MOONSTRUCK:
USING LUA FOR DETECTION AND MALWARE TRAFFIC ANALYSIS

Dr Chris Wakelin, Senior Threat Analyst, Proofpoint

November 2018

Introduction

- “Lua” is Portuguese for “Moon”
- Small extensible language
- Considered “well-engineered” by experts :
  - “*If you read ... you'll see that the Lua team were well aware of many developments in programming languages and were able to choose from among the best ideas. Most designers of popular scripting languages have been amateurs and have not been nearly so well informed ... Lua is superbly designed so that the pieces fit together very nicely, with an excellent power-to-weight ratio ... Lua is superbly engineered. The implementation is just staggeringly good ...*”
- Currently used in
  - Wireshark
  - Games (Angry Birds, Crysis, Far Cry, Gary’s Mod …)
  - etc.

Introduction

- Lua(jit) scripting support added initially in September 2012
  - After suggestion by Will Metcalf of Emerging Threats
- Lua Output support added March 2014
- Lua flowvar support added in 2013
  - but only viewable (logged) from December 2016

Lua vs LuaJIT

- LuaJIT – Just-In-Time compiler for Lua
  - Stable version 2.0.5
  - Ubuntu 16.04 LTS included 2.0.4
- Development – version 2.1beta3 (for 18 months …)
  - Included in Ubuntu 18.04 LTS though
- Some caveats
  - Based on older Lua 5.1
  - Latest Lua is version 5.4
  - Need to pre-allocate buffers in Suricata
- Probably best to stick to Lua 5.1 features

Lua/LuaJIT options

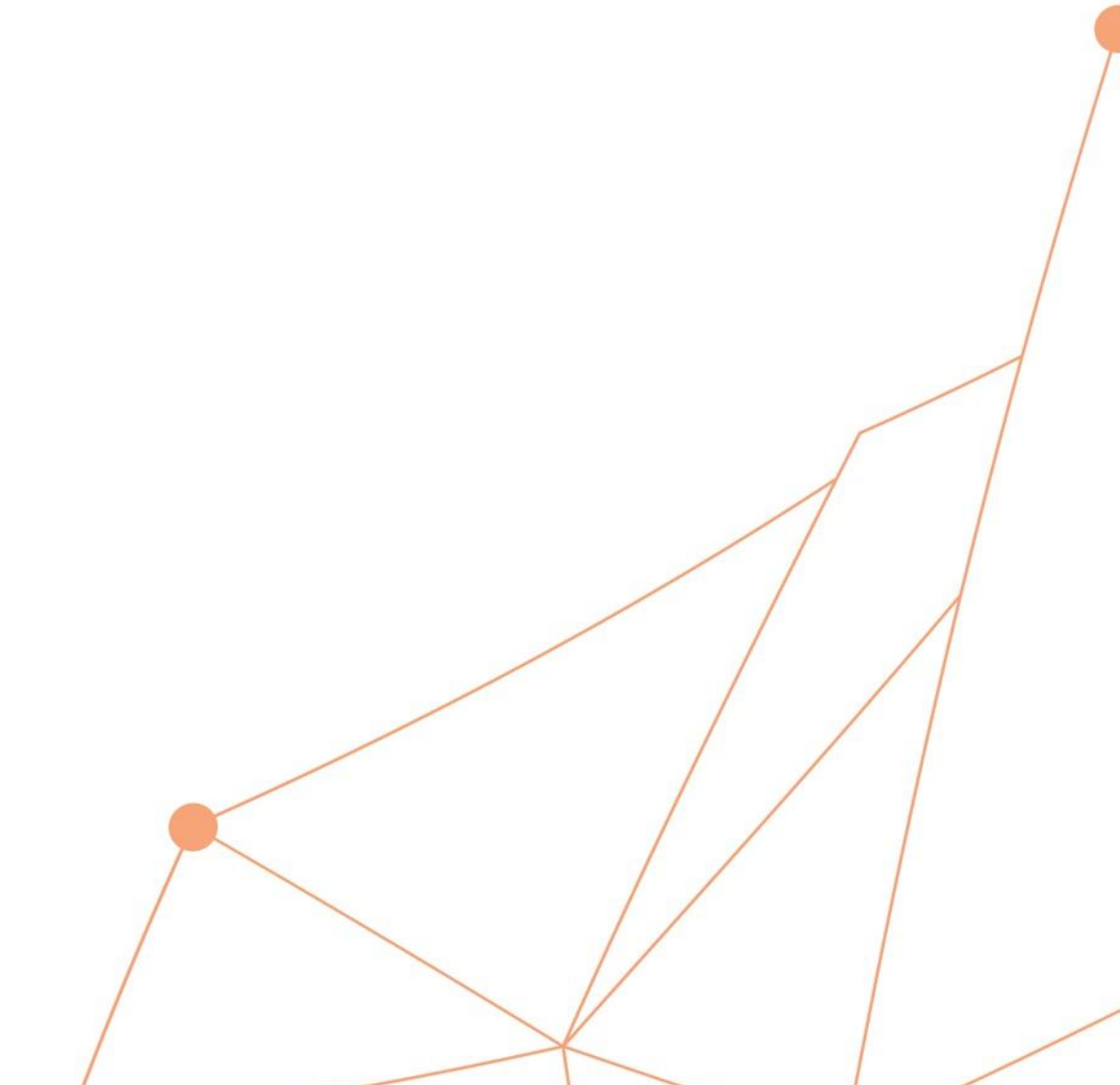
Suricata “configure” options (pick one)

```
--enable-lua            Enable Lua support
--enable-luajit         Enable Luajit support
```

