Sichere Konfigurationshärtung laufender Systeme

Patrick Stöckle
patrick.stoeckle@tum.de
Lst. f. Software und Systems Engineering
Technische Universität München (TUM)

Michael Sammereier
michael.sammereier@tum.de
Lst. f. Software und Systems Engineering
Technische Universität München (TUM)

Bernd Grobauer
bernd.grobauer@siemens.com
T CST
Siemens AG

Alexander Pretschner
alexander.pretschner@tum.de
Lst. f. Software und Systems Engineering
Technische Universität München (TUM)

Hamburg, 09.02.2023
30. DFN-Konferenz „Sicherheit in vernetzten Systemen”

Hintergrund

Grundproblem

Standardeinstellungen von Software sind nicht sicher

- Funktionen und Komfort meistens wichtiger als Sicherheit
- Interessenskonflikte
  - *Hersteller will wissen, was wir tun, um Produkte zu verbessern ⇒ Problematisch*

Lösung

Wir müssen Geräte härten!

Z.B. mit Richtlinien von

- Center for Internet Security (CIS)
- Defense Information Systems Agency (DISA)
- Bundesamt für Sicherheit in der Informationstechnik (BSI)

Bisheriger Ansatz

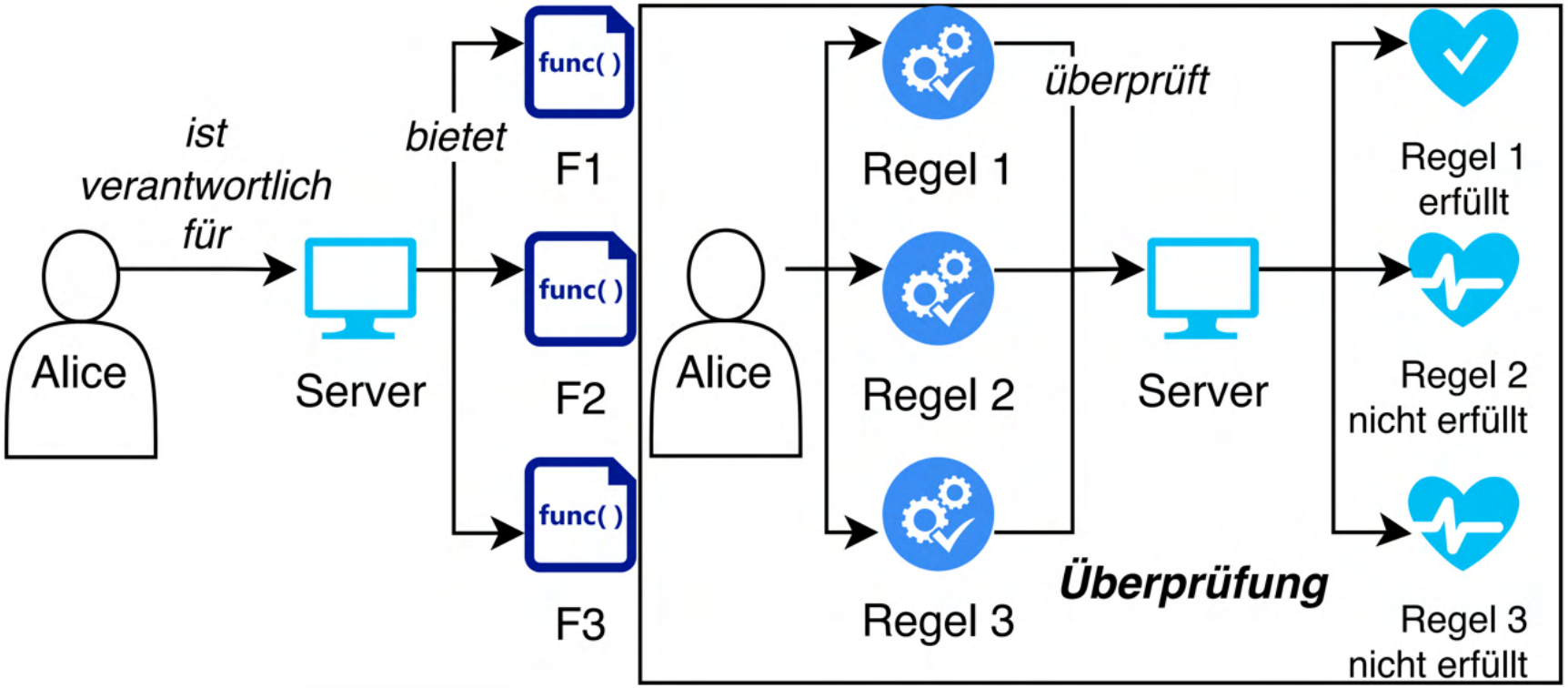
Siemens-Ansatz

Patrick Stöckle | patrick.stoeckle@tum.de | Technische Universität München (TUM)
Michael Sammereier | michael.sammereier@tum.de | Technische Universität München (TUM)

