www.cyberok.ru
info@cyberok.ru
+7 (495) 137-7337

Nuclei Fu

Станислав Савченко – ведущий эксперт базы знаний, CyberOK

Атаки на Nuclei

В сегодняшней статье речь пойдет о Nuclei — это мощный open-source инструмент для поиска уязвимостей (DAST), который активно развивается благодаря поддержке сообщества.

За последний год в Nuclei было несколько крупных обновлений, которые принесли множество фич, таких как кодовая вставка, использование javascript в шаблонах, управление запросами через flow, внедрение подписи шаблонов и пр. Сегодня хотелось бы рассказать о возможных атаках на Nuclei, а именно о клиентских атаках, которые стали доступны благодаря этим нововведениям.

Далее рассказ буду вести в контексте двух сторон – Алисы, которая хочет защитить свой веб-сервер от сканирования Nuclei, и Боба, который, собственно, и запускает Nuclei.

Рис.1 Автор: С. О. Савченко, 2024.

Локальный вектор атаки

Для начала определим векторы атаки, которые доступны Алисе. Первый и самый очевидный – локальный, если Алиса каким-то образом подменит легитимный шаблон или убедит Боба запустить вредоносный шаблон, который совершит какие-либо небезопасные действия.

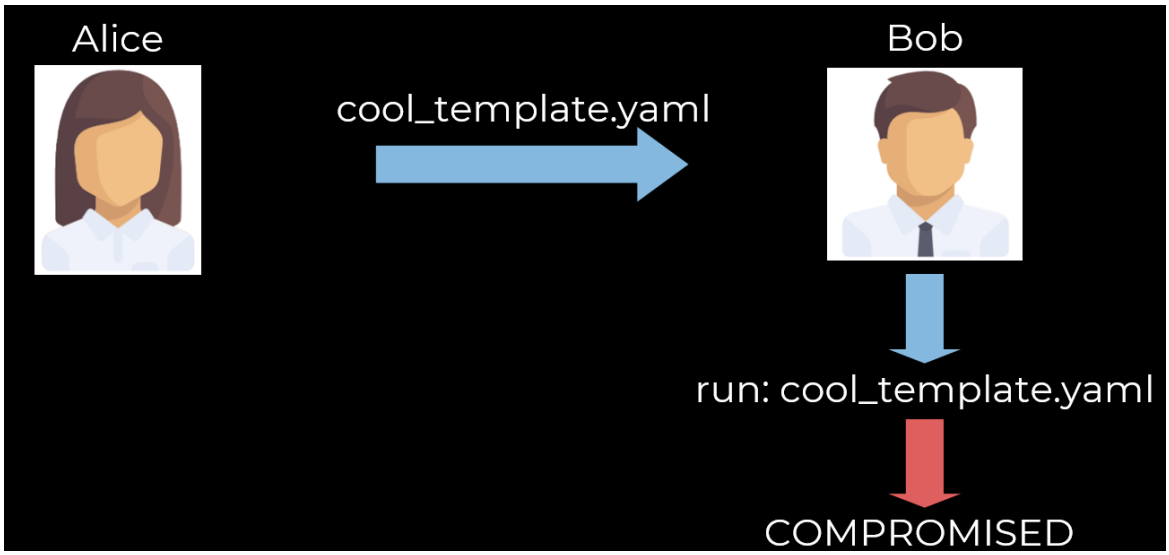


Рис.2 Локальный вектор. Автор: С. О. Савченко, 2024.

Рассмотрим существующий шаблон, который детектирует наличие уязвимости EternalBlue (CVE-2017-0144). В самом шаблоне используется кодовая вставка на python для взаимодействия с SMB, как видим на скриншоте ниже.

Предположим, что Алиса поместит нужный ей код за пределами экрана, в данном случае – реверс шелл, выделенный красным.

Рис.3 Автор: С. О. Савченко, 2024.

Алиса предварительно запускает netcat и отправляет Бобу шаблон. Боб запускает шаблон, Nuclei на секунду подвисает, а Алиса успешно ловит шелл.

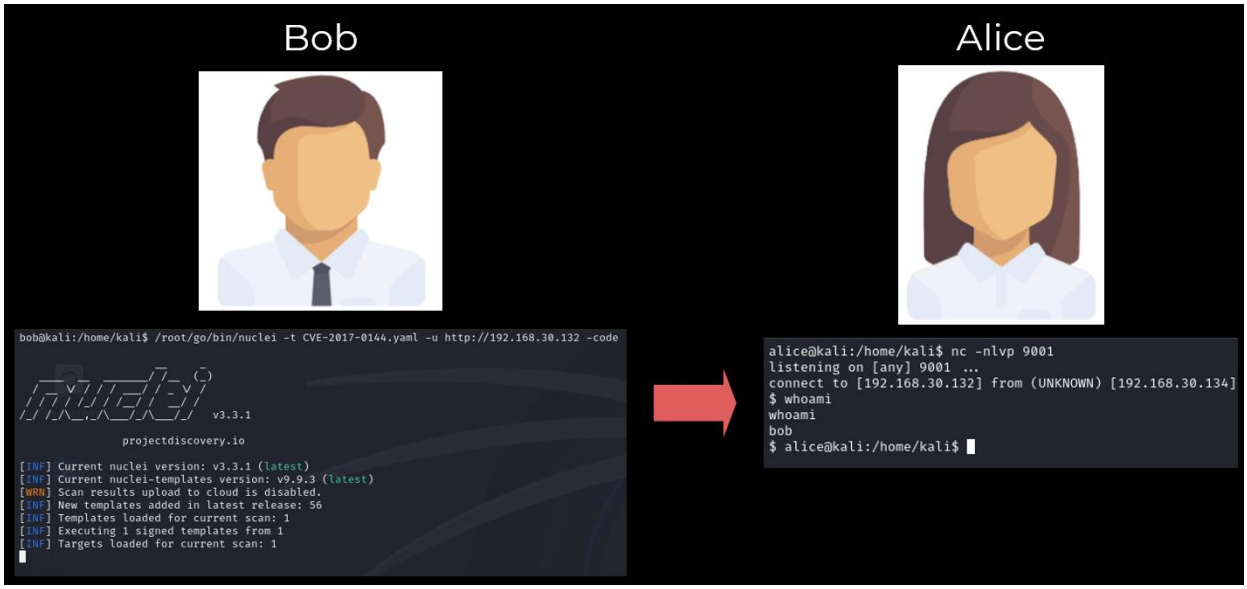


Рис.4 Автор: С. О. Савченко, 2024.

Можно сказать, что сценарий слишком надуманный и никто не будет запускать непонятный код у себя на хосте без проверок или песочницы, но буквально недавно мне попался репозиторий с 6к шаблонов для Nuclei. Кажется, что при желании можно спрятать где-то в глубине более хитрую обфусцированную нагрузку, учитывая намного меньшее количество проверяющих, по сравнению с официальным репозиторием. Кроме того, задача упрощается, если шаблоны предназначены в workflow, т.е. в последовательной цепочке.

Минусом данного подхода является то, что модифицированный шаблон нужно переподписать и, помимо этого, при запуске Nuclei применять специальный параметр для использования кодовых вставок.

Сетевой вектор атаки

Второй вектор – сетевой. Когда Боб запускает, казалось бы, легитимный шаблон, совершается запрос к серверу Алисы, но ответ сервера, благодаря специфике работы Nuclei, приводит к каким-то негативным последствиям.

