A web-based policy management tool to enable simple and robust management of end-user and server devices - Interim Review

William Stuart Oldham

23rd February 2022

University of Lincoln, School of Computer Science
19693813@students.lincoln.ac.uk

Abstract. A key part of any IT infrastructure is the system used to manage devices owned by the company. Such systems typically adopt a policy based approach where each policy is evaluated against each device and checked for compliance. Should a device not be compliant with the policies, a set of remedial actions are run to bring the device into compliance. Alongside knowing whether devices are compliant, knowing information about devices that are being managed allows IT professionals to work more efficiently. This project proposes a web-based management system used to manage and execute policies on client devices with minimal client device interaction.

Keywords: Software Development · Policy Management · Web GUI

Word Count:

2277 words

1 Introduction

Throughout the IT industry, a key part of a system administrators job is to manage a number of end-user client and server devices. While managing a device can consist of a number of tasks, one task is ensuring the device is compliant with policies that the system administrator puts in place. For large IT teams, there is typically a group of people designated with the task of monitoring compliance and modifying policies for the organisation. However, in smaller IT teams this is not always possible - especially if there is quite a high ratio of devices to IT staff.

This project aims to produce a piece of software which lowers the entry barrier for mass device automation allowing a user to distribute policies to different computers on a network. Policies in this context refers to a set of instructions determining what state a computer should be in. If a computer is not in the correct state, the policy will be run to make the computer "compliant". Existing systems that do this are either large complex pieces of software designed for enterprise, or are command line utilities which are not beginner friendly. This project will be designed in such a way that installation and operation of every part is as simplistic as possible.

There will be 3 main components to the application each tying into each-other:

- Back-end API server - Responsible for acting on data and communicating with the database.
- Front-end application - Allows the user to run actions through a web-based interface. This interfaces with the API server to perform its tasks.
- Client - A native executable which sits on a managed computer and is responsible for running policies assigned to that computer. This communicates with the API server to send and receive the data required for operation.

2 Background and academic literature review

Part of the basis for this project is experience gained through my role as a sole IT manager for a secondary school. I have found that having tools that presented simple ways to do day-to-day tasks reduces the amount of time spent on a given activity freeing up time for other jobs to be completed.

This project is a form of network automation to deploy configuration. Network automation is defined as "the process of automating the configuring, managing, testing, deploying, and operating of physical and virtual devices within a network" (Cisco 2020) covering a wide range of disciplines and devices. The focus for this project is specifically on the configuration management and inventory collection of computer devices within the context of a small business or personal network. A number of products exist in this space such as Ansible, SaltStack and Attune. This project will build upon each products pros and cons.

2.1 Review of existing products

Ansible[1]

Ansible is a "radically simple IT automation engine that automates [...] configuration management, application deployment, [...], and many other IT needs." (Ansible 2019). The main focus is to have a simple framework to manage servers which is enhanced through built-in and 3rd party plugins. Ansible's user interface consists of a command line interface and YAML files (Known as "playbooks") (Ansible 2019) which allows an administrator to describe the actions that clients should execute.

Ansible relies on a "push" model where the managing computer pushes out a configuration to a server. While Ansible can be configured to work in a "pull" model (Ansible 2021), this relies on a scheduled job which pulls playbooks from a Git repository meaning that there is no status checking or inventory collection of the client computers. Furthermore, a push model means that static hostnames or IP

[1] For more information the reader is referred to https://www.ansible.com/resources/get-started

addressed need to be set for client computers, which is not always possible or preferable with end-user devices such as desktop PCs. Ansible communicates with clients without using an agent by utilising SSH connections. The use of SSH means that SSH asymmetric keys can be used allowing the software to communicate with multiple servers using one set of credentials. This is particularly advantageous towards saving time since only one set of credentials has to be maintained for an unlimited amount of servers.

SaltStack[2]
SaltStack is an infrastructure management tool that makes use of an installed agent (Named a "Minion" in the Salt ecosystem) to install configuration known as "salt states" on clients (SaltStack 2021).

SaltStack also makes use of a push model like Ansible. However, since the clients are agent-based, a static IP for the client is not required. Since the agent permanently resides on the client, it also allows SaltStack to target clients based on OS information called "Grains" (SaltStack 2021).

SaltStack's agent installation is a relatively straight forward process. To connect to the "master" server, minions try to resolve the `salt` host-name on the local network (SaltStack 2020). To ensure secure communication, there is a key-verification process which must be manually completed for each minion (SaltStack 2020).