Siemens-Ansatz

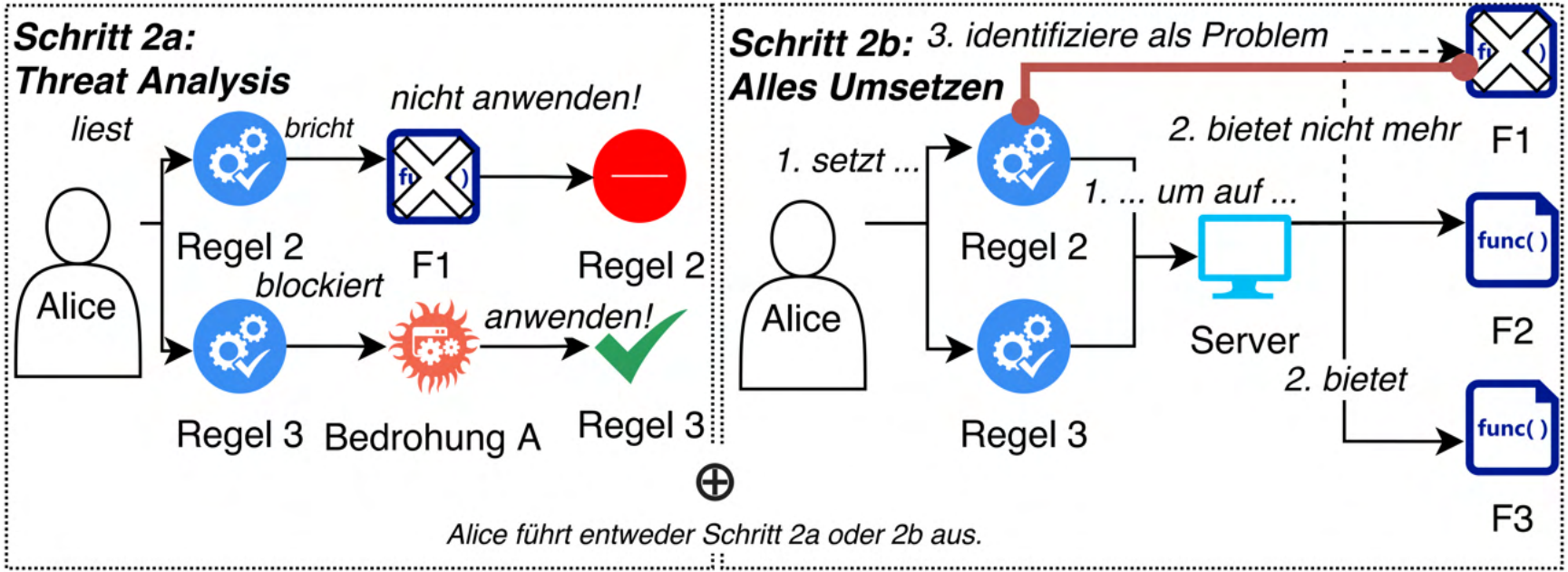
Patrick Stöckle | patrick.stoeckle@tum.de | Technische Universität München (TUM)
Michael Sammereier | michael.sammereier@tum.de | Technische Universität München (TUM)

Szenario



Patrick Stöckle | patrick.stoeckle@tum.de | Technische Universität München (TUM)
Michael Sammereier | michael.sammereier@tum.de | Technische Universität München (TUM)

Szenario



Patrick Stöckle | patrick.stoeckle@tum.de | Technische Universität München (TUM)
Michael Sammereier | michael.sammereier@tum.de | Technische Universität München (TUM)

Problem

Beide Ansätze sind zeitaufwändig und damit teuer

Threat Analysis braucht Sicherheitsexperten

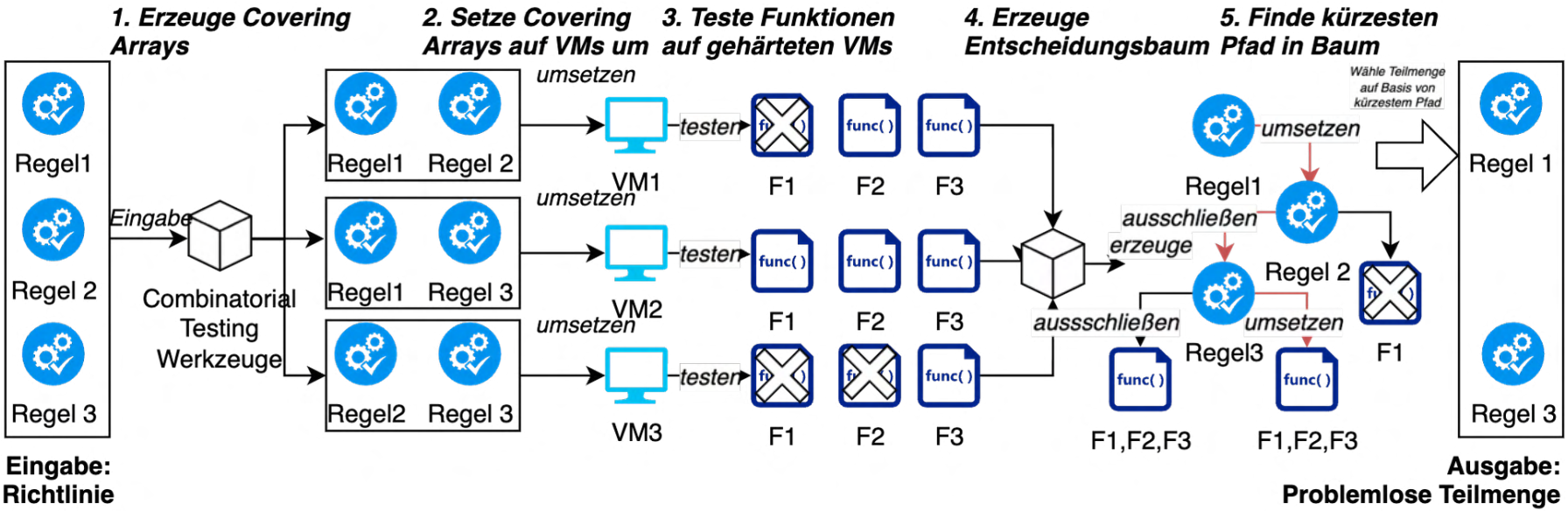
Alice weiß normal nicht, wie viele Regeln sie ausschließen muss und wie die Regeln zusammenwirken

