УДК 004.056
DOI: 10.33764/2618-981X-2019-6-2-92-98

РАЗРАБОТКА АЛГОРИТМА ОРГАНИЗАЦИИ ИНФРАСТРУКТУРЫ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ С ПРИМЕНЕНИЕМ СИСТЕМЫ УПРАВЛЕНИЯ КОНФИГУРАЦИЕЙ SALTSTACK ПРИ ПЕРЕДАЧЕ ДАННЫХ ПО ВОЛОКОННО-ОПТИЧЕСКОЙ ЛИНИИ СВЯЗИ

Сергей Андреевич Корягин
Сибирский государственный университет геосистем и технологий, 630108, Россия, г. Новосибирск, ул. Плахотного, 10, магистрант, тел. (999)452-12-45, e-mail: skoryagin96@gmail.com

Евгений Владимирович Грицкевич
Сибирский государственный университет геосистем и технологий, 630108, Россия, г. Новосибирск, ул. Плахотного, 10, кандидат технических наук, доцент кафедры информационной безопасности, тел. (383)343-91-11, e-mail: gricew@mail.ru

В работе рассматривается подход к организации инфраструктуры информационной безопасности при передаче данных по волоконно-оптической линии связи с использованием системы управления конфигурацией, которая позволяет снизить трудозатраты на обслуживание, ускорить установку, настройку информационно-технических сервисов и программного обеспечения на виртуальных и физических серверах. Приводится аналитический обзор и качественное сравнение наиболее распространенных в настоящее время подобных систем с учетом характеристик, напрямую связанных с безопасным хранением информации и взаимодействием между серверами, и выбор наиболее подходящей системы для данной работы. Приведены теоретическое описание алгоритма построения инфраструктуры информационной безопасности и примеры программной реализации с использованием системы управления конфигурацией SaltStack. Разработано программное обеспечение для обработки отчетов сканера уязвимостей.

Ключевые слова: инфраструктура, система управления конфигурацией, состояние системы, безопасность, программный комплекс, оптоволокно.

DEVELOPMENT OF AN ALGORITHM FOR INFORMATION SECURITY INFRASTRUCTURE ORGANIZATION USING CONFIGURATION MANAGEMENT SYSTEM SALTSTACK WHEN TRANSMITTING DATA VIA FIBER-OPTIC COMMUNICATION LINES

Sergey A. Koryagin
Siberian State University of Geosystems and Technologies, 10, Plakhotnogo St., Novosibirsk, 630108, Russia, Graduate, phone: (999)452-12-45, e-mail: skoryagin96@gmail.com

Evgenij V. Gritskevich
Siberian State University of Geosystems and Technologies, 10, Plakhotnogo St., Novosibirsk, 630108, Russia, Ph. D., Associate Professor, Department of Information Security, phone: (383)343-91-11, e-mail: gricew@mail.ru

In the article an approach to organization of information security infrastructure using the configuration management system is considered, which allows reducing of service labor costs, acceleration of installation, tuning of information and technical services and the software on virtual and physical servers. An analytical review and qualitative comparison of the most common similar cur-

rently existing systems are presented, taking into account the characteristics directly related to safe storage of information and interaction between the servers and the choice of the most suitable system for this work. A theoretical description of the security infrastructure building algorithm and examples of software implementation using the SaltStack configuration management system are given. Software for processing of vulnerability scanner reports is developed.

Key words: infrastructure, configuration management system, system state, security, software package, optical fiber.

Введение

Естественное расширение инфраструктуры любой информационной системы по мере ее развития в процессе непрерывного функционирования создает проблему увеличения трудовых и материально-финансовых затрат не только на обслуживание системы, внедрение новых и вывод из эксплуатации отработавших ресурсов, но и параллельно инициирует модификацию и адаптацию к новым условиям работы ресурсов информационной безопасности (ИБ) с точки зрения соответствия последних требованиям поддержания системы защиты информации на должном уровне, определяемом соответствующими нормативами. Для более быстрого и безопасного размещения и настройки модифицируемых ресурсов ИБ целесообразно использовать систему управления конфигурацией.

