Automatic Selection of Security-relevant Configurations

Bachelor's Thesis

Supervisor: Prof. Dr. Alexander Pretschner
Advisor: Patrick Stöckle
Email: {alexander.pretschner, patrick.stoeckle}@tum.de
Phone: +49 (89) 289 - 17314
Starting date: immediately

Fakultät für Informatik
Lehrstuhl 4
Software & Systems Engineering
Prof. Dr. Alexander Pretschner

Boltzmannstraße 3
85748 Garching bei München

Tel: +49 (89) 289 - 17314
https://www4.in.tum.de

Context

We at the chair of Software and Systems Engineering are currently developing together with an industry partner the Scapolite framework. The idea of the Scapolite framework is the definition of security-configuration guides in a human- and machine-readable form and the automatic implementation of these guides or the automatic check if a system is compliant to a guide. Usually, the input for a hardening process regarding security configurations is a guide with many rules. Here, every rule defines a value which has to be set for a specific configuration to make the system more secure.

The selection which configurations are security-relevant and which are not is conducted by experts who should know all security-relevant configurations of a specific system. If the experts who create the security guides are not aware of a specific security-relevant configuration or if they simply forget one, this security-relevant configuration may stay in an unsafe state although the system is configured compliantly to the security guide. Thus, this may lead to open attack points on a system which is regarded to be secure.

Goal

The idea of this bachelor's thesis is to automate this selection of security-relevant configurations. We want to develop a proof of concept which takes as input a set of configurations with additional information, e.g., description, help text, possible values, etc., and returns as output a set of configurations, which are possibly security-relevant. In other words, we want to define a function by rules or learn a function which maps a configuration enriched with its additional information to the security-relevant or not security-relevant.

To our best knowledge, this has not been tried yet. Two main problems here are the lack of the machine-readable definition of the configurations and their additional information and the lack of training data to learn or to deduct which configuration is security-relevant and which is not. During the development of our first proof of concept implementation for the automation of Windows-related security configuration guides, we had to reverse engineer the Windows' way of defining possible configurations in the form of administrative template files (ADMX/ADML). Thus, we now have for the Windows context exactly this machine-readable form of the configurations and their additional information needed for the aforementioned approach. Furthermore, with the security guides created by the Center for Internet Security (CIS) [1] and the IASE [2], we have the input to derive rules or learn which configurations are security-relevant. Thus, we think that the Scapolite framework is the perfect context to conduct this bachelor's thesis.

Example

In Windows 10 OS, we can configure a multitude of different settings. If we take a look at the *Group Policy Settings* under *Network \Windows Connection Manager*, we find 5 different settings.

1. `Computer Configuration \Policies \Administrative Templates \Network \Windows Connection Manager \Disable power management in connected standby mode`
2. `... \Enable Windows to soft-disconnect a computer from a network`
3. `... \Minimize the number of simultaneous connections to the Internet or a Windows Domain`
4. `... \Prohibit connection to non-domain networks when connected to domain authenticated network`
5. `... \Prohibit connection to roaming Mobile Broadband networks`

In the CIS Windows 10 (1806 version) guide, we find a rule that states that `Prohibit connection to non-domain networks when connected to domain authenticated network` should be *Enabled*.

They give the following as a rationale for that:

```
    The potential concern is that a user would unknowingly allow network traffic to flow
↪   between the insecure public network and the enterprise managed network.
```

When we take a look at the *description* (help text) of the specified setting, we find the following text.

```
This policy setting prevents computers from connecting to both a domain based network and a
↪   non-domain based network at the same time.
If this policy setting is enabled, the computer responds to automatic and manual network
↪   connection attempts based on the following circumstances:

Automatic connection attempts
- When the computer is already connected to a domain based network, all automatic connection
↪   attempts to non-domain networks are blocked.
- When the computer is already connected to a non-domain based network, automatic connection
↪   attempts to domain based networks are blocked.

Manual connection attempts
- When the computer is already connected to either a non-domain based network or a domain
↪   based network over media other than Ethernet, and a user attempts to create a manual
↪   connection to an additional network in violation of this policy setting, the existing
↪   network connection is disconnected and the manual connection is allowed.
- When the computer is already connected to either a non-domain based network or a domain
↪   based network over Ethernet, and a user attempts to create a manual connection to an
↪   additional network in violation of this policy setting, the existing Ethernet connection
↪   is maintained and the manual connection attempt is blocked.

If this policy setting is not configured or is disabled, computers are allowed to connect
↪   simultaneously to both domain and non-domain networks.
```

The question is now: Is there something – a keyword, specific words used together, etc. – in the text or the path that we can use to guess that this setting is security-relevant? If so, how can we use this *something* to train a classifier that tells us whether a given setting is security-relevant or how like this setting is security-relevant. And if the setting is security-relevant, can we guess from the description, what the secure values are and which values are insecure?