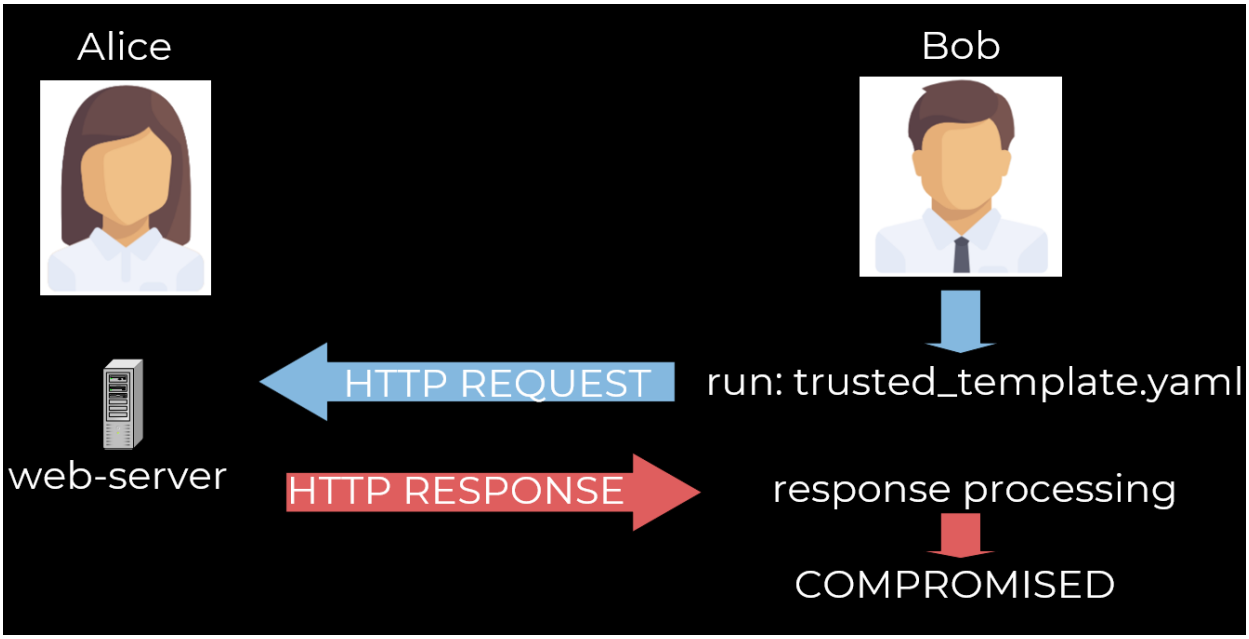


Рис.5 Автор: С. О. Савченко, 2024.

Для начала нужно определиться, что Алиса вообще может внедрить и куда. Учитывая, что Nuclei должен взять ответ сервера Алисы и как-то использовать, наиболее вероятным местом атаки являются экстракторы.

Экстракторы позволяют извлекать из ответа хоста различную информацию, например, с помощью регулярных выражений и далее при необходимости с помощью встроенных функций Nuclei (называемых «хелперы») преобразовывать ее – например, брать хэш-сумму, кодировать в base64, приводить к нижнему регистру и прочее.

На скриншоте ниже можно увидеть пример, когда из HTTP-ответа сервера с помощью регулярного выражения извлекается заголовок Server и числовое значение (в данном случае – 1.14.1) помещается в переменную version.

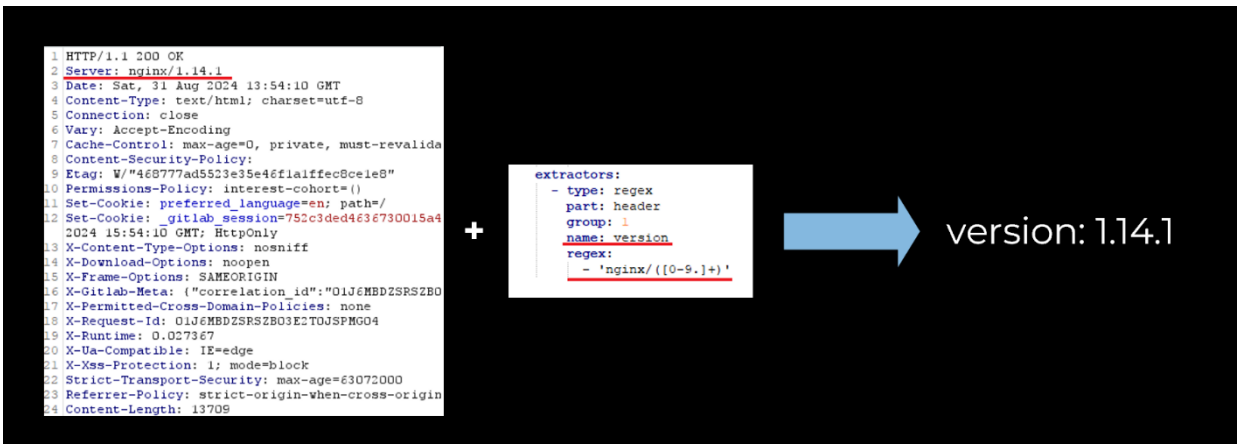
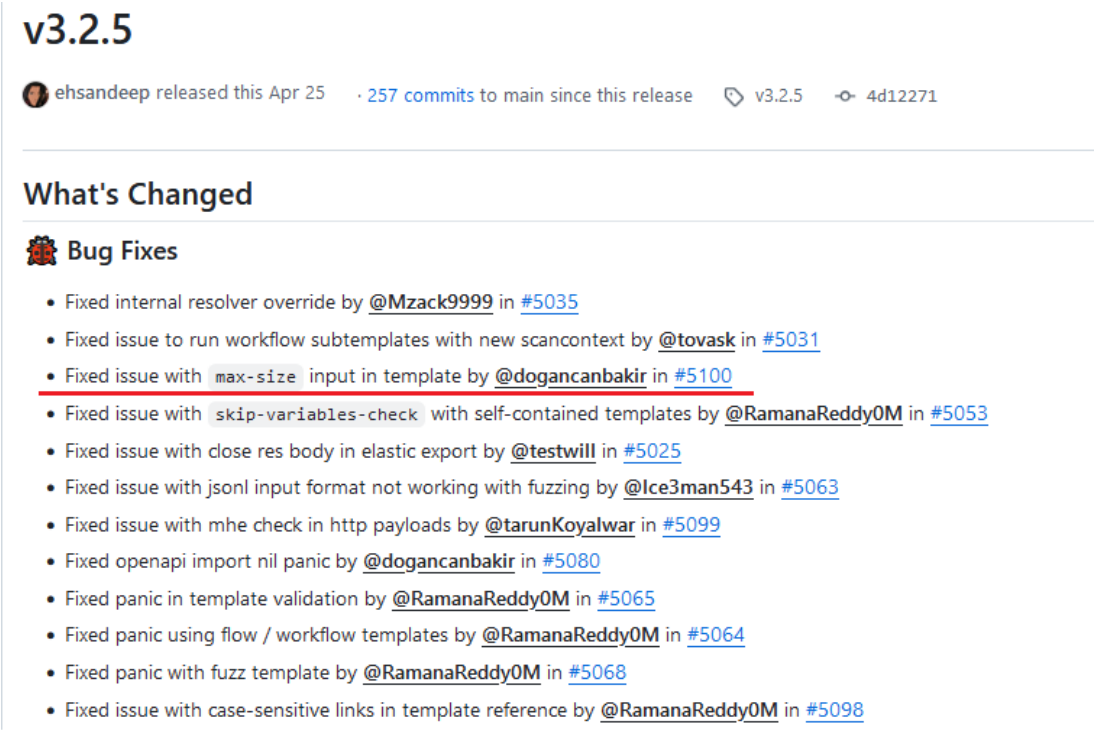


Рис.6 Экстракторы. Автор: С. О. Савченко, 2024.