⇒ Alice wird nur Regeln implementieren, bei denen sie sicher ist, dass die Regeln keine Funktion brechen

… oder generell auf die Härtung laufender Systeme verzichten, solange sie nicht dazu gezwungen ist

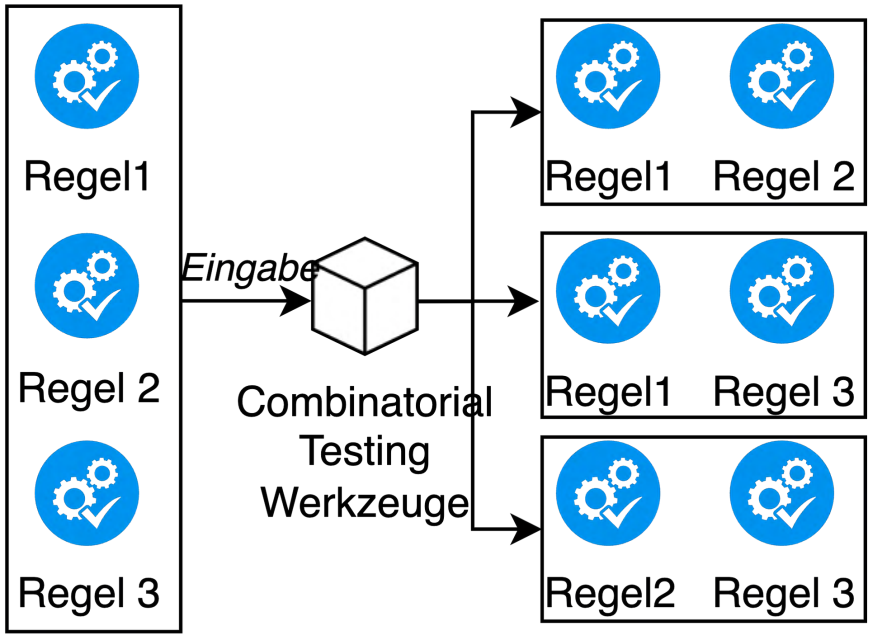
Patrick Stöckle | patrick.stoeckle@tum.de | Technische Universität München (TUM)
Michael Sammereier | michael.sammereier@tum.de | Technische Universität München (TUM)

Lösung



Sichere Konfigurationshärtung laufender Systeme
Patrick Stöckle | patrick.stoeckle@tum.de | Technische Universität München (TUM)
Michael Sammereier | michael.sammereier@tum.de | Technische Universität München (TUM)