ServerTribe Attune[3]
ServerTribe's Attune is one of the only reviewed projects that was a GUI application instead of a command line product. Attune is described as "the easiest to use and fastest server/node automation solution" (ServerTribe 2021).

Attune's goal is to allow administrators to run "blueprints" which can be used to either run scripts or transfer files to remote "nodes". It natively supports both Linux/Unix and Windows nodes supporting both SSH and WinRM connection methods.

It is a simple concept which allows for fantastic flexibility in the amount of actions an administrator can run. Unfortunately, upon reviewing the product, it was clear that the user experience of Attune had not been focused on. The product failed on 4 of Nielsen's Usability Heuristics: Match between system and the real world, consistency and standards, error prevention and recognition rather than recall (Nielsen 1994).

2.2 Takeaways from the review of existing products

From the review of existing products, there was a number of points that will be used to inform the development of the project. These are as follows:

The lack of GUI based products
Out of the products that were reviewed, only ServerTribe Attune adopted a GUI first approach. Products such as Ansible and SaltStack both have third-party GUI add-ons (Ansible Semaphore (2022) for Ansible and Dondorp (2022) for SaltStack) but these add-ons are often an after-thought with large amounts of functionality missing. For example, the GUI for SaltStack only allows running of jobs and viewing of configuration. You are unable to edit any configuration from the GUI.

Agent-less vs Agent-based
Comparing products such as Ansible and SaltStack showed the difference between connecting to clients with an installed agent compared to without. The key difference in the context of this project is that a system that doesn't use agents requires clients to have a known IP address or DNS name. In agent based systems, this is not required and allows configuration of clients which are mobile. For this reason, this project will be agent-based to ensure maximum flexibility. While it will not be implemented in this project, agent-based systems also allow clients to be configured which don't lie within the local network. Since a client can connect to a publicly accessible endpoint from anywhere in the world.

[2] For more information the reader is referred to `https://docs.saltproject.io/salt/user-guide/en/latest/topics/overview.html`

[3] For more information the reader is referred to `https://www.servertribe.com/get-started/`

Client to server communication

All of network automation has the concept of a "client" and a "server". The client is the system being configured and the server is the central storage of configuration to be distributed to clients. When referring to client and server communication, the terms "push" and "pull" are often used to describe the direction of data flow. Martin-Flatin (1999) describes the push and pull models in the context of SNMP monitoring where push is when the client pushes data to the server. Within the context of configuration management, push is where the server sends configuration to the client and pull is where the client requests configuration from the server. In line with using agent-based communication, a pull model will be used where the agent will request configuration from the central server.

2.3 End-to-end encryption

An important part of network automation is ensuring communication between parties maintains both integrity and confidentiality (Samonas & Coss 2014). For this reason, end-to-end encryption is best known to provide protection for digital communications which may be intercepted by 3rd parties (Bai et al. 2020, 210). Using a asymmetric encryption method such as RSA (Rivest–Shamir–Adleman) or ECC (Elliptic-curve cryptography) not only ensures data cannot be seen by 3rd parties, the asymmetric nature of the keys also allows both parties to verify the received data to ensure a man-in-the-middle attack has not been performed (Simmons 1979, 306). Referencing studies by Singh et al. (2016) demonstrated that ECC out-performed RSA cryptography in both speed and key size. With the key size of ECC being between 6-30 times smaller than that of an equivalent RSA key (Singh et al. 2016, 626), it is the obvious choice for this project to provide the best security in minimum space.

