

# securitykitten.github.io/\_posts/2014-11-25-curious-korlia.md at master · malware-kitten/securitykitten.github.io

By Nick Hoffman

Archived: 2026-04-05 23:39:20 UTC

<b>layout</b>	category-post
<b>title</b>	Curious Korlia
<b>excerpt</b>	aka, the bisonal backdoor
<b>date</b>	2014-11-25 18:43:44 -0500

## Introduction

Reverse engineers organize discrete of pieces of malware into families. While digging through my malware collection I stumbled across this hash (B8FDfEE08DEEE5CCC1794BAF9ED553CE).

It turns out that this is a sample of the backdoor family known as Korlia. After doing some more digging, it turns out that Korlia doesn't seem to be that well documented or widely known. There is a little bit more written about it here.

Korlia shares a lot of features with common remote access tools. Such as:

- Downloading and executing files
- Listing and controlling processes
- Creating and deleting files
- Creating a remote shell

Given the little amount of public information on Korlia, this made it a good candidate for further research. There isn't an obvious C2 address called out in strings, although there are some bizarre strings. Depending on luck, those might be actual strings, or code that is being misinterpreted as a string.

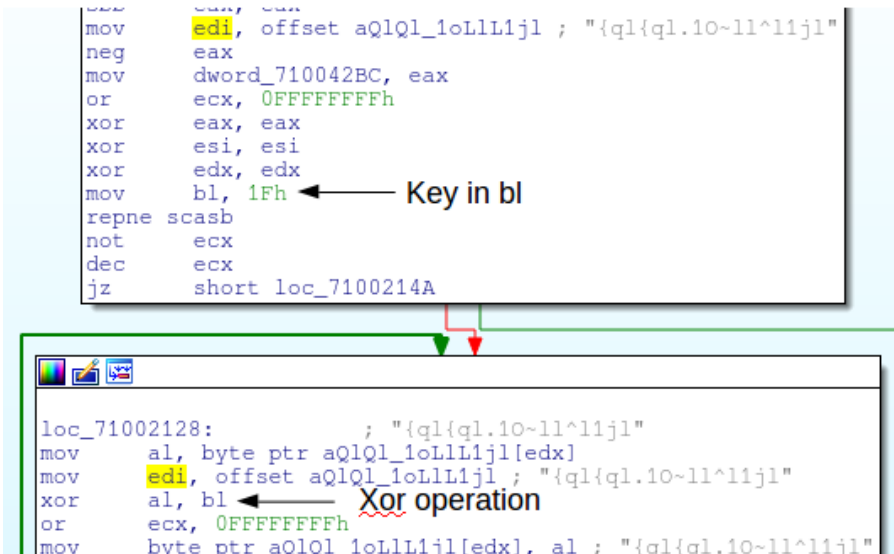
```
.data:71004... 00000008 C bisonal
.data:71004... 00000012 C {q|{q|.1O~|l^|1j|
.data:71004... 00000012 C {q|{q|.1O~|l^|1j|
.data:71004... 0000002F C wkko%00yjq{1|r|1pm1tm0]osp~{Yvsz0y~rz0g0p/1~lo
```

Diving in deeper on our first string:

```
Up o StartAddress+5A mov edi, offset aQ|Q|_1oL|L1j|; "{q|{q|.1O~|l^|1j|"
Up r StartAddress:loc_71002... mov al, byte ptr aQ|Q|_1oL|L1j|[edx]; "{q|{q|.1O~|l^|1j|"
```

There exists a cross reference to an address. For this case, that is a great sign! This particular piece of data is being referenced somewhere in the code. Let's follow.

In this case we can see that the data is being referenced as global data, and it's mov'd into an EDI. Shortly after, the value 0x1f is loaded into BL. As a general side note, when you see static values being pushed into the lower bytes of a general purpose register this usually means that some loop is going to follow and byte by byte modify a string or array.



This can be roughly written in Ruby with the following code.

```
{% highlight ruby %} {% raw %} "{ql{ql.10~1l^11j1".each_byte {|x| print "#{(x^0x1f).chr}"} {% endraw %} {%
endhighlight %}
```

This will return the following information:

```
dnsdns1.PassAs.us
```

After a little bit of hunting on VirusTotal, I was able to find the following samples. Which also have the following configurations.