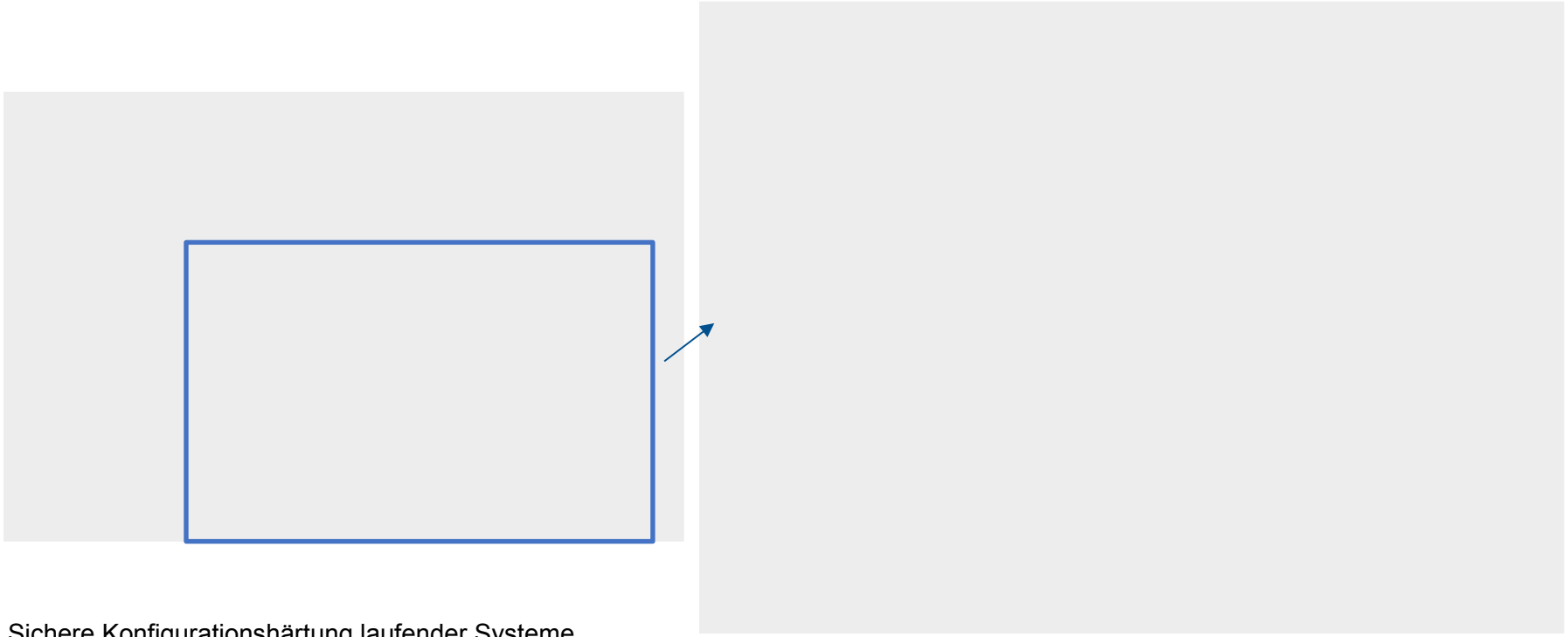
Erzeuge Covering Arrays



Eingabe:
Richtlinie

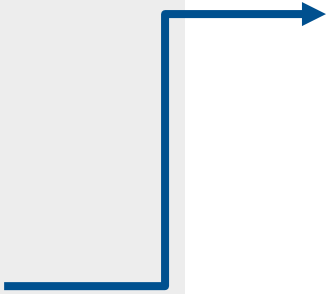
Patrick Stöckle | patrick.stoeckle@tum.de | Technische Universität München (TUM)
Michael Sammereier | michael.sammereier@tum.de | Technische Universität München (TUM)

Regel in Scapolite: Beispiel



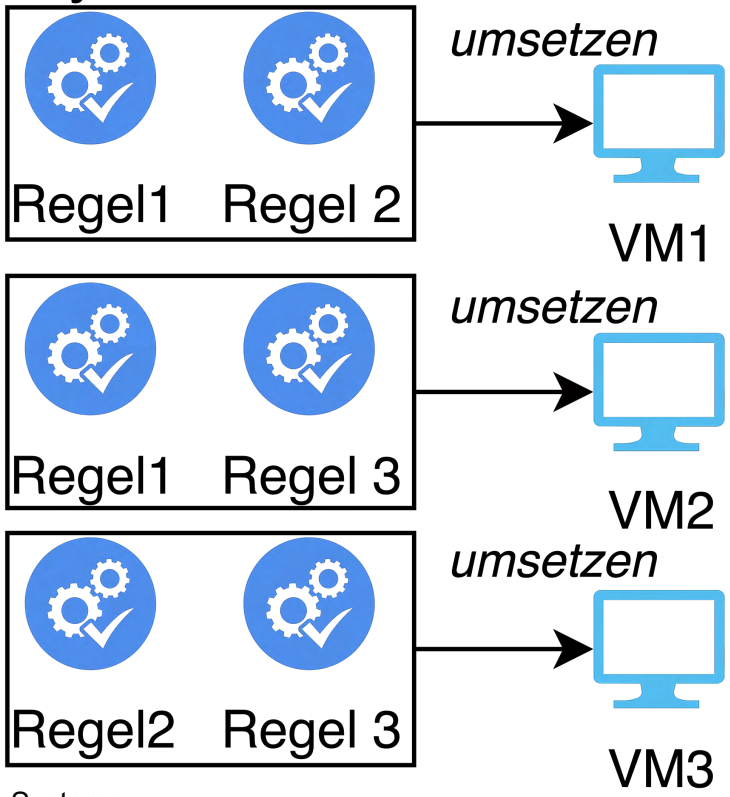
Patrick Stöckle | patrick.stoeckle@tum.de | Technische Universität München (TUM)
Michael Sammereier | michael.sammereier@tum.de | Technische Universität München (TUM)

Erzeuge Covering Arrays



```
# Degree of interaction coverage: 2
# Number of parameters: 517
BL696-0031,BL696-0249,BL696-0452,BL696-0362,BL696-
0988,BL696-0716,…
true,true,true,true,false,false,...
true,true,false,false,true,true,...
false,false,true,true,true,true,...
false,false,false,false,false,false,...
...
```

Setze Covering Arrays auf VMs um



Patrick Stöckle | patrick.stoeckle@tum.de | Technische Universität München (TUM)
Michael Sammereier | michael.sammereier@tum.de | Technische Universität München (TUM)