Целью работы является разработка алгоритма организации инфраструктуры информационной безопасности при передаче данных по волоконно-оптическим линиям связи с использованием системы управления конфигурацией SaltStack.

Для достижения цели были поставлены следующие задачи:

– анализ систем управления конфигурацией;

– анализ механизмов системы управления конфигурацией SaltStack;

– разработка алгоритма развертывания и администрирования инфраструктуры безопасности с использованием системы управления конфигурацией SaltStack;

– программная реализация библиотеки state-файлов для алгоритма;

– разработка программного обеспечения для обработки отчетов сканера уязвимостей.

Методы

Система управления конфигурацией – это программный комплекс для централизованного управления множеством разрозненных операционных систем. При использовании системы управления конфигурацией наиболее важным представляется соблюдение принципа минимального выполнения операций на конечных узлах. Основное управление осуществляется с помощью мастерхоста (master-host, masterhost, master-хост). При этом вся инфраструктура, ее внутреннее взаимодействие, настройки программного обеспечения могут быть описаны в нескольких файлах [1–3].

В настоящее время наиболее популярными системами управления конфигурацией являются: Ansible, Chef, Puppet и SaltStack.

К одному из преимуществ системы Ansible относится использование в качестве основного инструмента программной реализации популярного языка Python. У этой системы низкий порог вхождения. Система использует безагентную структуру. Недостатками Ansible являются плохая поддержка операционной системы Windows, более трудозатратное добавление новых подчиненных операционных систем и тестирование написанных состояний, а также небольшое сообщество разработчиков [4, 5].

Преимуществом систем Chef и Puppet является наличие большой базы готовых модулей и рецептов для настройки подчиненных операционных систем [6, 7]. Среди недостатков стоит выделить использование для программной реализации недостаточно популярного языка исполнения Ruby [6, 8], высокий порог вхождения для начала работы и необходимость оптимизации готовых модулей, получаемых из системных баз.

Наиболее перспективным инструментом управления конфигурацией представляется система SaltStack, которая обладает рядом значительных преимуществ перед остальными системами – язык разработки Python, легкая масштабируемость, работа в режиме masterless (то есть при отсутствии master-хоста), а также в режиме syndic (подчиненная система является master-хостом для других подчиненных систем, за счет чего выстраивается иерархическая структура). Для общения с подчиненными системами используется протокол ZeroMQ. Среди недостатков стоит выделить высокий порог вхождения.

Из приведенного обзора можно сделать вывод, что для управления инфраструктурой безопасности наиболее подходящей является система SaltStack благодаря возможности работы по отдельному протоколу ZeroMQ, наличию внутреннего аудита и хорошей масштабируемости.

Поэтому представляется целесообразным более подробно рассмотреть основные понятия и принципы работы данного программного комплекса.

С точки зрения системы SaltStack все машины делятся на два типа – salt-master (машины, с которых ведется управление) и salt-minion (управляемые машины). Если по каким-то причинам на подчиненной системе нельзя установить salt-minion, команды для управления могут исполняться по ssh-протоколу. При работе в syndic-режиме salt-master может являться salt-minion, а при работе в masterless-режиме salt-minion управляет сам собой [9].

В рассматриваемой системе существует понятие salt-state (SLS) – конфигурационные файлы, написанные на языке YAML. Понятие Templates обозначает шаблоны, содержащие обобщенные описания формул и позволяющие применять типовые действия для различных операционных систем. Термин Grains относится к информации о системе, генерируемой при подключении salt-minion. Понятие Pillars определяет защищенные хранилища информации, описывающие параметры системы, которые применяются при выполнении SLS-файлов [10].

Salt-master аутентифицирует salt-minion, используя открытый ключ шифрования и аутентификацию. Когда salt-minion впервые запускается, он генерирует криптографический ключ и пытается подключиться к salt-master. Чтобы salt-minion смог принимать команды от salt-master, он должен быть принят последним.