3 Supervision meeting logs

Meeting Date	**Meeting Details**
Week 3 - 26th October	Initial meeting to discuss project idea. After presenting ideas, my supervisor suggested that market research should be done to see what the niche in the market is.
Week 5 - 9th November	– Meeting was scheduled but due to enhancement week did not take place. –
Week 6 - 16th November	Presented plan to do project which fit into a niche that was researched in the market. Technologies to be used were discussed with my supervisor and the recommendation was made that the number of programming languages were reduced from 3 languages to 2. Academic background was discussed. It was agreed that some working code would be produced for the next meeting as well as a draft literature piece.
Week 8 - 30th November	Showed my supervisor a demo of the basic API and web-page structure. The Gantt chart was discussed with a particular focus on risk management to ensure the project remains on track. Parts of the Gantt chart were identified to be part of the minimum viable product. Talked about restructuring the Gantt chart to prioritise MVP components. Talked about parts of the program to be completed by the next meeting: Authentication and Authorisation, Communication with Clients and a Login System.
Week 10 - 14th December	– Meeting was scheduled but due to enhancement week did not take place. –
Week 14 - 11th January	Further discussion about the MVP and making sure I am on-track for completion. At time of meeting, I was behind on the Gantt chart but confident it would be caught up. Discussed about the 2nd assignment and how I might go about finding literature for this software development project. Suggestion was made to look at specialised areas where the product could be useful such as robotics.
Week 16 - 25th January	Demonstrated a working back-end and front-end relationship. Discussed the stage in the project management I was at and discussed what will be worked on in the coming weeks. Literature review was discussed and how relevant literature could be tied in to produce the literature review.
Week 18 - 8th February	– Meeting was scheduled but due to exams week did not take place. –

4 Project progress

At the time of writing (Updated: 15th February 2022), progress for the project is good overall with the 3 different components each having a large portion completed ready for testing. The timeline for each component can be seen in fig. 1 with the progress as of February 15th 2022 shown in fig. 2. A development log can be seen below.

The current focus within the project is finalising the client to back-end communication, finalising the front-end design and implementing this design into both the back-end API and the front-end rendering.

On each week in the development log, there is a key to show at a glance which component had been worked on each week:

- Client component: ●
- Back-end component: ▲
- Front-end component: ◆

4.1 Development log

Week 1 - w/c 22nd November ▲
This week involved creating the Node.JS back-end project and setting up the dependencies for serving a web API. A JSON Web Token authentication system was implemented in the back-end as well as a request validation system.

Week 2 - w/c 29th November ▲
Database entities were planned, created in SQL migrations, and had models created for each entity. A PostgreSQL client was also setup within the back-end. Within the API server, authentication endpoints were created and linked to the database.

Week 3 - w/c 6th December ▲ ●
API endpoints were created for the different entities and an enhanced Express HTTP middleware was created to assist with creation of future endpoints. As well as this, the protocol specification for the client to server communication was started as well as how end-to-end encryption would be established.

Week 4 - w/c 13th December ▲ ●
The SQL migration and models were created for the "client" and "policy" entities. This week, the basic Golang project was created for the client program also.

Week 5 - w/c 20th December ▲
A TCP server was implemented in the back-end application and the packet structure was further researched and documented.

Week 6 - w/c 27th December ▲
The back-end log format was improved for better readability. The back-end also had packet decoding and encoding added into it. Following this, the end-to-end encryption was also implemented.

Week 7 - w/c 3rd January ● ◆
Following from the implementation of packet handling on the back-end in the previous week, the client had the TCP client created, packet handling implemented and encryption implemented.

This week, the front-end design for some of the pages was created and implemented using Tailwind CSS, Nuxt.js and Vue.js.

Week 8 - w/c 10th January ● ◆
Table components were created within the front-end framework for the user and policy listing tables.

Within the client, further work was done on the TCP packet handling and execution flow.

Week 9 - w/c 17th January ▲ ● ◆

Work was completed on the client which enabled the client to register with the back-end server, send heart-beats to the server and including OS information in the heart-beats.

Authentication middleware was added to the front-end to prevent navigation to restricted pages without being logged in. Buttons for the navigation bar were styled and implemented. A search filter for table components was also created.

The back-end's handling of SQL errors was improved. The SQL schema was also updated to reflect the latest entity structure.

Week 11 - w/c 31st January ● ◆

Within the front-end, methods were added to request data from the back-end server and render it in the front-end.

Policy evaluation for commands and packages was implemented in the client to allow it to process and execute received policies. Storage for the received policies was also implemented to allow the application to keep track of which policies have already been executed.

Week 12 - w/c 7th February ▲ ●

The back-end was modified to sent policies in heart-beats to the client.

Following from this, the client was modified to parse the received policies and run each one through the evaluation logic. The client's policy storage was also modified to use a map instead of an array to make it easy to not have duplicate entries.