Setze Covering Arrays auf VMs um

1. Zunächst bereiten wir ein Image mit unserer Software und allen benötigten Abhängigkeiten vor
2. Wir bereiten für jede VM-Instanz ein Verzeichnis mit der auszuführenden Software, den Tests, der Richtlinie und den zu testenden Covering Arrays vor
3. Wir starten die Instanzen entweder parallel oder sequentiell
4. Wir führen alle Tests auf einer beliebigen Instanz aus, um zu sehen, ob die Funktionalität in der Standardkonfiguration funktioniert
5. Wir setzen automatisiert alle Regeln auf einer Instanz um
6. Wenn keine Tests fehlschlagen, können wir alle Regeln sicher anwenden und den Prozess an diesem Punkt beenden
7. Wir kehren alle Regeln um und führen die Tests erneut aus. Wenn einige Tests immer noch fehlschlagen, gibt es ein Problem mit dem Umkehrmechanismus
8. Wir nehmen das erste ungetestete Covering Array und setzen alle Regeln aus diesem auf einer Instanz um

Patrick Stöckle | patrick.stoeckle@tum.de | Technische Universität München (TUM)
Michael Sammereier | michael.sammereier@tum.de | Technische Universität München (TUM)

Setze Covering Arrays auf VMs um

SIEMENS TUM

Vagrantfile

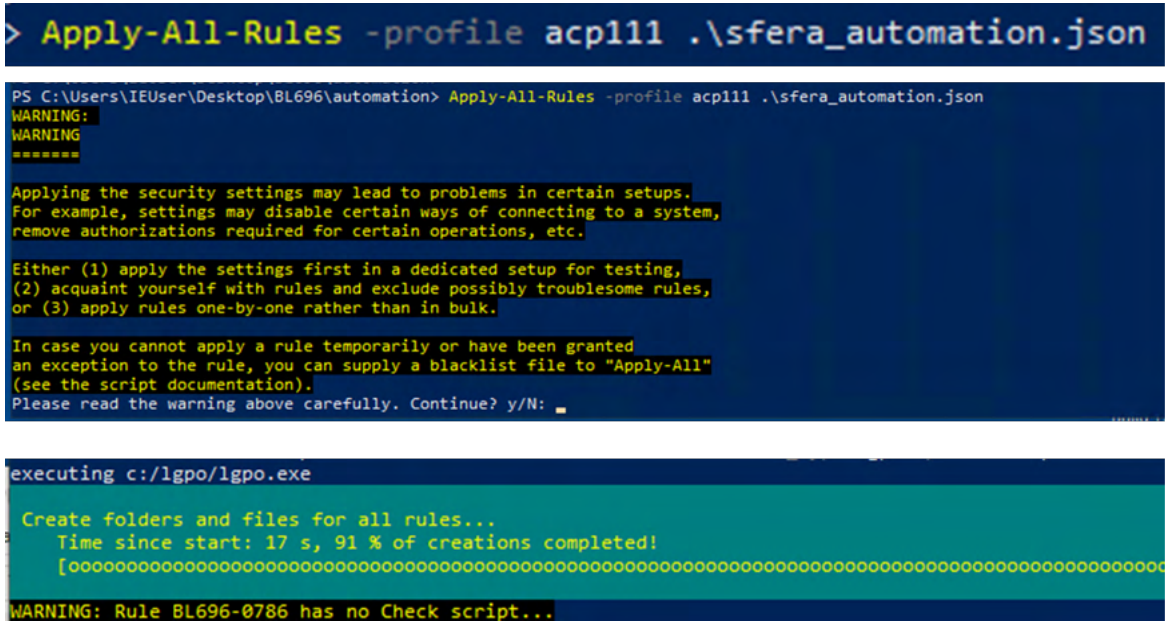
```
Vagrant.configure("2") do |config|
  config.vm.box = "WinServer2016"
  config.vm.communicator = "winrm"
  # Authentication
  config.winrm.username = "Administrator"
  config.winrm.password = "K3aPXxr7EroHfU5WY2y5"
  config.vm.provider "virtualbox" do |vb|
    vb.memory = "4096"
    vb.cpus = "4"
  end
 # Automatically execute the following script after startup
  config.vm.provision "shell", path: "setup.ps1"
end
```

setup.ps1

```
# ========== START SETTING UP THE VM ==========
# Set-ExecutionPolicy in order to allow the execution of the scripts
Set-ExecutionPolicy -ExecutionPolicy Bypass
# Move all files to the VMs storage (e.g. Desktop).
# This is necessary because the files cannot be executed from the
shared directory.
mkdir $dir
cp -r $vbox_dir\* $dir
# Change to the local directory
cd $dir
# Load the test-configuration
$test_config = Get-Content .\$tests_dir\$test_config_filename | Out-
String | ConvertFrom-Json
# Use sfera directory defined in the test_configuration
$sfera_dir = $($test_config.sfera_directory)
# Load sfera_automation.json
$sfera_json = "$sfera_dir\sfera_automation.json"
$sfera_automation_json = Get-Content $sfera_json | Out-String |
ConvertFrom-Json
# Source the sfera_automation scripts
. .\$sfera_dir\sfera_automation.ps1
# Download LGPO
Download-Lgpo
# ========== SETUP COMPLETE ==========
```