- MD5|Config Offset|C2|C2|URL --- | --- | --- | --- | ---
- 172d68e10715b915ab3268db2174192b|11280|kfsinfo.ByInter.net|61.90.202.197|<http://fund.cmc.or.kr/UploadFile/fame/x/mh/o.asp>
- 211c25cdf120f5da8a2258b5d65cc263|14364|0906.toh.info|wew.myMom.info|<http://fund.cmc.or.kr/UploadFile/fame/x/o0.asp>
- 37513c17acfb0b122ffdc3e51501ecc3|11792|since.qpoe.com|69.197.149.98|<http://fund.cmc.or.kr/UploadFile/fame/x/o0.asp>
- 3f7b8f90acc4a01b3377942c409031dc|11808|mycount.MrsLove.com|mycount.MrsLove.com|<http://fund.cmc.or.kr/UploadFile/fame/x/c>
- 5217a2fc910479d36947d8fe6791d734|12816|mycount.MrsLove.com|mycount.MrsLove.com|<http://fund.cmc.or.kr/UploadFile/fame/x/>
- 7807036a74b811c28f1fbb167ef545e3|15900|kazama.myfw.us|<http://fund.cmc.or.kr/UploadFile/fame/x/o0.asp>
- 7865b3c7e7f40ead123e97aae5dc0a57|17948|shinkhek.myfw.us|<http://61.90.202.198/jp/log2.asp>
- 932875565fc6a1356800aa9d3af01670|11792|usababa.myfw.us|indbaba.myfw.us|<http://indbabababa.dns94.com/o.asp>
- b57a30d94872e47186c7ef2e08e6e905|17440|mycount.MrsLove.com|mycount.MrsLove.com|<http://fund.cmc.or.kr/UploadFile/fame/x/>
- b7981c7d028cbfd2f0fe2089de02b391|11792|jennifer998.lookin.at|196.44.49.154|<http://fund.cmc.or.kr/UploadFile/fame/x/o0.asp>
- b8fdfee08deee5ccc1794baf9ed553ce|11280|dnsdns1.PassAs.us|dnsdns1.PassAs.us|<http://fund.cmc.or.kr/UploadFile/fame/x/o0.asp>
- c96a92565553c7dc67267c78bc2809bb|14352|since.qpoe.com|applejp.myfw.us|<http://fund.cmc.or.kr/UploadFile/fame/x/o0.asp>
- cb0e358b534bdce8e2587ef3745b1723|11808|v3net.rr.nu|faceto.UglyAs.com|<http://fund.cmc.or.kr/UploadFile/fame/x/mh/o.asp>
- e47f4ca37db57a9f22d85e021dc891a6|12816|mycount.MrsLove.com|mycount.MrsLove.com|<http://fund.cmc.or.kr/UploadFile/fame/x/c>
- efe7598c675c1c71f0ad44cc686de587|17948|61.90.202.198|10.0.0.102|<http://61.90.202.198/jp/log.asp>

The next step in this process is to write a Yara rule looking for this sort of behavior. Writing Yara rules based on strings alone is often problematic as strings are very easy to change and modify. In this case, since we understand how the decoder works, writing a Yara rule for the loop is probably a better bet. While hunting I did find slight variations of the loop (highlighted in the Yara rule below). Those are accounted for in the final rule. The following rules will catch several variants of Korlia.

```
rule korlia
{
meta:
author = "Nick Hoffman "
company = "CBTS - ACS"
information = "korlia malware found in apt dump"

//case a
//b2 1f mov dl, 0x1f ; mov key (wildcard)
// -----
//8A 86 98 40 00 71 mov al, byte ptr url[esi]
//BF 98 40 00 71 mov edi, offset url
//32 C2 xor al, dl
//83 C9 FF or ecx, 0FFFFFFFh
//88 86 98 40 00 71 mov byte ptr url[esi], al
//33 C0 xor eax, eax
//46 inc esi
//F2 AE repne scasb
//F7 D1 not ecx
//49 dec ecx
//3B F1 cmp esi, ecx
//72 DE jb short loc_71001DE0

//case b (variant of loop a)
//8A 8A 28 50 40 00 mov cl, byte_405028[edx]
//BF 28 50 40 00 mov edi, offset byte_405028
//32 CB xor cl, bl
//33 C0 xor eax, eax
//88 8A 28 50 40 00 mov byte_405028[edx], cl
//83 C9 FF or ecx, 0FFFFFFFh
//42 inc edx
//F2 AE repne scasb
//F7 D1 not ecx
//49 dec ecx
//3B D1 cmp edx, ecx
//72 DE jb short loc_4047F2

//case c (not a variant of the above loop)
//8A 0C 28 mov cl, [eax+ebp]
//80 F1 28 xor cl, 28h
//88 0C 28 mov [eax+ebp], cl
//8B 4C 24 14 mov ecx, [esp+0D78h+var_D64]
//40 inc eax
//3B C1 cmp eax, ecx
//7C EE jl short loc_404F1C

strings:
$a = {b2 ?? 8A 86 98 40 00 71 BF 98 40 00 71 32 c2 83 C9 FF 88 86 98 40 00 71 33 C0 46 F2 AE F7 D1 49 3B F1}
$b = {B3 ?? ?? ?? 8A 8A 28 50 40 00 BF 28 50 40 00 32 CB 33 C0 88 8A 28 50 40 00 83 C9 FF 42 F2 AE F7 D1 49 3B F1}
$c = {8A 0C 28 80 F1 ?? 88 0C 28 8B 4C 24 14 40 3B C1}
$d = {00 62 69 73 6F 6E 61 6C 00} //config marker "\x00bisonal\x00"
condition:
any of them
}
```

---

Source: [https://github.com/malware-kitten/securitykitten.github.io/blob/master/\\_posts/2014-11-25-curious-korlia.md](https://github.com/malware-kitten/securitykitten.github.io/blob/master/_posts/2014-11-25-curious-korlia.md)