suricata.yaml

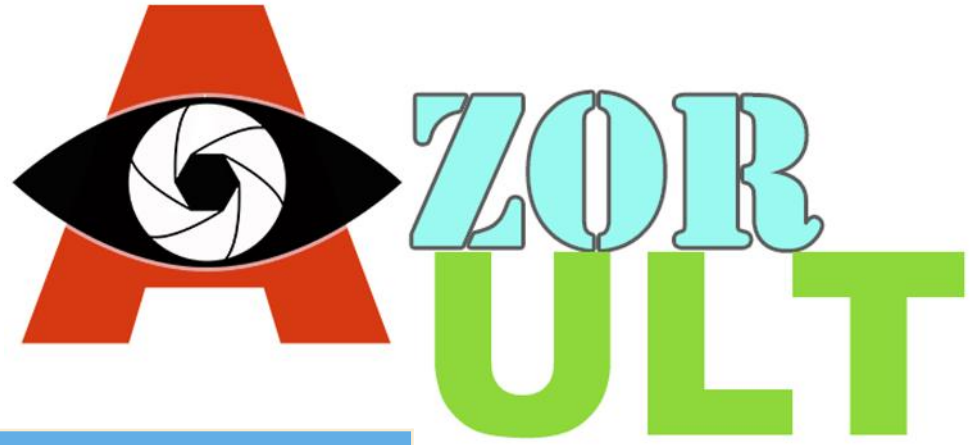
```
…
# Luajit has a strange memory requirement, it's 'states' need to be in the
# first 2G of the process' memory.
#
# 'luajit.states' is used to control how many states are preallocated.
# State use: per detect script: 1 per detect thread. Per output script: 1 per
# script.
luajit:
  states: 128
…
```

DETECTION



Example: AZORult

- Commodity “stealer” sold on underground forums
- Command-and-control traffic is obfuscated with XOR
- Only a couple of different keys seen
  - But maybe some variation
- Several ET Sigs:



SID	Description
2831936	ETPRO TROJAN AZORult Variant.4 XORed Downloa
2025885	ET TROJAN AZORult Variant.4 Checkin M2
2833315	ETPRO TROJAN AZORult Variant.5 Checkin M1
2833316	ETPRO TROJAN AZORult Variant.5 Checkin M2
2833317	ETPRO TROJAN AZORult Variant.5 Checkin Response
2810276	ETPRO TROJAN Azorult CnC Beacon

