the Clustered Security Configuration Service Admin Utility (CSCSAU). This utility uses an RSA private key to sign key values and submit

them using an implementation of the Copycat client to the cluster. The source code for CSCSAU in Github in a repository titled "cscs-admin-utility".[11]

Given below is a pictorial representation of the working of the solution and how the changes flow in the system.



VI. Inference

The experiment done makes it evident that the proposed solution to implement Raft consensus algorithm as a security management solution for IoT is a feasible idea. Such a solution is not only economic from a monetary point of view but also saves a lot of human efforts. The solution brings in automation, stability and reliability in security management. *All the three primary goals were successfully achieved.* An ideal version of the solution will need only one admin who can roll out updates from any one machine and the update will be communicated to all the target devices. The results showed that any new security update or new configuration roll out can be done with communicating the same to only the leader devices in their respective domain who in turn will communicate the change to all other devices in their domain. This way all the shortcomings of the client server model can be avoided without losing any of its advantages. The distributed system could be broken down in smaller domains and then a change can be communicated to that particular domain. This feature is desirable in a situation when a new security update is targeted to only a particular kind of devices in IoT or if the update is to geographically restricted, then all the devices can be assigned to a domain and changes can be updated easily. Thus the project was successful in achieving its aim of proving that the idea of developing such solution is a viable option; though further research, discussed later, is required before the product will be industry ready.

VII. Challenges faced

Due to time constraints implementing from ground up a raft consensus algorithm optimized for this particular project was not possible. Hence an open source software implementing consensus algorithm was the only other option. Looking for such software we found LogCabin which fit best the requirements. So the LogCabin[8] implementation was initially used to setup the consensus algorithm in our distributed system. Soon after all the configuration and testing were done to achieve our primary goal and when it was time to integrate our custom script to communicate with the LogCabin we realized that both are not directly compatible. All the services in LogCabin were implemented using C++ while our application was authored in Java. So both modules were not directly compatible. Even though they can still be integrated but that would require much more complex coding and time. Pursuing this solution would result in falling back on deadlines. Fortunately CopyCat another open source software for raft consensus algorithm matched our requirements and put us back on track. Developing a custom script that interacts with an open source software is a complex task. Moving past this complex task, another question needed to be answered before all the pieces were put in their place. The custom script was required to check for any new configurations available repeatedly. But how often does it needs to check for security updates ? Every few minutes even though seems to make sense, also results in poor overall performance as the custom script should not consume much of the memory of the device its running on; making check interval too big will also destroy the purpose of the solution. So a compromise has to be made between the need to check for a security update and performance. For the experimental purpose the interval was set to a few minutes as the experimental devices were not being used for any other purpose so performance drops becomes trivial but for commercial use the check interval needs to be chosen circumspectly.

VIII. Impact & stakeholders

1) The Internet of Things is predicted to be the next wave in the era of computing .[6 Managing & securing such a huge number of interconnected devices will be a high value to multiple industry sectors such as health, entertainment, utilities, appliances. As well community and governance requires reliable and secure IoT devices for utilities, infrastructure, monitoring, transportation, and emergency services. The impact of having secure IoT devices allows these devices to be used consistently and reliably.
2) Stakeholders: Home and business users, Smart Device vendors, IoT infrastructure providers, Enterprises, and Communities.

IX. Future Prospects

Further research to implement optimization, testing, and large scale validation will require sufficient time and funding. With enough time and funding, further research can also be done to include some more features in the solution like detecting a compromised node. This feature will be desirable in a situation where a device has been compromised and some hacker is trying to upload malicious files from that the system. The solution in that case should be able to detect this anomaly and find out which particular node is responsible for this anomaly. Performance can also be further improved with use of some complex programming techniques and an optimized implementation of the consensus algorithm. A better client can also be developed that further reduces the human efforts for managing the security in an IoT. Any such graphical interface will make tasks more simpler and easy. Strong encryption can be also implemented to sign the configuration files to add another layer of protection in the solution. The devices in that case will have a way to check if the security update is indeed coming from a trusted host and can be updated in the device. Hence, we believe that with enough research done in this direction, the solution can be commercialized for managing security in the Internet Of Things.

References

[1] J. Stankovic, Research Directions for the Internet of Things, IEEE Internet of Things Journal, Vol. 1, No 1, Feb 2014
[2] Mattern, F. and Floerkemeier, C. 2010. From the Internet of Computers to the Internet of Things. From Active Data Management to Event-Based Systems and More, 242259.
[3] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. Computer networks, 54(15):27872805, 2010.
[4] A. Gheorghe, The Internet of Things: Risks in the Connected Home, Bitdefender Case Study, February 2016
[5] D. Ongaro and J. Ousterhout. In search of an understandable consensus algorithm. In USENIX Annual Technical Conference.
[6] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, Internet of things (IoT): A vision, architectural elements, and future directions, Future Gener. Comput. Syst., vol. 29, no. 7, pp. 16451660, 2013.
[7] http://atomix.io/copycat/
[8] https://github.com/logcabin/logcabin
[9] https://projects.spring.io/spring-boot/
[10] https://github.com/kjfallon/ClusteredSecurityConfigService/
[11] https://github.com/kjfallon/cscs-admin-utility/