Устаревшая версия Nuclei

Если Алиса посмотрит в официальном репозитории changelog, то увидит, что не так давно в Nuclei исправили ошибку, которая вызывала падение процесса при слишком большом HTTP-ответе.



v3.2.5

ehsandeep released this Apr 25 · 257 commits to main since this release v3.2.5 4d12271

What's Changed

Bug Fixes

- Fixed internal resolver override by @Mzack9999 in #5035
- Fixed issue to run workflow subtemplates with new scancontext by @tovask in #5031
- Fixed issue with `max-size` input in template by @dogancanbakir in #5100
- Fixed issue with `skip-variables-check` with self-contained templates by @RamanaReddy0M in #5053
- Fixed issue with close res body in elastic export by @testwill in #5025
- Fixed issue with jsonl input format not working with fuzzing by @Ice3man543 in #5063
- Fixed issue with mhe check in http payloads by @tarunKoyalwar in #5099
- Fixed openapi import nil panic by @dogancanbakir in #5080
- Fixed panic in template validation by @RamanaReddy0M in #5065
- Fixed panic using flow / workflow templates by @RamanaReddy0M in #5064
- Fixed panic with fuzz template by @RamanaReddy0M in #5068
- Fixed issue with case-sensitive links in template reference by @RamanaReddy0M in #5098

Рис.7 Nuclei changelog

Рассмотрим сценарий, что у Боба установлен устаревший Nuclei, подверженный этой проблеме.

Схема взаимодействия выглядит следующим образом: шаблон отправляет HTTP-запрос серверу Алисы, получает ответ и извлекает какие-либо значения, после чего передает их кодовой вставке для обработки, но из-за ограничения на длину ответа возникает отказ в обслуживании. Ошибка возникает именно на этапе передачи значений из экстрактора в кодовую вставку.

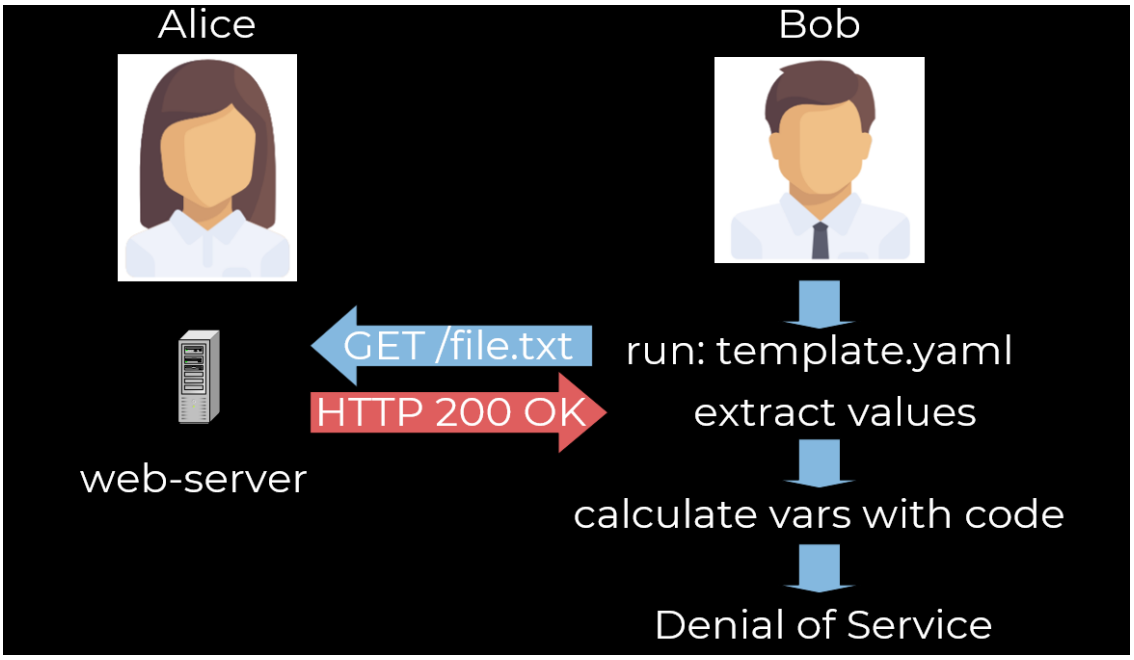


Рис.8 Nuclei DoS. Автор: С. О. Савченко, 2024.

Для тестирования Алиса поднимает простой веб-сервер и создает файл core.js размером 23мб. Боб запускает Nuclei с шаблоном, который должен обратиться к файлу core.js, взять тело ответа и обработать с помощью кодовой вставки. Nuclei показывает, что сканирование завершилось успешно и не принесло детектов, но, если посмотреть в режиме отладки, то Боб увидит, что процесс на самом деле упал из-за слишком длинного аргумента, переданного кодовой вставке.

Рис.9 Nuclei DoS PoC. Автор: С. О. Савченко, 2024.

Данный сценарий не всегда может быть применим, потому что у Боба может быть обновленная версия Nuclei. Кроме того, если у веб-сервера Алисы есть пользовательский функционал, наличие больших js-файлов может негативно на него повлиять.

Доказательство концепции CSTI

Рассмотрим другой сценарий, с актуальной версией Nuclei. Предположим, у Боба есть легитимный шаблон, который делает первый HTTP-запрос к файлу file.txt, извлекает из тела ответа что-то с помощью регулярного выражения и делает второй запрос, подставляя в заголовок CustomHeader извлеченное значение.

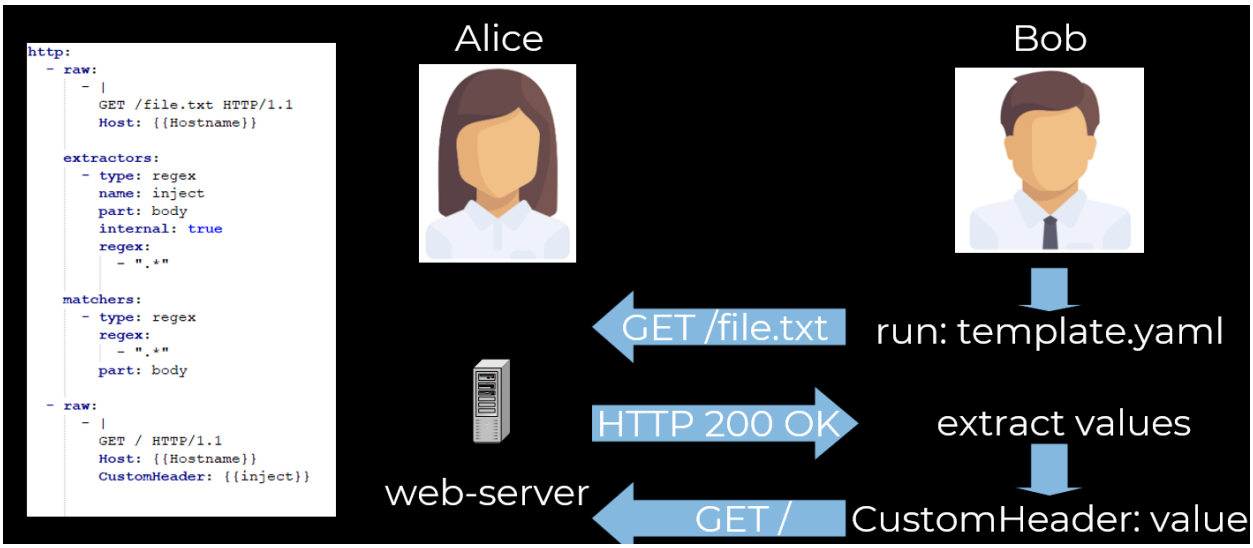


Рис.10 CSTI PoC. Автор: С. О. Савченко, 2024.

Это довольно распространенная ситуация, которая в чуть более сложном виде встречается в официальных шаблонах от ProjectDiscovery.

Одновременно с этим Алиса поднимает свой сервер и в файл file.txt помещает хелпер, который можно увидеть на картинке ниже, который при успешной инъекции должен вернуть строку CSTITEST, закодированную в base64.

Боб запускает сканирование сервера Алисы и, если он включит режим отладки, то увидит, что первый запрос выполнился корректно, содержимое файла file.txt отображено в первоначальном виде, но во втором запросе в заголовке CustomHeader фигурирует значение base64, а не исходная строка.

