
iq option
Automatic
Linux Patch
Management
Ilya Dubov
18.06.2019

Whoami

iq option

Дубов Илья, руководитель команды IT security
Работаю в компании IQ Option Software
6 лет опыта в информационной безопасности

Context

iq option

Когда мы начали строить процесс, мы имели:

> 200 физических серверов и > 800 контейнеров

около 20 команд разработки

более тысячи уязвимостей, включая критические

Problems

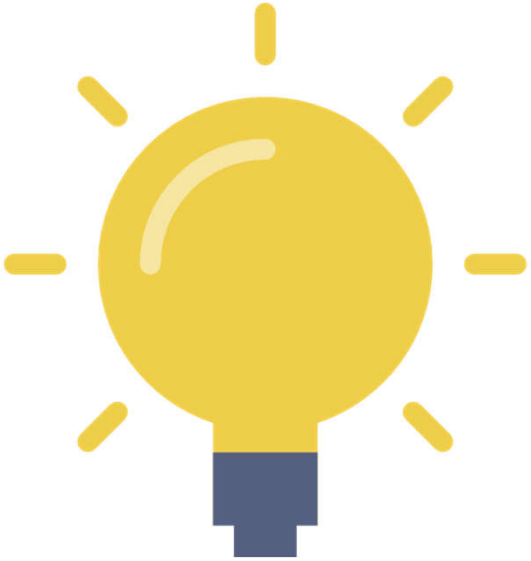
iq option

Проанализировав данные и пообщавшись с IT командами, мы выделили следующие проблемы:

- Невозможность оперативно сообщать системным администраторам о найденных уязвимостях
- Нехватка системных администраторов
- От 3-х до 4-х недель времени работы системного администратора на обработку отчетов сканера команд разработки

Idea

iq option



Разработать инструмент, который:

- Автоматизирует деятельность специалиста по безопасности в части информирования системных администраторов об уязвимостях
- Автоматизирует деятельность системного администратора в части тестирования и установки патчей

Solution Choosing

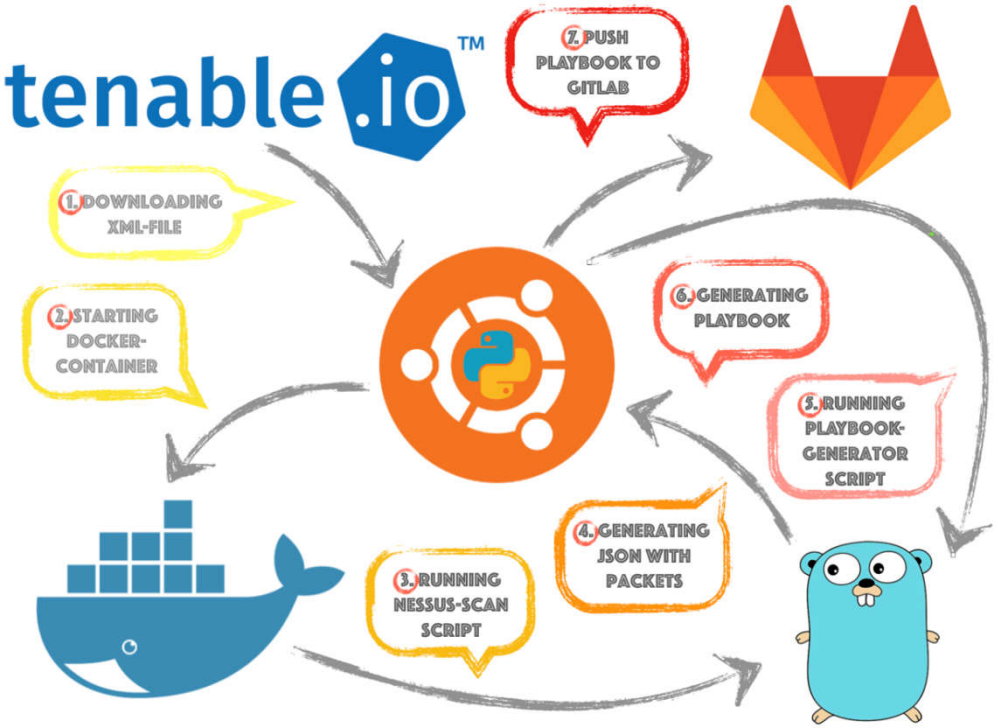
iq option

В нашем случае решение было практически безальтернативным, так как мы использовали инструменты и принципы, используемые нашими системными администраторами:

- Автоматизация - Ansible Playbooks
- Infrastructure As Code - Gitlab
- APT репозиторий - Artifactory
- Communications - Slack, Task Tracker

Final Design

iq option



1 Автоматическая выгрузка отчета из сканера безопасности.
2, 3, 4 Автоматическая обработка отчета и поиск доступных в сорслистах APTa версий программного обеспечения и библиотек.
5, 6 Автоматическая генерация Ansible Playbook, который содержит требуемые версии программного обеспечения и библиотек, для используемых версий ОС Ubuntu. Генерятся разные плейбуки для уязвимостей разного уровня критичности.
7 Коммит Ansible Playbook в репозиторий Gitlab нужного проекта с уведомлением системного администратора в Slack/Task Tracker

Tool

iq option

Для реализации решения были написаны скрипты для генерирования плейбука на Golang и автоматизации всего процесса патч менеджмента на Python.
Генератор плейбука состоит из двух скриптов.

Report processing

Первый скрипт Golang выполняет следующие действия:

1. Парсинг отчета сканера уязвимостей с целью идентификации информации о пакете и его версии.
2. Выявление версии пакета, предложенной сканером уязвимостей.
3. Поиск пакета, предлагаемого сканером уязвимостей к обновлению, в сорслистах АТРа, в том числе в Artifactory. Проверка на наличие пакета для обновления осуществляется АРТом в докер-контейнере с нужной ОС.
4. Формирование отчета в .JSON формате со списком и версиями пакетов к обновлению, а также с названием хостов, на которых нужно произвести обновление.

Если в репозитории АРТа существует та версия пакета, на которую предлагает обновиться сканер уязвимостей, ставится она, если ее уже нет, ставится последнее минорное обновление. Идентичность версий пакетов в разных окружениях, например, integration и production, обеспечивается с помощью Artifactory, который кеширует пакеты, скачиваемые с публичных репозиториев Ubuntu.

Ansible playbook building

Второй скрипт Golang - генерирует плейбук с нужной конфигурацией, ролью APT и информацией, зафиксированной в .JSON отчете

```
./
└── { playbook_name }/
    ├── ansible.cfg
    ├── { env_name_inventory }
    ├── functions/
    │   └── compare_packages.yml
    ├── group_vars/
    │   ├── all.yml
    │   └── { package_name }.yml
    ├── host_vars/
    ├── main.yml
    └── roles/
        └── requirements.yml
```

group_vars/ - содержит yaml файлы - группы, содержащие версии пакетов, на которые они обновятся впоследствии прокатки плейбука.
functions/ - содержит функцию сравнения групп, которая каждый раз подставляет при сравнении следующую. Без этой функции плейбук обновит только один последний пакет на всех серверах.
host_vars/ - содержит файлы, названные в соответствии с тем, в каком окружении находятся в нем указанные хосты - Infrastructure, Integration, Production, Development и Other.
roles/ - содержит единственную роль - APT.

Patch Management Automation

Остальная часть процесса - это обвязка Golang скриптов Python’ом, который выполняет следующие функции:

1. Выгрузка отчета в .XML формате из сканера безопасности
2. Запуск первого скрипта генератора nessus-scan в docker-контейнерах
3. Запуск второго скрипта генератора playbook-generator с необходимыми входными параметрами
4. Пуш в Gitlab созданного плейбука
5. Уведомление системного администратора в Slack / Task Tracker

Profit

iq option

- Экономия времени специалиста по информационной безопасности на подготовку отчетов для команд - с нескольких часов до нескольких минут
- Экономия времени системного администратора на тестирование и установку патчей в Production окружении - с 3-4 недель до нескольких дней
- Идентичные версии пакетов на тестовых и боевых окружениях
- Уведомление системных администраторов о новых уязвимостях

To whom it may interest in

iq option

- Вы обладатель масштабной linux-инфраструктуры
- У вас нет отдела со специалистами безопасности, который все время занимается подготовкой отчетов из сканеров уязвимостей, уведомлением системных администраторов, постановкой задач :)
- У вас нет отдела системных администраторов, который все время занимается накаткой патчей, тестированием работоспособности ОС и бизнес-приложений :)
- Вы любите эффективно использовать имеющиеся у вас ресурсы: человеческие и программные

To Do

iq option

- Сделать решение кросс-платформенным (изначально не стояла такая задача)
- Улучшить архитектуру решения
- ! Сделать единый framework

Thank you!

Facebook https://www.facebook.com/elijahdubov
Telegram @elijahduboff