5 Figures

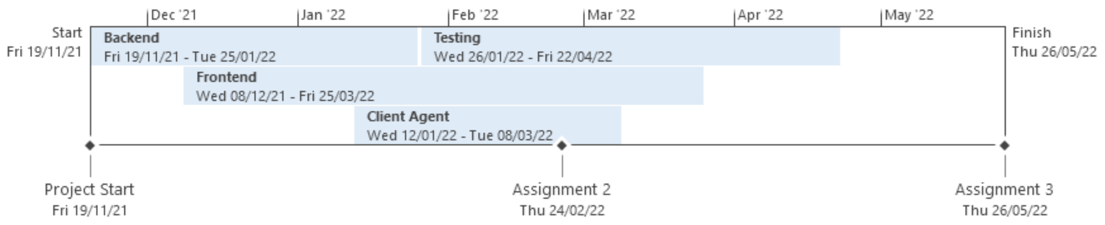

Fig. 1. Initially proposed project timeline

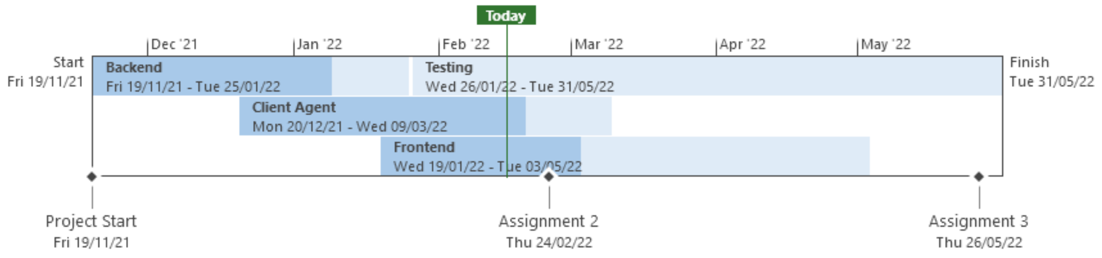

Fig. 2. Timeline as of February 15th

Bibliography

Ansible (2019), 'How ansible works'. Available from https://www.ansible.com/overview/how-ansible-works [Accessed 11 Feb 2022].

Ansible (2021), 'Intro to playbooks'. Available from https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html#ansible-pull [Accessed 11 Feb 2022].

Ansible Semaphore (2022), 'Ansible semaphore: Modern ui for ansible.'. Available from https://github.com/ansible-semaphore/semaphore [Accessed 20 Feb 2022].

Bai, W., Pearson, M., Kelley, P. G. & Mazurek, M. L. (2020), Improving non-experts' understanding of end-to-end encryption: An exploratory study, *in* '2020 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)', pp. 210–219.

Cisco (2020), 'What is network automation?'. Available from https://www.cisco.com/c/en_uk/solutions/automation/network-automation.html [Accessed 9 Feb 2022].

Dondorp, E. (2022), 'Saltgui: A web interface for managing saltstack based infrastructure.'. Available from https://github.com/erwindon/SaltGUI [Accessed 20 Feb 2022].

Martin-Flatin, J.-P. (1999), *Push vs. pull in web-based network management*, *in* 'Integrated Network Management VI. Distributed Management for the Networked Millennium. Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management.(Cat. No. 99EX302)', IEEE, pp. 3–18. Available from https://ieeexplore.ieee.org/abstract/document/770671 [Accessed 7 Nov 2021].

Nielsen, J. (1994), Enhancing the explanatory power of usability heuristics, *in* 'Proceedings of the SIGCHI conference on Human Factors in Computing Systems', pp. 152–158.

SaltStack (2020), 'Configuring salt'. Available from https://docs.saltproject.io/en/latest/ref/configuration/index.html#configuring-salt [Accessed 13 Feb 2022].

SaltStack (2021), 'Salt in 10 minutes'. Available from https://docs.saltproject.io/en/master/topics/tutorials/walkthrough.html [Accessed 11 Feb 2022].

Samonas, S. & Coss, D. (2014), 'The cia strikes back: Redefining confidentiality, integrity and availability in security.', *Journal of Information System Security* **10**(3).

ServerTribe (2021), 'Node automation & orchestration: Attune community edition - servertribe'. Available from https://www.servertribe.com/ [Accessed 11 Feb 2022].

Simmons, G. J. (1979), 'Symmetric and asymmetric encryption', *ACM Comput. Surv.* **11**(4), 305–330. **URL:** *https://doi.org/10.1145/356789.356793*

Singh, S. R., Khan, A. K. & Singh, S. R. (2016), Performance evaluation of rsa and elliptic curve cryptography, *in* '2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)', pp. 302–306.