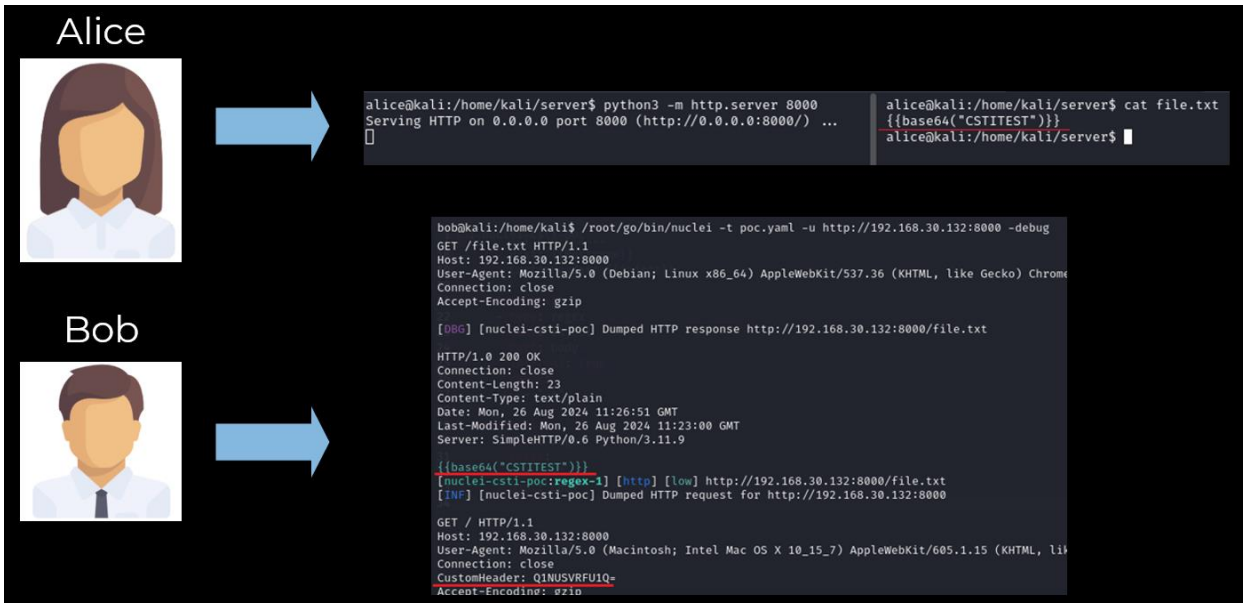


Рис.11 CSTI PoC. Автор: С. О. Савченко, 2024.

Таким образом, практически любой хелпер, попавший в экстрактор, не подвергнется экранированию и будет выполнен.

Несмотря на то, что эта атака – в чистом виде Client Side Template Injection, Алиса не может обращаться к системным файлам, перезаписывать что-то или открывать соединения, потому что под капотом используется собственная библиотека от ProjectDiscovery, функционал которой с точки зрения зловредной нагрузки довольно ограничен и содержит около 80 функций. Но даже этих функций достаточно для развития трёх направлений, о которых речь пойдет далее.

Отказ в обслуживании

Если Алиса внимательно посмотрит на список функций, то ее внимание привлекут две – номер 28 (вычисление md5 хэш-суммы от строки) и 53 (повторение символа заданное количество раз), комбинация которых может привести к старому-доброму отказу в обслуживании.

Рис.12 CSTI to DoS. Автор: С. О. Савченко, 2024.

Итак, для этого повторим предыдущий сценарий, только вместо полезной нагрузки Алиса использует функцию вычисления MD5 хэш-суммы от 4 миллиардов букв A.

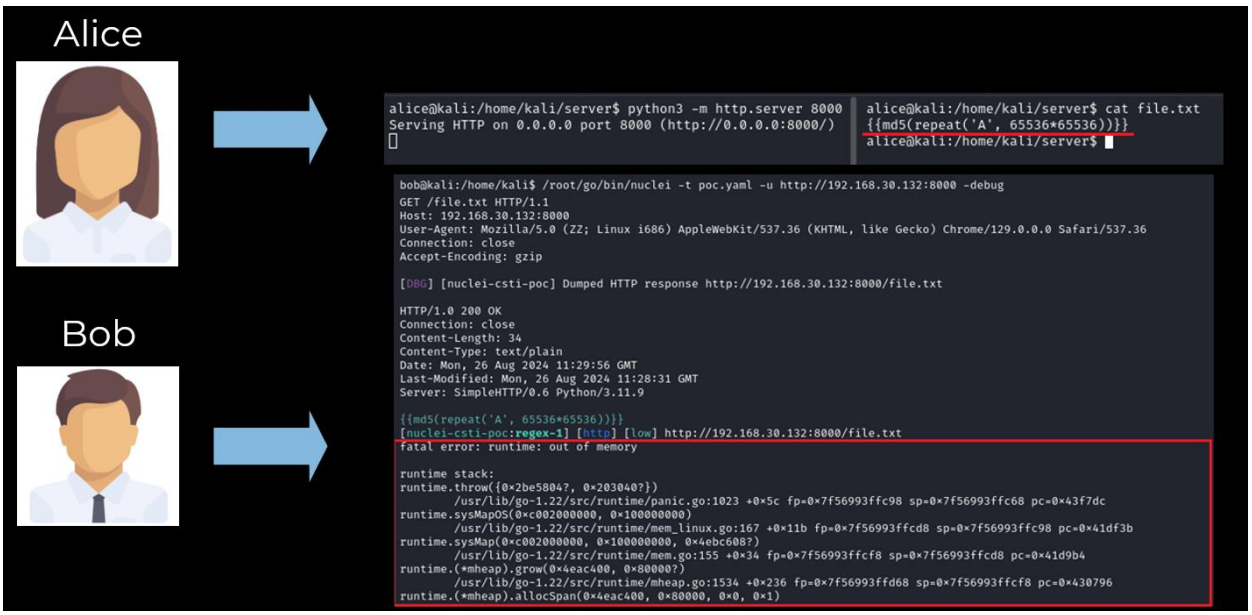


Рис.13 CSTI to DoS. Автор: С. О. Савченко, 2024.

Боб аналогичным образом запускает Nuclei со своим шаблоном, первый запрос выполняется успешно, но вместо выполнения второго запроса процесс падает через несколько секунд из-за исчерпания всей выделенной памяти.

Следует отметить, что подобная атака может быть реализована и другой полезной нагрузкой, как показано на картинке ниже. Здесь используется функция, дописывающая 4 миллиарда букв A в качестве префикса к строке с тем же эффектом. Таким образом, используя этот или ранее приведенный метод, Алиса может стать невидимкой для Боба, потому что ни один запущенный процесс сканирования не даст детектов. При этом весь пользовательский функционал может быть сохранен, ведь полезная нагрузка (в зависимости от экстрактора в шаблоне) может быть размещена даже в комментариях кода страницы.