Public submissions

azorult ✖ | Significant tasks | Type hash to search

Windows 7 Professional 32bit 07 November 2018, 16:41	✓	Malicious activity	Shipping Details_PDF.scr PE32 executable (GUI) Intel 80386, for MS Windows trojan rat azorult	MD5: SHA1: SHA256:	5325B02699FFD208D5A124057DEA5438 EC93D34CAA357261485DEB5761267780A6901649 73BAA9C511FC131BC293B320C96AE4F86CA302B10C422B32BB35B4429C93176E
Windows 7 Professional 32bit 07 November 2018, 16:30	✓	Malicious activity	http://dodhmlaethandi.com/go/file1.exe PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows trojan rat azorult stealer	MD5: SHA1: SHA256:	6207454498608392A40FCE6675556DBC F73C90FECD3C1048619ACD4EF8103BD3D059E817 6EA8EEDD28490DEC588A6516E17F37891BE822DDFE88F9C6200972621DFEFA3B
Windows 7 Professional 32bit 07 November 2018, 06:30	✓	Malicious activity	b33bb258c1fca2448f248f578dd3dd1b488e92c01179a1f32f7ea1dba5cd... ASCII text, with very long lines, with CRLF, CR line terminators, with escape sequences exploit CVE-2017-11882 trojan rat azorult	MD5: SHA1: SHA256:	30474604715058SF329D12B25C6BE4FA 2125D522169F2FB15BC2BE5DB8A68B55B3910331 B33BB258C1FCA2448F248F578DD3DD1B488E92C01179A1F32F7EA1DBA5CDFA0D
Windows 7 Professional 32bit 07 November 2018, 03:57	✓	Malicious activity	http://gtfurobertopol.org/1/1.exe PE32 executable (GUI) Intel 80386, for MS Windows trojan rat azorult	MD5: SHA1: SHA256:	6766B75FAA519F755DAC79C04915BC35 B57AA793D63BDB425D10BD5BF2137B1E0E0908C3 7F66B08D4A420C01130B40E8491EFA3ABA0FF8D6119AC5D0FCB04340BB42CB65
Windows 7 Professional 32bit 06 November 2018, 15:29	✓	Malicious activity	OrderRequestList.docx Microsoft Word 2007+ trojan exploit CVE-2017-11882 loader rat azorult	MD5: SHA1: SHA256:	908293AA19389B914A899A2C7CE70593 CC62D922862BCB92E06E7809E4D346F9006955B4 D93F819A73FF9597F13106F7DAA5BB832BB9CA42692EEBEDA613DC3BA38787F1
Windows 7 Professional 32bit 06 November 2018, 14:46	✓	Malicious activity	GetBit install.exe PE32 executable (GUI) Intel 80386, for MS Windows trojan rat azorult	MD5: SHA1: SHA256:	BDD8984B04738420F0A13BDA342CD17F A9D1FD0EAA675B5021B2D33D465E119B1B25B95F FAD892142F180312391AD036AB7D32110604994262EE00988C4CFE75AA96001E
Windows 7 Professional 32bit 06 November 2018, 14:30	✓	Malicious activity	Scanned payment slip_JPG.rar RAR archive data, v5 trojan rat azorult	MD5: SHA1: SHA256:	A95F5BB64A3ED72E1D1D6B1CCD29603B 5BDD6187D5496E49E01C1A1E74C398CBE946E4F7 E33C92C483E6350A80C33CBC47E9F42EFD75D2A7F2B2DBB5129B380C32009394
Windows 7 Professional 32bit 06 November 2018, 14:11	✓	Malicious activity	http://dev.microcravate.com/crypt_AU3_EXE.exe PE32 executable (GUI) Intel 80386, for MS Windows trojan rat azorult	MD5: SHA1: SHA256:	CB6DE4D7079EB79E8F80C2013533A9B3 387C0FAE3883E3BCCDC39B1874DE397FCCCB76E9 5D1F46D751C9A9F630AF70957830E510E4BA5BE5C5774712573DB5FF3EDB1A38
Windows 7 Professional 32bit 06 November 2018, 13:58	✓	Malicious activity	Overdue Invoice.PDF.iso ISO 9660 CD-ROM filesystem data 'Overdue Invoice.PDF' trojan rat azorult	MD5: SHA1: SHA256:	BA418538E69505AD2DA0F2BEFBB10576 7FB3E5BA09D48686146B2EEF82DD2D09E4E729BB F3F83E2C43D675B67932C7EAD09F39A70E43F3D2722D0B237999A6EDCAE209BE
Windows 7 Professional 32bit 06 November 2018, 13:52	✓	Malicious activity	c975a2bce48679050db3fca9f335ddbaa53992f4778e8833e021a5b67fc7... ISO 9660 CD-ROM filesystem data 'Doc_3405960_PDF' trojan rat azorult	MD5: SHA1: SHA256:	EED3F3186396880855711E53AAE12EDD 85DD9AFE860BFC9D63F27E9F7DD71E2BA6488785 C975A2BCE48679050DB3FCA9F335DDBAA53992F4778E8833E021A5B67FC76B41
Windows 7 Professional 32bit 06 November 2018, 13:51	✓	Malicious activity	cecae5fbc2e73178362dd3e5e242e7d1540e959504675e834584de59e89... PE32 executable (GUI) Intel 80386, for MS Windows trojan rat azorult	MD5: SHA1: SHA256:	314BC120DAA8D8D871690A3676C18437 E6E8CF02BA04E666A9D66D7BD519846DC3D84B18 CECAE5FBC2E73178362DD3E5E242E7D1540E959504675E834584DE59E89B6EDF
Windows 7 Professional 32bit		Malicious activity	b8c60e39a832b511ab220ec88bacf5a86272fe31d0f7ec2c45f8beaed114... PE32 executable (GUI) Intel 80386, for MS Windows	MD5:	190627D08B45E1BE25FDD9B57701C87D