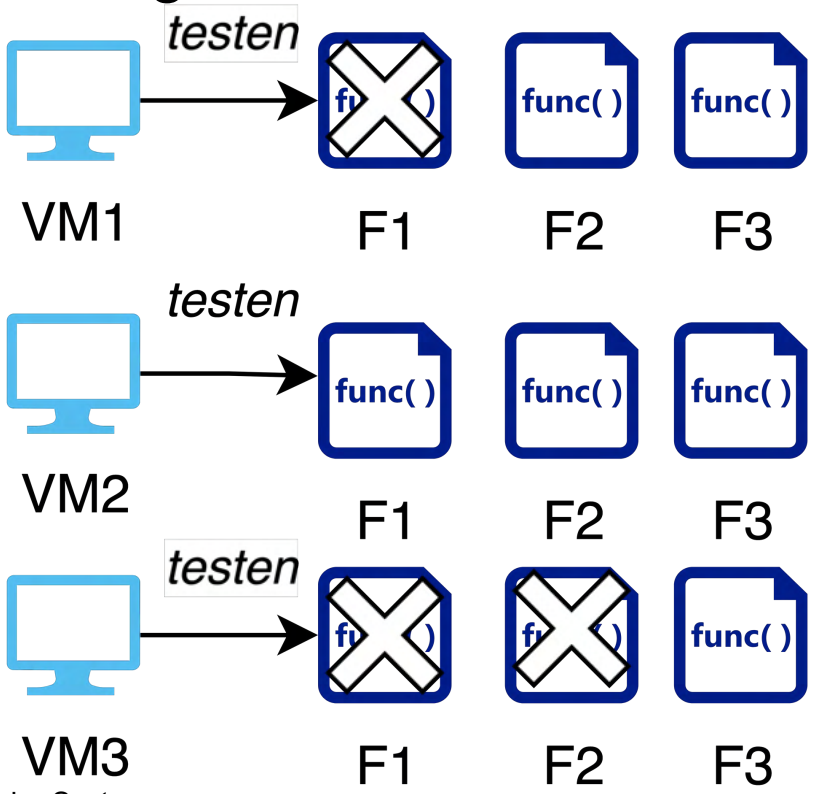

Patrick Stöckle | patrick.stoeckle@tum.de | Technische Universität München (TUM)
Michael Sammereier | michael.sammereier@tum.de | Technische Universität München (TUM)

Setze Covering Arrays auf VMs um



Patrick Stöckle | patrick.stoeckle@tum.de | Technische Universität München (TUM)
Michael Sammereier | michael.sammereier@tum.de | Technische Universität München (TUM)

Teste Funktionen auf gehärteten VMs



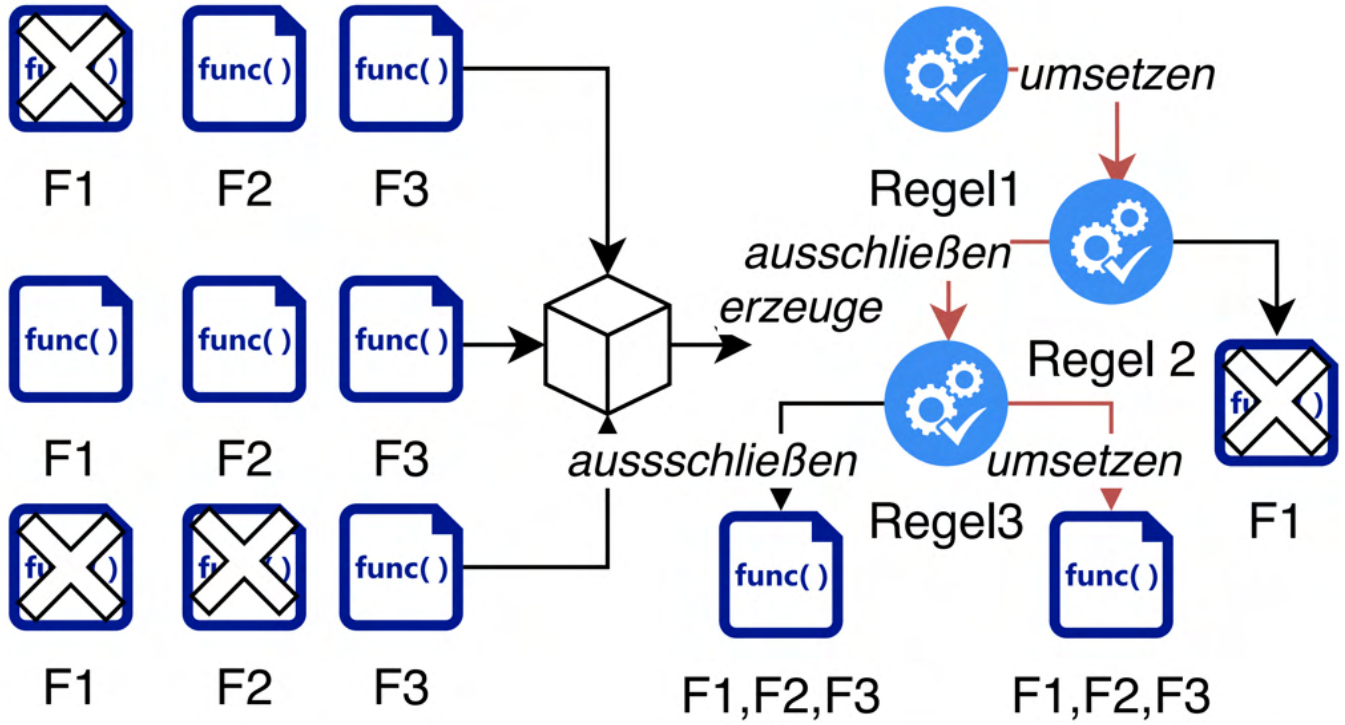
Patrick Stöckle | patrick.stoeckle@tum.de | Technische Universität München (TUM)
Michael Sammereier | michael.sammereier@tum.de | Technische Universität München (TUM)