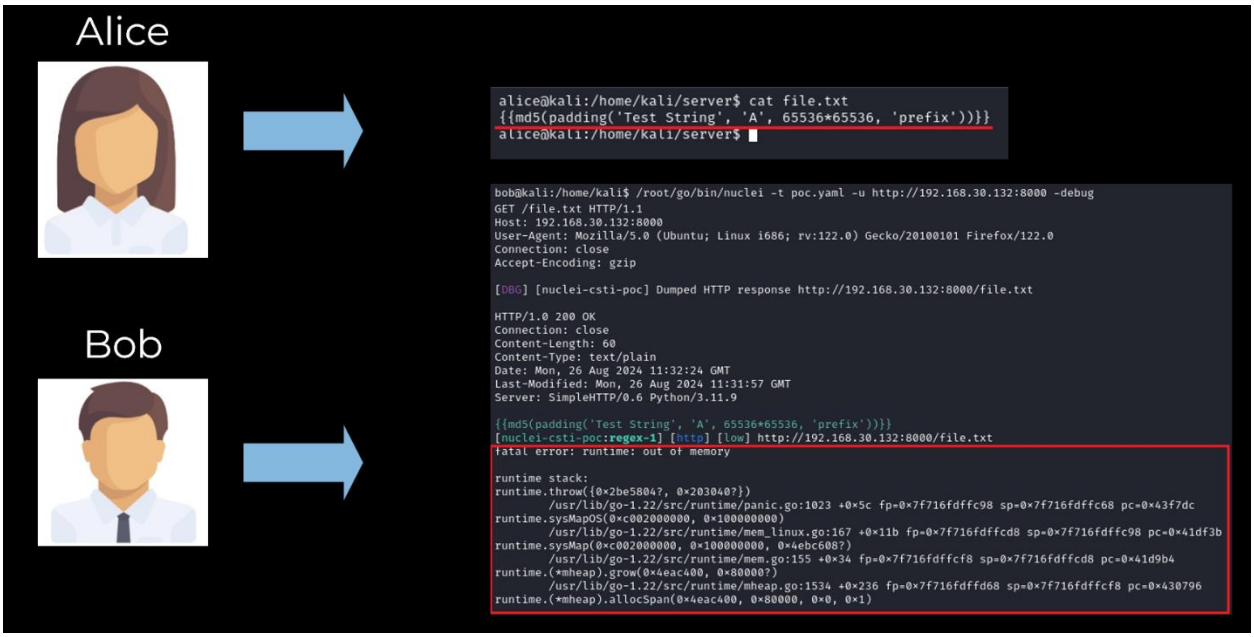


Рис.14 CSTI to DoS. Автор: С. О. Савченко, 2024.

Эксфильтрация данных

В одном из последних обновлений Nuclei была добавлена интеграция с ASM-системами – такими, как Shodan, Netlas, Fofa и пр. Для того, чтобы найти хосты, подверженные определенной уязвимости и запустить их сканирование с помощью Nuclei, в официальной документации ProjectDiscovery есть соответствующая инструкция, в которой API ключи для ASM просто импортируются в переменные среды.

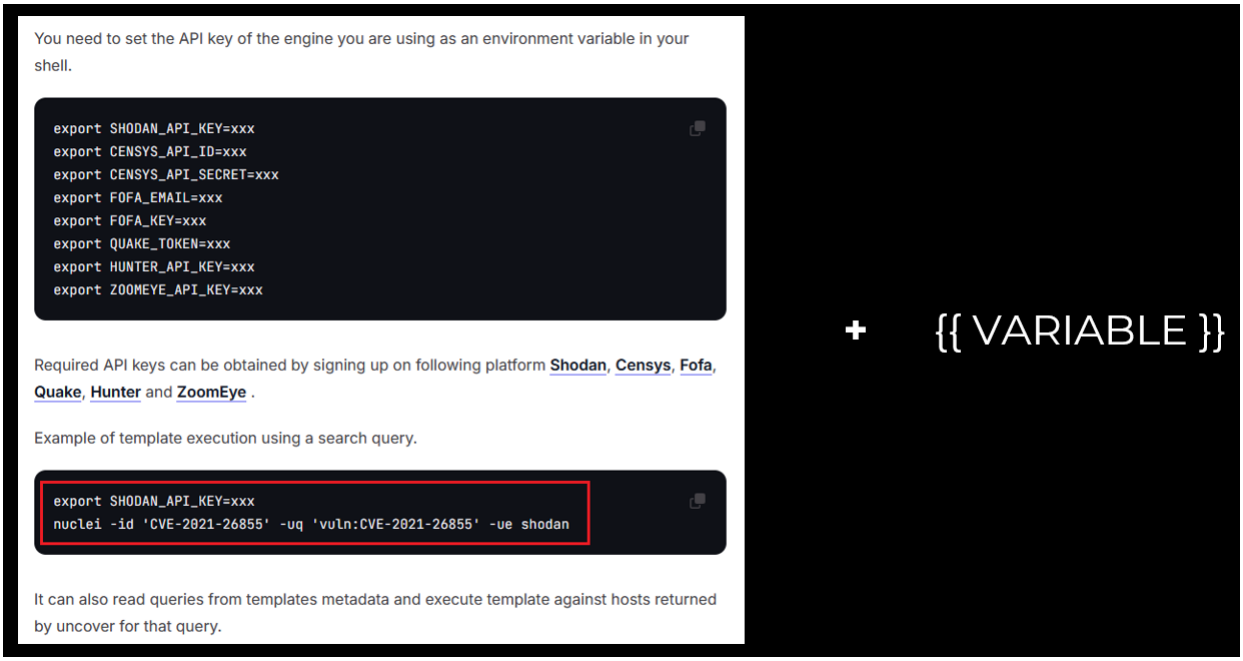


Рис. 15 Автор: С. О. Савченко, 2024.

Одновременно с этим хэлперы Nuclei позволяют читать переменные для их дальнейшего использования – локальные и даже глобальные, если Nuclei запущен с соответствующим флагом. Таким образом, комбинациях этих фактов позволяет реализовать эксфильтрацию чувствительных данных через HTTP-запросы.

Рассмотрим более сложный сценарий. В этот раз Боб использует существующий шаблон из официального репозитория. В шаблоне идет HTTP-запрос по специфичному пути, в ответ приходит json, из которого извлекается значение поля name. Далее это значение используется в качестве URL для второго HTTP-запроса.

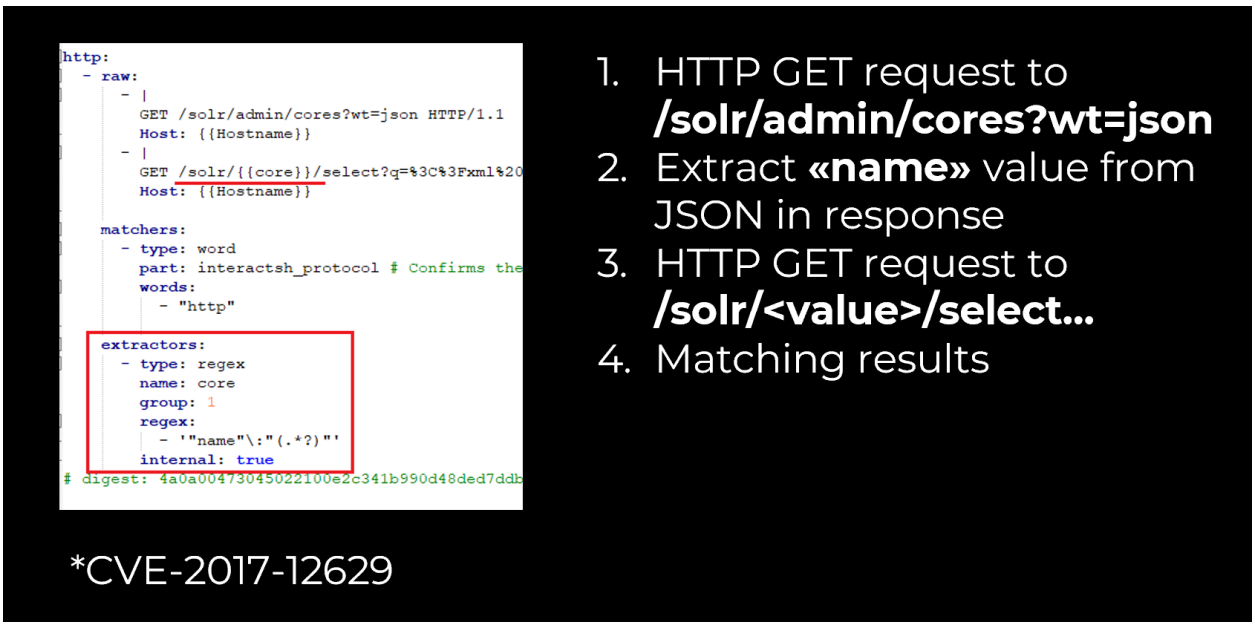


Рис.16 CSTI to HTTP Data Exfiltraton. Автор: С. О. Савченко, 2024.