Formalization

In our context, a configuration decision (setting) $\gamma \in \Gamma$ is a tuple (p, d) in which p denotes its path, d its description; for an example γ' , c.f. above. We can map a configuration decision γ to its possible values

$$\beta(\gamma) = \{x_1, x_2, \dots\}$$

For the above mentioned example, this would result in

$$\beta(\gamma') = \{Enabled, Disabled, Not\ Configured\}$$

A system role $\theta \in \Theta$, e.g., the variant of a OS, can be mapped to a set of configuration decisions

$$\omega(\theta) = \{\gamma_1, \gamma_2, \dots\}$$

that can be configured for this role. Example:

$$\gamma' \in \omega(Win10)$$

A security-configuration guide is a set of tuples

$$\mathcal{G} = \{(\gamma_1, \mathcal{X}_1), (\gamma_2, \mathcal{X}_2), \dots\}$$

in which $\mathcal{X}_i$ denotes the set of secure values of configuration decision γ_i. Thus,

$$\forall (\gamma_i, \mathcal{X}_i) \in \mathcal{G} : \mathcal{X}_i \subset \beta(\gamma_i)$$

Example

$$(\gamma', \{Enabled\}) \in CIS\_Win10$$

We assume that there is a predicate

$$f : \Gamma \to \{True, False\} \text{ with } f(\gamma) = True \Leftrightarrow \gamma \text{ is security-relevant.}$$

Furthermore, we assume that there is a second predicate

$$g : \gamma \times \beta(\gamma) \to \{True, False\} \text{ with } g(\gamma, x) = True \Leftrightarrow x \text{ is a secure value for } \gamma.$$

Fakultät für Informatik
Lehrstuhl 4
Software & Systems Engineering
Prof. Dr. Alexander Pretschner

Boltzmannstraße 3
85748 Garching bei München

Tel: +49 (89) 289 - 17314
https://www4.in.tum.de

The research question is now: Can we model a classifier f' that approximates f? As our ground truth, we hold the CIS and IASE guides. Thus, we assume that

$$\forall\,(\gamma, \_) \in \mathcal{G} : f(\gamma) = True$$

and

$$\forall \gamma \in \omega(\theta) : (\neg\,(\exists\,(\gamma_i, \_) \in \mathcal{G} : \gamma_i = \gamma)) \Leftrightarrow f(\gamma) = False$$

In the same way, we want to model a classifier g' that approximates g.

$$\forall\,(\gamma, \mathcal{X}) : \forall x \in \mathcal{X} : g\,(\gamma, x) = True$$

and

$$\forall \gamma \in \omega(\theta) : \forall x \in \beta\,(\gamma) : (\neg\,(\exists\,(\gamma_i, \mathcal{X}) \in \mathcal{G} : \gamma_i = \gamma \wedge x \in \mathcal{X})) \Leftrightarrow g(\gamma, x) = False$$

Working Plan

1. Literature review: **3 weeks** Especially the literature in the topic of natural language processing and more specific of sentiment analysis, topic recognition, or natural language understanding might be interesting. Furthermore, literature on classification algorithms, especially if they can classify natural language, might be interesting.
2. Implementation. **6 weeks**
   - Make familiar with the configuration description format and the guides from CIS and IASE
   - Determine set of possible approaches to implement the mapping function
3. Evaluation. **3 weeks**
4. Write thesis. **4 weeks**

References

[1] Center for Internet Security (CIS). `https://www.cisecurity.org/`. Accessed: 2019-01-29.

[2] Information Assurance Support Environment (IASE): Scap content. `https://iase.disa.mil/stigs/scap/Pages/index.aspx`. Accessed: 2019-04-03.

Fakultät für Informatik
Lehrstuhl 4
Software & Systems Engineering
Prof. Dr. Alexander Pretschner

Boltzmannstraße 3
85748 Garching bei München

Tel: +49 (89) 289 - 17314
https://www4.in.tum.de