Teste Funktionen auf gehärteten VMs

9. Wir führen die Tests aus und speichern in einer JSON-Datei, ob es fehlgeschlagene Tests gab
10. Wir führen einen Soft- oder Hard-Reset durch, um Reihenfolgeneffekte zu vermeiden. Danach gehen wir zurück zu Schritt 7, bis wir alle Covering Arrays angewendet haben
11. Nachdem wir alle Covering Arrays angewendet und getestet haben, sammeln wir alle Ergebnisse aus den verschiedenen Instanzen und kombinieren sie zu einer einzigen JSON-Datei
12. Wir löschen die Instanzen

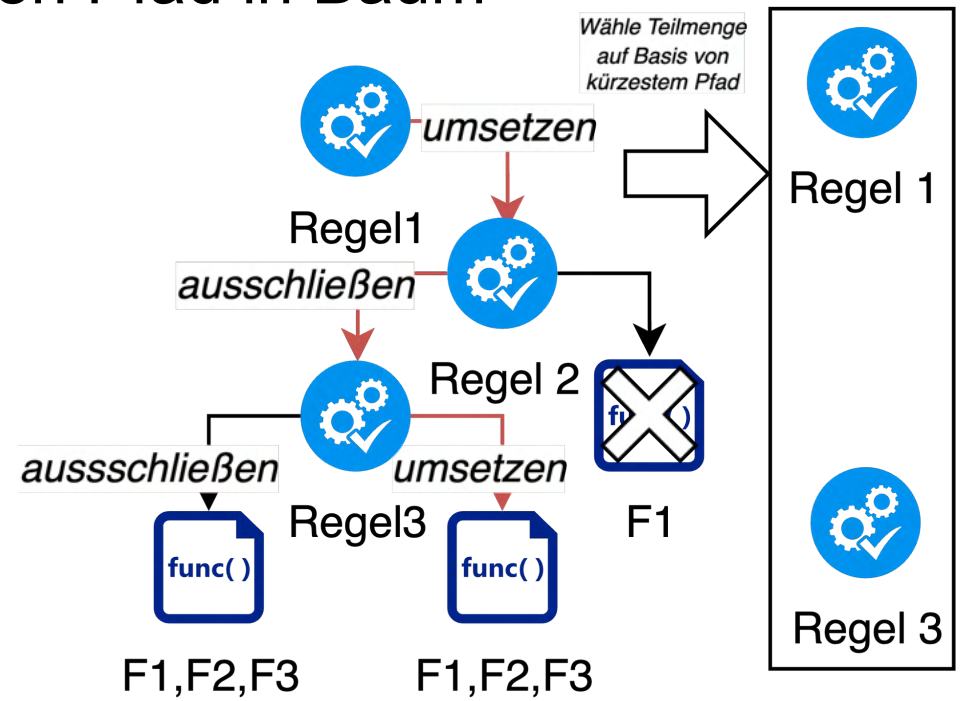
Patrick Stöckle | patrick.stoeckle@tum.de | Technische Universität München (TUM)
Michael Sammereier | michael.sammereier@tum.de | Technische Universität München (TUM)