AZORult – C2 Traffic

Request:

```
00000000  00 00 00 26 66 9a 26 66  9c 45 70 9d 33 70 9d 30  |...&f.&f.Ep.3p.0|
00000010  70 9d 37 70 9d 3b 10 8b  31 11 8b 30 63 8b 30 6c  |p.7p.;..1..0c.0l|
00000020  ec 47 70 9d 33 70 9d 3b  70 9d 33 70 9d 33 70 9c  |.Gp.3p.;p.3p.3p.|
00000030  47 70 9d 30 70 9d 36 70  9d 37 10 8b 30 65 ed 26  |Gp.0p.6p.7..0e.&|
00000040  66 96 26 66 9a 26 67 ea  26 66 9c 26 66 9e 26 66  |f.&f.&g.&f.&f.&f|
00000050  9d 26 66 97 41 16 8b 30  6d ea 26 67 ea 26 66 9b  |.&f.A..0m.&g.&f.|
00000060  26 66 9b 26 66 98 26 66  9f 26 66 9f 26 66 9c 42  |&f.&f.&f.&f.&f.B|
00000070  16                                               |.|
```

Decoded:

```
00000000  25 33 34 25 33 32 46 25  33 30 25 33 33 25 33 34  |%34%32F%30%33%34|
00000010  25 33 38 45 25 32 44 25  33 36 25 33 39 42 44 25  |%38E%2D%36%39BD%|
00000020  33 30 25 33 38 25 33 30  25 33 30 25 32 44 25 33  |30%38%30%30%2D%3|
00000030  33 25 33 35 25 33 34 45  25 33 30 43 25 33 38 25  |3%35%34E%30C%38%|
00000040  33 34 25 32 44 25 33 32  25 33 30 25 33 33 25 33  |34%2D%32%30%33%3|
00000050  39 42 43 25 33 38 44 25  32 44 25 33 35 25 33 35  |9BC%38D%2D%35%35|
00000060  25 33 36 25 33 31 25 33  31 25 33 32 41 43        |%36%31%31%32AC|
```

Problem: “%” - escaping of randomly-selected characters –> no fixed strings

(Decodes to “42F0348E-69BD0800-354E0C84-2039BC8D-556112AC”)

AZORult – C2 Traffic

Response:

```
00000000  3f 36 90 4f 06 dd 71 1e  d7 33 21 e2 50 65 dc 4f  |?6.O..q..3!.Pe.O|
00000010  04 9e 48 07 c9 6f 22 f7  5b 1b d4 67 67 97 7a 0f  |..H..o".[..gg.z.|
00000020  e6 4e 1f e4 55 03 fa 51  03 e4 52 00 c5 3a 12 fd  |.N..U..Q..R..:..|
00000030  56 2d e8 49 03 d9 49 1e  c0 41 3d cd 30 18 df 4f  |V-.I..I..A=.0..O|
00000040  3b fc 37 31 ed 74 24 cd  4b 31 c5 48 3c 9b 33 30  |;.71.t$.K1.H<.30|
00000050  e6 52 26 e5 6e 39 c5 48  3c 9b 33 30 e6 52 26 e5  |.R&.n9.H<.30.R&.|
00000060  6e 1f 9e 5a 2c c1 76 31  e6 6b 65 e2 40 25 c4 61  |n..Z,.v1.ke.@%.a|
00000070  67 c2 76 1e c7 36 65 cb  4b 04 dd 48 3b ca 6b 37  |g.v..6e.K..H;.k7|
00000080  e9 74 24 e2 6d 07 9a 67  16 d9 72 37 e9 3a 3b cf  |.t$.m..g..r7.:;.|
00000090  54 61 df 4f 3b fc 37 31  ef 68 2d e3 47 14 e4 4f  |Ta.O;.71.h-.G..O|
000000a0  04 c5 71 16 ff 33 1e fa  42 39 c1 67 1d fc 74 1a  |..q..3..B9.g..t.|
000000b0  c7 3b 23 cd 31 13 c3 62  06 9b 69 37 d7 36 63 f7  |.;#.1..b..i7.6c.|
000000c0  50 6c 9d 62 02 9b 76 31  e6 64 20 f4 5b 3d c2 40  |Pl.b..v1.d .[=.@|
000000d0  06 dd 49 1e c9 33 1e fd  52 3e d6 4e 11 ff 76 18  |..I..3..R>.N..v.|
000000e0  c4 52 64 e2 69 10 9e 4d  06 9a 7a 1b fa 73 11 ff  |.Rd.i..M..z..s..|
000000f0  52 65 e5 3f 7a cd 3d 69  c0 3d fc bb 5a 79 0b 15  |Re.?z.=i.=..Zy..|
...
```