Машины типа salt-minion подключаются и общаются с машинами типа salt-master по портам 4505\4506 TCP [9, 11] (рис. 1).

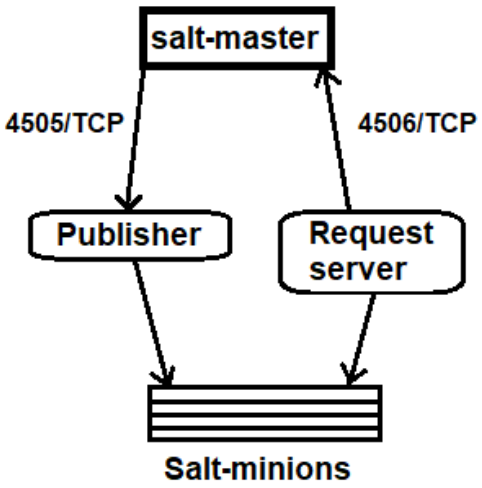

Рис. 1. Схема взаимодействия управляющего salt-master
с подчиненными salt-minions

Master аутентифицирует minion, используя открытый ключ шифрования и аутентификацию. Для удаленного выполнения команд не требуется установки salt-minion.

Алгоритм развертывания и администрирования включает в себя следующие действия:

1) установка salt-master на master-хост, настройка политик доступа и подключение библиотеки state-файлов с помощью системы контроля версий;

2) установка операционных систем на целевые виртуальные машины и физические сервера;

3) установка salt-minion на целевые хосты с использованием state-файлов;

4) настройка операционных систем, установка и настройка программного обеспечения на salt-minion;

5) автоматизация обновлений безопасности с использованием системы контроля версий, системы управления конфигурацией и разработанного программного обеспечения на Python.

Библиотека state-файлов представляет собой коллекцию yaml-файлов для автоматической установки и настройки программного обеспечения на rhel и debian-based операционных системах, которую рекомендуется разместить

в системе контроля версий (например, Gitlab или Bitbucket) для версионного модифицирования state-файлов, а также централизованного распространения на master-хосты. Также рекомендуется использовать систему контроля версий для размещения файлов конфигурации, написанных с использованием языка шаблонов Jinja, позволяющего при размещении на хостах подставлять значения для разных операционных систем.

Установку операционных систем рекомендуется производить с использованием файлов автоматической установки (файлы «kickstart.cfg» для rhel-based и «preceed.cfg» для debian-based операционных систем).

При установке salt-master необходимо настроить правила доступа с помощью опции «access control list», которые ограничивают выполнение модулей SaltStack для пользователей на master-хосте.

Установка salt-minion производится через применение соответствующего state-файла (рис. 2) на roster-файл со списком машин.

```
salt-minion:
  pkg.installed:
    - order: 4
    - pkgs:
       - salt-minion
       - virt-what
       - python-augeas
  service.running:
    - name: salt-minion
    - order: 4
    - enable: True
    - require:
      - pkg: salt-minion
  cmd.wait:
    - name: echo service salt-minion restart | at now + 3 minute
    - order: last
    - watch:
      - file: /etc/salt/minion

/etc/salt/minion:
  file:
    - managed
    - order: 4
    - source: salt://{{ tpldir }}/files/salt-minion.cfg
    - user: root
    - group: root
    - mode: '0640'
    - makedirs: True
    - template: jinja
    - require:
      - pkg: salt-minion
```

Рис. 2. State-файл для установки и настройки salt-minion

State-файлы программного обеспечения включают в себя файлы установки и настройки для таких пакетов, как selinux (система принудительного контроля доступа), sshd, clamav (антивирус), zabbix-agent (агент мониторинга) и т. д.

Также было разработано программное обеспечение на языке Python для реализации автоматической обработки запросов обновления безопасности сканера уязвимостей, настроенного на отправку отчетов в xml-формате.