Предположим, Боб активно использует поисковую систему Shodan, у него есть API ключ для взаимодействия с ней. Для удобства он поместил этот ключ в качестве переменной SHODAN_KEY командой export.

Алиса же, в свою очередь, хочет получить этот ключ. Для этого она подобрала такое содержимое файла, чтобы при работе экстрактора запускалась функция, которая извлекает данные из окружения и далее подставляется во второй запрос.

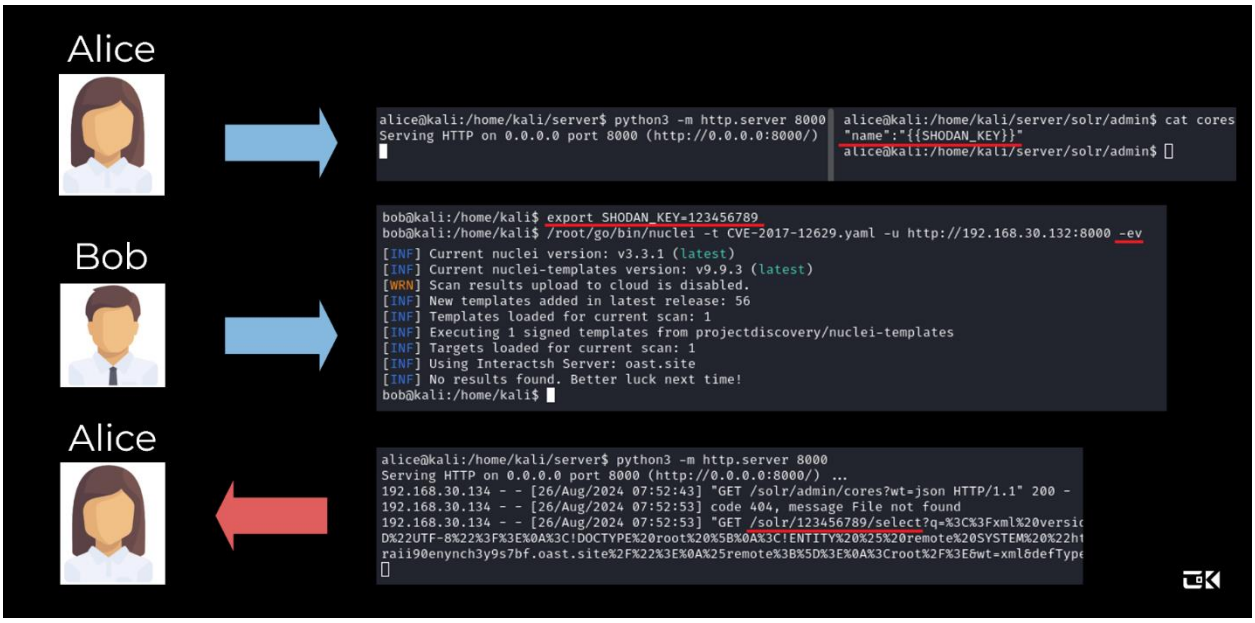


Рис.17 CSTI to HTTP Data Exfiltraton (single). Автор: С. О. Савченко, 2024.

Боб запускает описанный шаблон, выполняется первый запрос, который в ответ получает JSON. Из JSON извлекается значение поля name – в данном случае, хэлпер, вызывающий переменную SHODAN_KEY. Как описывалось ранее, практически любой хэлпер, попавший в экстрактор и переданный далее, автоматически выполняется. Таким образом, в URL для второго запроса подставляется не исходное значение поля name, а фактическое значение переменной SHODAN_KEY. Алиса может увидеть это значение, посмотрев в логи своего сервера, на скриншоте выделенное красным.

Помимо этого, Алиса может извлечь сразу значения для других систем – например, для Netlas и Fofa, немного поменяв полезную нагрузку. После запуска шаблона у Алисы в логах отобразятся все извлеченные ключи.

Рис.18 CSTI to HTTP Data Exfiltraton (chain). Автор: С. О. Савченко, 2024.

Вполне возможно подобрать нагрузку, которая, например, ворует ключи или токены от Telegram, Discord, AWS и прочее.

Аналогичная ситуация, если Боб запускает Nuclei через docker. Зная возможные имена переменных, Алиса легко извлекает их, при том, что переменные в докере зачастую хранят более чувствительную информацию.

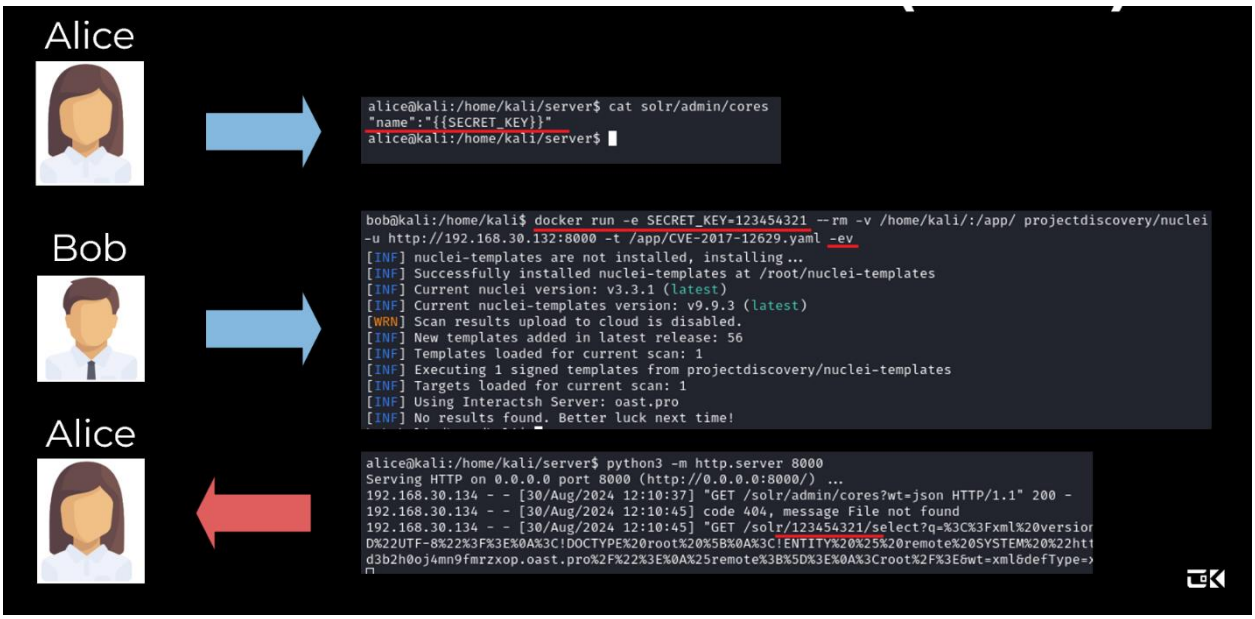


Рис.19 CSTI to HTTP Data Exfiltraton (docker). Автор: С. О. Савченко, 2024.

Минусом данного подхода является то, что без специального флага для Nuclei остаются доступными только внутренние переменные, обычно не хранящие какую-либо чувствительную информацию.

Атака на внешние обработчики

По мере распространения Nuclei, его поддержка начала появляться и в других инструментах, которые позволяют просматривать результаты сканирования большого количества целей в удобной форме или даже управлять этими сканированиями. При этом, зачастую при отображении результатов отсутствует проверка входных данных, что делает веб-интерфейс подверженным таким уязвимостям, как XSS.

Одним из таких инструментов является Osmedeus – фреймворк для автоматизации процесса пентеста и багбаунти, который поддерживает обычный запуск Nuclei и представление результатов сканирования в формате html.