AZORult – C2 Traffic

Decoded Response:

```
00000000  3c 63 3e 4c 53 73 72 4b  79 30 74 4c 53 30 72 4c  |<c>LSsrKy0tLS0rL|
00000010  51 30 4b 52 67 6c 77 59  58 4e 7a 64 32 39 79 5a  |Q0KRglwYXNzd29yZ|
00000020  48 4d 4a 4a 56 56 54 52  56 4a 51 55 6b 39 47 53  |HMJJVVTRVJQUk9GS|
00000030  55 78 46 4a 56 77 4a 4b  6e 42 68 63 33 4d 71 4c  |UxFJVwJKnBhc3MqL|
00000040  6e 52 34 64 43 77 71 63  48 64 6b 4b 69 35 30 65  |nR4dCwqcHdkKi50e|
00000050  48 51 73 4b 6d 6c 6b 4b  69 35 30 65 48 51 73 4b  |HQsKmlkKi50eHQsK|
00000060  6d 4a 30 59 79 6f 75 64  48 68 30 4c 43 70 6a 62  |mJ0YyoudHh0LCpjb|
00000070  32 6c 75 4b 69 35 30 65  48 51 73 4b 6e 64 68 62  |2luKi50eHQsKndhb|
00000080  47 77 71 4c 6e 52 34 64  43 77 71 62 47 39 6e 61  |GwqLnR4dCwqbG9na|
00000090  57 34 71 4c 6e 52 34 64  41 6b 78 4d 44 41 4a 4c  |W4qLnR4dAkxMDAJL|
000000a0  51 6b 72 43 51 30 4b 54  41 6c 6f 64 48 52 77 4f  |QkrCQ0KTAlodHRwO|
000000b0  69 38 76 63 32 46 6d 61  53 35 6a 62 79 35 36 59  |i8vc2FmaS5jby56Y|
000000c0  53 39 33 61 57 35 75 64  48 67 75 5a 58 68 6c 43  |S93aW5udHguZXhlC|
000000d0  53 73 4a 4b 67 30 4b 53  51 6b 78 4d 44 51 75 4d  |SsJKg0KSQkxMDQuM|
000000e0  6a 51 31 4c 6a 45 30 4e  53 34 79 4e 54 70 44 51  |jQ1LjE0NS4yNTpDQ|
000000f0  51 30 4b 3c 2f 63 3e 3c  6e 3e a9 15 59 2c a5 16  |Q0K</c><n>..Y,..|
...
```

Now we're talking! "<c>[Base64 string]</c><n>[other stuff]…."

AZORult – Lua detection script (simplified)

```
local bit = require(“bit”)

function init (args)
    local needs = {}
    needs["http.response_body"] = tostring(true)
    return needs
end

...

-- return match via table
function match(args)
    local a = tostring(args["http.response_body"])
    return common(a, 0)
end

function run()
  local f = io.open(arg[1])
  local a = f:read("*all")
  f:close()
  common(a, 1)
end
```