Программа позволяет обработать xml-отчет сканера и выполняет автоматическое обновление уязвимых компонентов на контролируемых операционных системах. State-файлы программного обеспечения включают в себя файлы установки и настройки для таких пакетов, как selinux (система принудительного контроля доступа), sshd, clamav (антивирус), zabbix-agent (агент мониторинга) и т. д.

Также было разработано программное обеспечение на языке Python для реализации автоматической обработки запросов обновления безопасности сканера уязвимостей, настроенного на отправку отчетов в xml-формате.

Программа позволяет обработать xml-отчет сканера и выполняет автоматическое обновление уязвимых компонентов на контролируемых операционных системах. Часть кода программы представлена ниже на рис. 3.

```
all_hosts = raw_xml.findall('//{http://www.ptsecurity.ru/reports}host')

for host in all_hosts:
    pkgs = ''
    hostname = host.attrib["fqdn"]
    soft = host.findall('.//{http://www.ptsecurity.ru/reports}soft')
    for pkg in soft:
        for elem in pkg.getchildren():
            if 'vulners' in elem.tag:
                pkgs += pkg.find(".//{http://www.ptsecurity.ru/reports}name").text
    if len(pkgs) > 0:
        roster.addElement(hostname, pkgs)

with open('gitlab-ci.yml', 'w') as output:
    yaml.dump(roster, output, default_flow_style=False)
```

Рис. 3. Часть кода программы обработки отчетов сканера уязвимостей

Результаты

Преимущества системы управления конфигурацией SaltStack за счет использования протокола ZeroMQ, наличия внутреннего аудита и защищенных хранилищ информации обеспечивают надежность соединения и уверенность в

целостности системы. Разработана библиотека state-файлов, позволяющая централизованно разворачивать и администрировать инфраструктуру безопасности. Благодаря языку шаблонов Jinja допускается использование одного state-файла на нескольких операционных системах.

Разработано программное обеспечение, позволяющее автоматизировать процесс поддержания безопасной пакетной базы на целевых хостах.

Заключение

Применяя сочетание конфигурационных state-файлов и pillars-хранилищ, описывающих параметры системы, возможно конфигурировать подчиненную систему и запущенное на ней программное обеспечение за несколько команд. Применение данного программного комплекса позволит сократить время развертывания и настройки системы безопасности. Разработанную библиотеку state-файлов можно использовать для развертывания и настройки системы безопасности новых информационных систем. Разработанное программное обеспечение, в совокупности со сканером уязвимостей, позволит поддерживать безопасную пакетную базу информационной системы.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Система управления конфигурацией [Электронный ресурс]. – Режим доступа: http://xgu.ru/wiki/Система_управления_конфигурацией.

2. Безопасность систем с открытым исходным кодом [Электронный ресурс]. – Режим доступа: http://citforum.ru/security/articles/open_source/.

3. Обзор: Puppet, Chef, Ansible, SaltStack [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/post/211306/.

4. Jesse Keating. Mastering Ansible. – Packt Publishing, 1 edition, 2015. – 236 с.

5. Lorin Hochstein, Rene Moser. Ansible: Up and Running – O`Relly Media, 2 edition, 2017. – 430 с.

6. Earl Waud. Mastering Chef Provisioning. – Packt Publishing, 1 edition, 2016. – 262 с.

7. Thomas Uphill, John Arundel. Puppet Cookbook. – Packt Publishing, 3 edition, 2015. – 338 с.

8. Martin Alfke, Felix Frank. Puppet 5 Essentials. – Packt Publishing, 3 edition, 2017. – 262 с.

9. Colton Myers. Learning SaltStack. – Packt Publishing, 2015. – 158 с.

10. Christer Edwards. Into The Salt Mine Documentation [Электронный ресурс]. – Режим доступа: https://intothesaltmine.readthedocs.io/en/latest/.

11. SaltStack Tutorial [Электронный ресурс]. – Tutorials Point. – Режим доступа: https://www.tutorialspoint.com/saltstack/saltstack_tutorial.pdf.