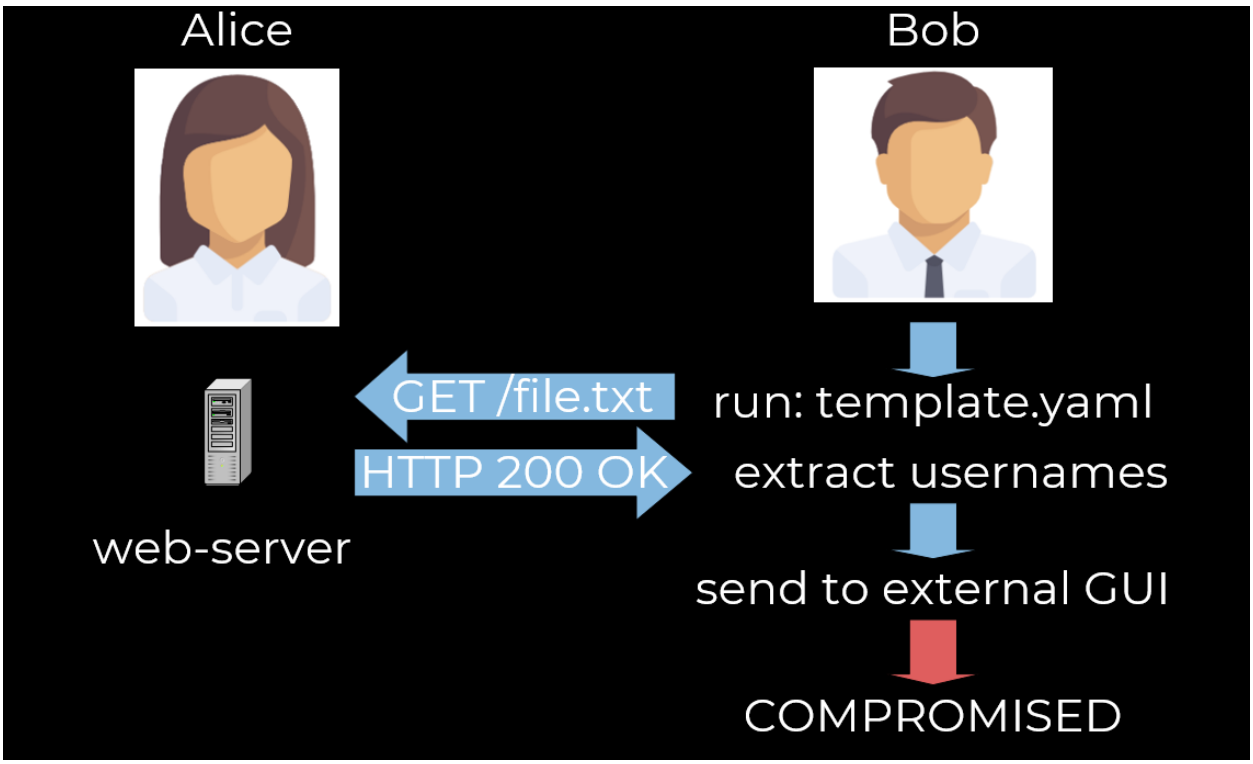


Рис.20 CSTI to Attack on external UI. Автор: С. О. Савченко, 2024.

Продолжим сценарий, когда Боб использует ASM-систему Shodan. С ее помощью Боб выгрузил некоторое количество хостов, подверженных определенной уязвимости. Пусть уязвимость заключается в возможности раскрытия юзернейма администратора в ответ на HTTP-запрос, и у Боба есть шаблон для эксплуатации данной уязвимости. Предполагая, что сканирование будет массовым, Боб использует Osmedeus. Для удобства он немного меняет стандартную форму отчета и добавляет столбец с извлеченными юзернеймами.

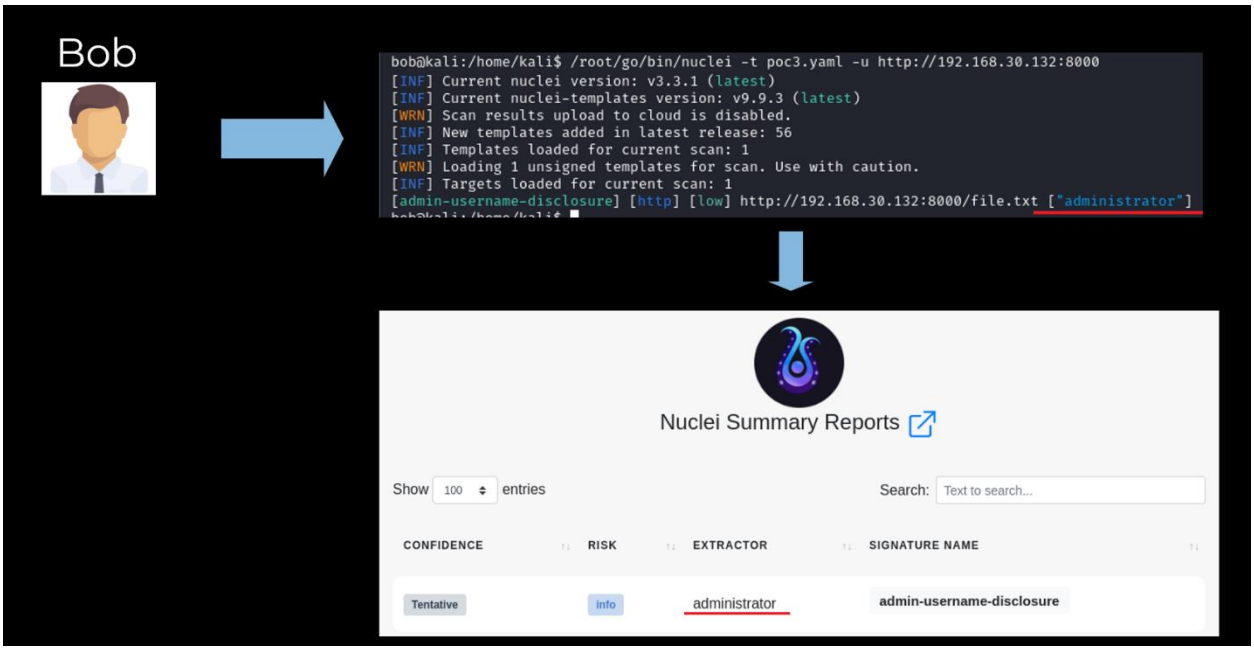


Рис.21 CSTI to Attack on external UI. Автор: С. О. Савченко, 2024.

При работе с обычным веб-сервером, без полезной нагрузки, Nuclei успешно завершает сканирование и результаты корректно отображаются сначала в консоли, а потом и в браузере.

Если же Алиса, зная принципы работы шаблона, вместо юзернейма подставит полезную нагрузку для XSS, то Nuclei успешно ее извлечет, сначала отобразит в консоли (что не несет негативных последствий), а затем передаст в веб-интерфейс, в результате чего появится всплывающее окно.