AZORult – Lua detection script (simplified)

```
function common(a, verbose)

    if #a < 256 then
        return 0
    end

    -- cleartext starts "<c>" followed by Base64 string so deduce 3-byte XOR key
    key = {bit.bxor(a:byte(1), 0x3c), bit.bxor(a:byte(2), 0x63), bit.bxor(a:byte(3), 0x3e)}

    -- decode first 16 bytes after "<c>" and make sure they're valid Base64
    b = "<c>"
    for i = 4, 20, 1 do
       b = b .. string.char(bit.bxor(a:byte(i), key[((i - 1) % 3) + 1]))
    end

    data = string.match(b, "^<c>[A-Za-z0-9+/]*$")
    if data then
        if verbose == 1 then
            print("Found AZORult XORed Base64 command")
        end
        return 1
    end
    return 0
end
```

AZORult – Rules/Manual test

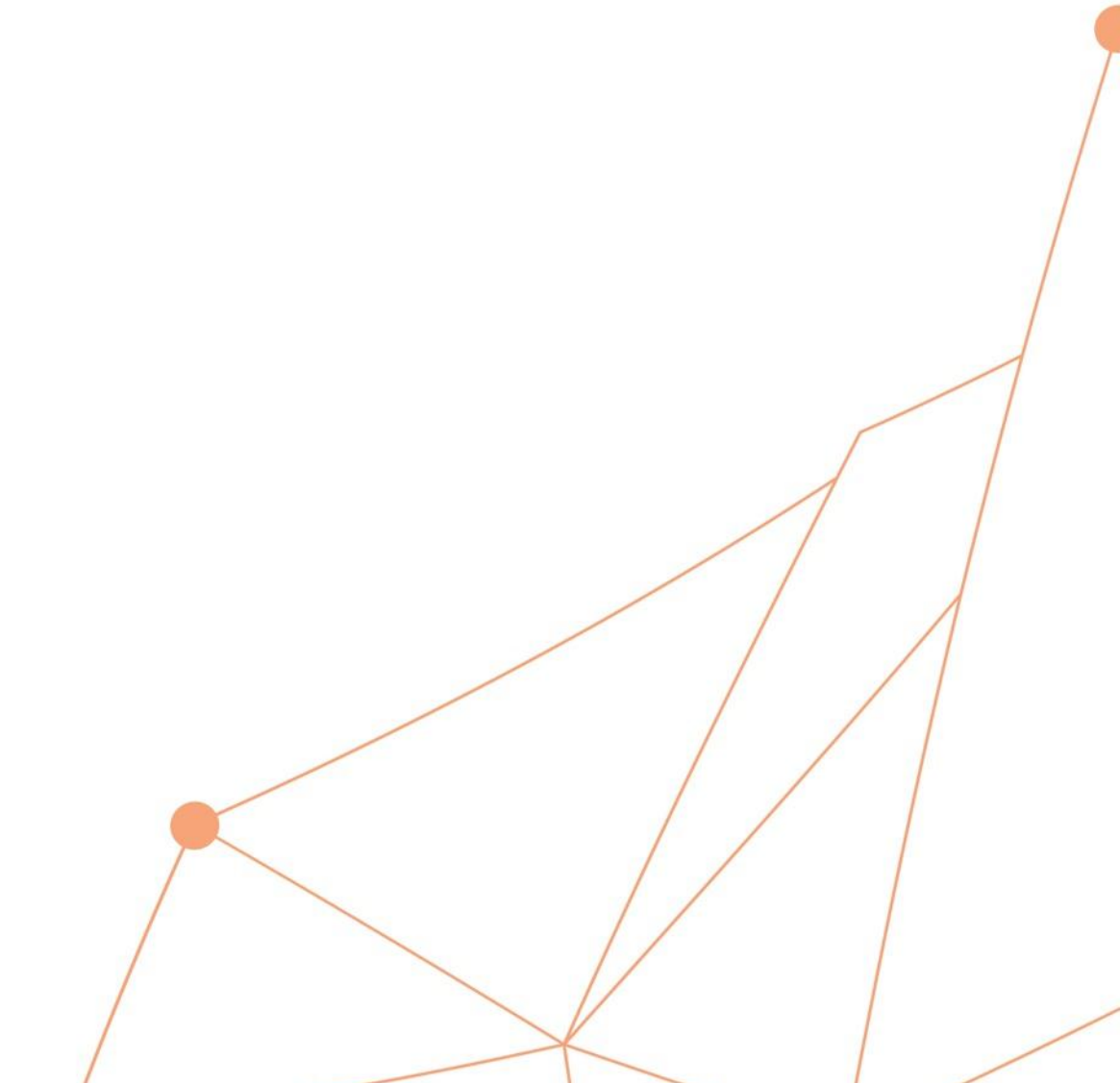
```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ET SUSPICIOUS .php POST with no Referer";
flow:established,to_server; content:"POST"; http_method; content:".php"; http_uri; http_header_names;
content:!"Referer"; flowbits:noalert; flowbits:set,ET.phppostnoref; classtype:bad-unknown; sid:xxxxx;
rev:1;)

alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ET LUA TROJAN AZORult XORed download";
flow:established,to_client; flowbits:isset,ET.phppostnoref; luajit:suri-azorult-detect.lua;
classtype:trojan-activity; sid:xxxxx; rev:1;)
```

```
echo 'run()' | lua5.1 -i suri-azorult-detect.lua suricata/files/file.6
Lua 5.1.5  Copyright (C) 1994-2012 Lua.org, PUC-Rio
> run()
Found AZORult XORed Base64 command
>
```