Erzeuge Entscheidungsbaum



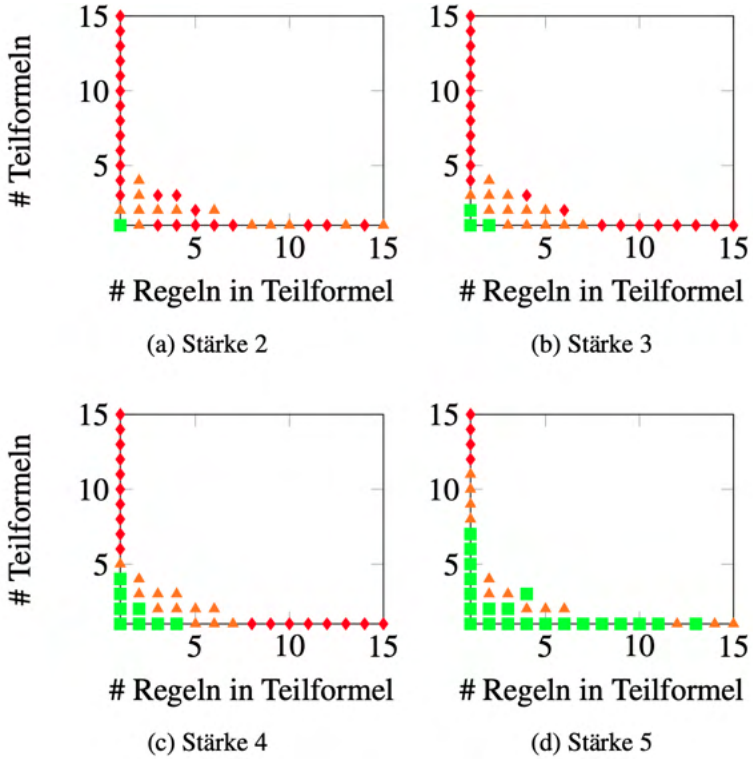
Patrick Stöckle | patrick.stoeckle@tum.de | Technische Universität München (TUM)
Michael Sammereier | michael.sammereier@tum.de | Technische Universität München (TUM)

Finde kürzesten Pfad in Baum



Patrick Stöckle | patrick.stoeckle@tum.de | Technische Universität München (TUM)
Michael Sammereier | michael.sammereier@tum.de | Technische Universität München (TUM)

Ergebnisse



Patrick Stöckle | patrick.stoeckle@tum.de | Technische Universität München (TUM)
Michael Sammereier | michael.sammereier@tum.de | Technische Universität München (TUM)

Ergebnisse

Korrektheit

- Der Ansatz findet insgesamt für 77 % unserer Beispiele eine optimale Lösung
- Bei den *realistischen* Beispielen findet der Ansatz in fast 100 % Fällen eine optimale Lösung

Zeitaufwand

- Zeit für Erzeugung der Covering Arrays hängt von Stärke ab ⇒ Unter 1s für Stärke 2 und 14 Tage für Stärke 5
- Zeitaufwand für Erzeugung nur bei Covering Arrays bis zur Stärke 4 vertretbar
- Zeitaufwand für die Durchführung der Tests bei Stärke 4 über 12h ⇒ Möglich, aber eher für Integrationstests geeignet

Patrick Stöckle | patrick.stoeckle@tum.de | Technische Universität München (TUM)
Michael Sammereier | michael.sammereier@tum.de | Technische Universität München (TUM)

Einschränkungen

Tests

- Wir brauchen automatisierte Tests ⇒ gerade bei komplexen Systeme oder grafischen Oberflächen wird häufig noch manuell getestet
- Wir brauchen *gute* Tests. Wenn die Tests durchlaufen, es aber im produktiven Betrieb zu Problemen kommt, müssen wir im Zweifel alle Regeln zurücknehmen
- Wir brauchen schnelle Tests, da wir die Tests für jedes Covering Array einmal ausführen müssen ⇒ gerade komplexe Systeme brauchen viele und komplexe Tests, die dauern

Setup

- Wir brauchen Systeme, die wir automatisiert aufsetzen, installieren und testen können ⇒ gerade bei komplexen und heterogenen Systemen schwierig bis unmöglich

Fehlende Daten über Konfigurationsprobleme

- Da wenige Organisationen die Konfiguration ihrer Systeme härten, haben wir keine empirischen Informationen über problematische Kombinationen und müssen stattdessen Daten aus dem SW Test heranziehen ⇒ Übertragbarkeit fraglich

Fazit

Mehr Tests!

Wir brauchen mehr Tests und automatisierte Tests, um Härtungsmaßnahmen auf laufenden Systemen ohne Nebenwirkungen umsetzen zu können!

A/B-Tests?

Solange wir diese automatisierten Tests nicht haben, könnten die Covering Arrays auch im Rahmen von A/B-Tests getestet werden: Jeweils ein Covering Array wird auf einer Teilmenge angewendet. Sobald ein Problem auftritt, werden die Regeln direkt zurückgesetzt und das Array als fehlerhaft markiert. Wenn alle Arrays getestet wurden, können wir die unproblematische Teilmenge bestimmen

Härtung von Anfang an!

Die Härtung von laufenden Systemen ist und bleibt aufwändig. Daher sollten die Systeme möglichst direkt gehärtet werden, sodass man Probleme direkt analysieren und beheben kann

Patrick Stöckle | patrick.stoeckle@tum.de | Technische Universität München (TUM)
Michael Sammereier | michael.sammereier@tum.de | Technische Universität München (TUM)

Sichere Konfigurationshärtung laufender Systeme

Patrick Stöckle
patrick.stoeckle@tum.de
Lst. f. Software und Systems Engineering
Technische Universität München (TUM)

patrick-stoeckle

p_stoeckle

Michael Sammereier
michael.sammereier@tum.de
Lst. f. Software und Systems Engineering
Technische Universität München (TUM)

michael-sammereier

Hamburg, 09.02.2023
30. DFN-Konferenz „Sicherheit in vernetzten Systemen”