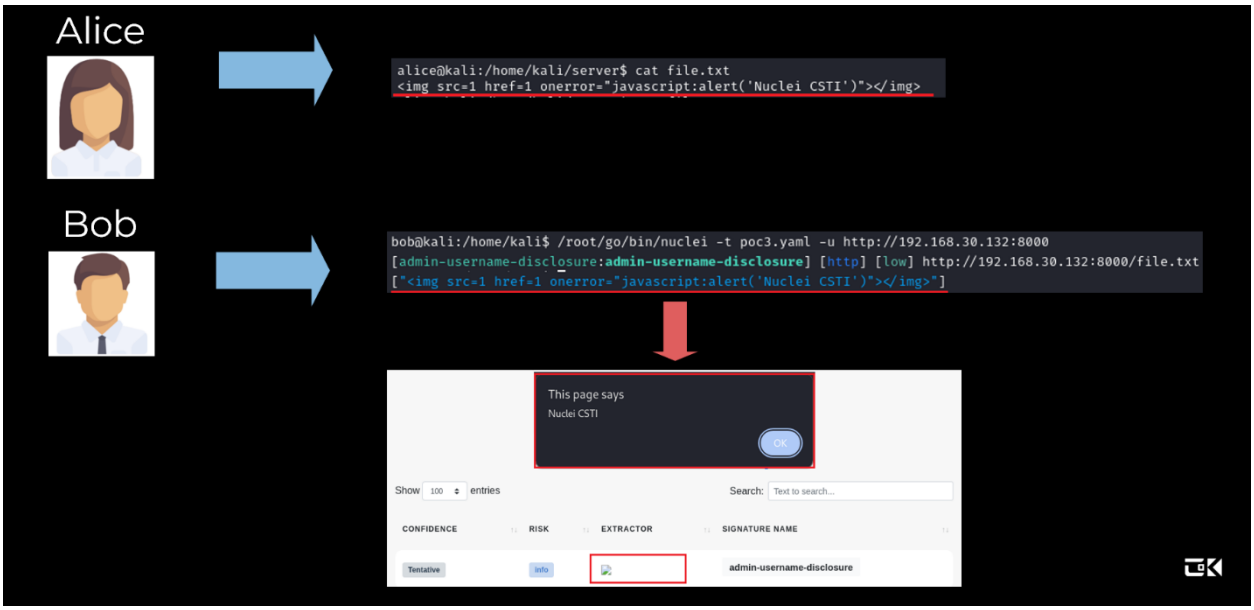


Рис.22 CSTI to Attack on external UI. Автор: С. О. Савченко, 2024.

В зависимости от обработчика, используемых технологий и контекста, атака может быть сведена к разным типам уязвимостей – как XSS, так XXE и прочие.

Нереализованные вектора

Далее расскажу о тех векторах, которые не удалось реализовать.

1) Не удалось развить полноценную атаку на внешние обработчики. Рассматривались Osmedeus, DefectDojo и ProjectDiscovery CloudPlatform и, в отличие от предыдущего примера, ни в одном обработчике напрямую не выводятся данные, полученные от хоста, по крайней мере в стандартных вариантах отчетов.

2) Не удалось реализовать инъекцию кода в кодовую вставку на python, bash или javascript без изменения легитимного шаблона. Судя по всему, в этой части экранирование происходит корректно, и полезная нагрузка передается только в виде строки и никак иначе.

3) При тестировании эксфильтрации данных не удалось прочитать переменные, которые прописываются в конфиг-файле Nuclei. В теории этот вектор может представлять большую опасность, так как в конфиге могут храниться, в том числе чувствительные данные для Jira, GitHub или GitLab для автоматического создания тикетов по итогам проведенного сканирования.

4) Не удалось развить ошибку исчерпания памяти в переполнение кучи или буфера для достижения удаленного выполнения кода.

При этом, вектор я бы оценил как очень перспективный, так как у Nuclei внутри используется несколько общедоступных библиотек. Некоторые из них обновляются регулярно (как goja), а вот некоторые, как, например, отвечающие за вычисления в экстракторах или подстановке значений в шаблонах, выглядят заброшенными – от 2 до 7 лет, а значит, могут содержать баги или ошибки в логике.

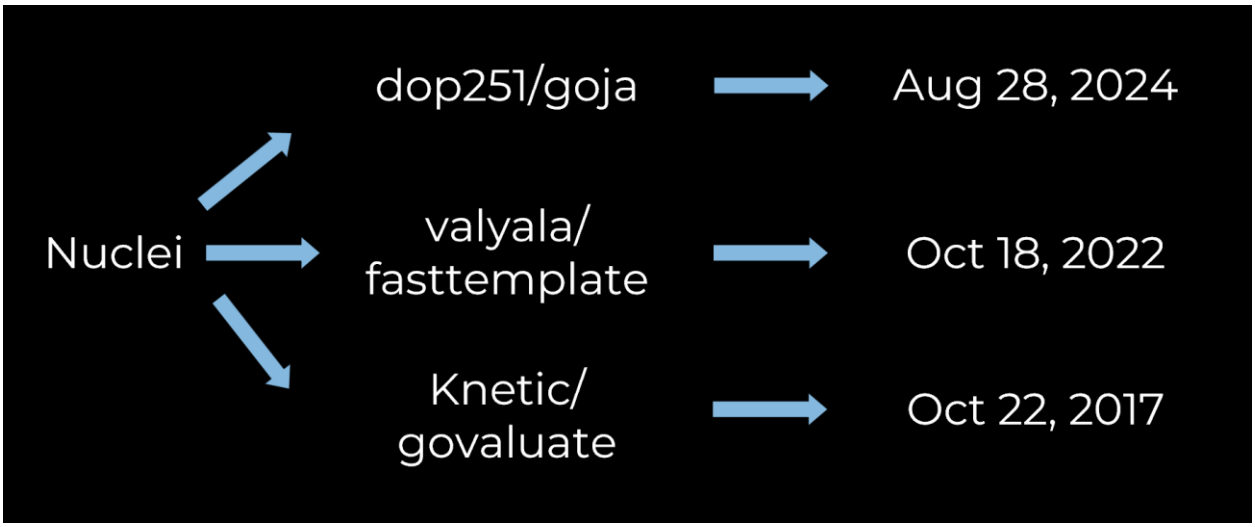


Рис.23 Автор: С. О. Савченко, 2024.

Рекомендации

В качестве рекомендаций по противодействию хотелось бы выделить несколько пунктов:

1. Запускать Nuclei нужно в изолированной среде.
2. Более тщательно прорабатывать матчеры и экстракторы для снижения доли false-negative и false-positive срабатываний.
3. Контролировать содержимое сложных шаблонов, особенно полученных из сторонних источников.
4. Использовать ханипоты – начиная от простого сбора статистики по используемым шаблонам и заканчивая методами невидимости, о которых я рассказывал ранее.

Заключение

Напоследок хотел бы дать ответ на два вопроса, которые напрашиваются сами собой. Первый из них – уведомляли ли мы Project Discovery о своих находках? Дело в том, что около года назад выходила небольшая статья от зарубежных коллег о возможных клиентских атаках в nuclei. В тот раз Project Discovery сообщили, что это не баг, а фича и разрешили публиковать любые подробности. С тех пор у Nuclei значительно расширился функционал, но возможность для атак не прикрыли. Поэтому ответ на первый вопрос – нет.

Reporting to ProjectDiscovery

As soon as we had all the details and a draft of this blog post, we reached out to ProjectDiscovery about our findings.

ProjectDiscovery responded, that they wish to clarify, that they use Knetic's govaluate for expression evaluation and not a template processor. Further they told us that the highlighted features pose no security risk and that they are intended.

We agree, that govaluate is not problematic here. The same goes for the ProjectDiscovery DSL which is also used to evaluate expressions in the same manner. The replacement of the placeholders in a given template is done with valyala/fasttemplate, which once again is not problematic itself.

The problematic part is in our opinion the way how nuclei uses those libraries. As we described nuclei takes unsanitized data from the server it is scanning and then evaluates it with the help of fasttemplate, dsl and govaluate. When extracting data from the scanned target for reuse, one would expect that nuclei sends the same data as-is. However, if the extracted data happens to be in the format of `{{data}}` or `§data§` (either by chance or malicious intent), it will undergo evaluation, and the resulting evaluated value will be used in subsequent requests.

We discussed this opinion with ProjectDiscovery, but they still don't see it as a security problem, and asked us to create a Github issue if we feel there is a bug that leads to a crash of nuclei.

Рис.24 https://threatcat.ch/blog/when-hunters-become-prey/ , 2023.

И второй вопрос – насколько угроза распространена? Здесь можно ответить, что беглый поиск по официальному репозиторию показал как минимум 5 шаблонов, уязвимых к инъекциям.