Lua Logging

Malware Traffic Analysis



Lua Logging

Going back to the Base64-encoded string we found earlier, decoding it leads to:

```
-++++----+-
F       passwords       %USERPROFILE%\
*pass*.txt,*pwd*.txt,*id*.txt,*btc*.txt,*coin*.txt,*wall*.txt,*login*.txt       100     -       +
L       hxxp://safi[.]co[.]za/winntx.exe    +       *
I       104.245.145[.]25:CA
```

(in this case what files to steal, a secondary download and the IP and locale of the victim)

Wouldn't it be nice if we could extract this automatically and store it Suricata Logs?

Lua Logging – Lua Output

```
local bit = require("bit")

function init(args)
    local needs = {}
    needs["protocol"] = "http"
    return needs
end

function setup(args)
    filename = SCLogPath() .. "/" .. "azorult.log"
    file = assert(io.open(filename, "a"))
    SCLogInfo("AZORult Log Filename " .. filename)
    azpost = 0
end

function deinit(args)
    SCLogInfo("AZORult posts logged: " .. azpost)
    file:close(file)
end
```

Lua Logging – Lua Output

```
function hex_esc(s)
    hex = string.gsub(s, "[^%w%p ]", function(c)
        return string.format("\\x%02x", string.byte(c))
    end)
    return hex
end

function base64_d(a)
-- requires a to be a valid base64 string
    keyStr = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/="
    b = ""
    for i = 1, #a, 4 do
        enc1 = keyStr:find(a:sub(i, i), 1, true)-1
        enc2 = keyStr:find(a:sub(i + 1, i + 1), 1, true) - 1
        enc3 = keyStr:find(a:sub(i + 2, i + 2), 1, true) - 1
        enc4 = keyStr:find(a:sub(i + 3, i + 3), 1, true) - 1
        chr1 = bit.bor(bit.lshift(enc1, 2), bit.rshift(enc2, 4))
        chr2 = bit.bor(bit.lshift(bit.band(enc2, 15), 4),bit.rshift(enc3, 2))
        chr3 = bit.bor(bit.lshift(bit.band(enc3, 3), 6), enc4)
        b = b .. string.char(chr1)
        if enc3 ~= 64 then b = b .. string.char(chr2) end
        if enc4 ~= 64 then b = b .. string.char(chr3) end
    end
    return b
end
```

Lua Logging – Lua Output

```
function log(args)

    r = HttpGetRequestLine()
    if r == nil then
        return
    end

    if string.sub(r,1,4) ~= "POST" or not string.match(r, "%.php ") then
        return
    end

    if HttpGetRequestHeader("Referer") then
        return
    end

    A = HttpGetResponseBody()
    a = ""
    for i = 1, #A, 1 do
        a = a .. A[i]
    end

    if #a < 256 then
        return
    end
```

Lua Logging – Lua Output

```
    h = HttpGetRequestHost()
    if h == nil then h = "<hostname unknown>" end

    ts = SCPacketTimeString()
    ipver, srcip, dstip, proto, sp, dp = SCFlowTuple()
    file:write(ts .. " " .. h .. " [**] Found possible AZORult POST : " .. r .. " [**] " .. srcip ..
":" .. sp .. " -> " ..  dstip .. ":" .. dp .. "\n")
    file:flush()

    -- cleartext starts "<c>" followed by Base64 string so deduce 3-byte XOR key
    key = {bit.bxor(a:byte(1), 0x3c), bit.bxor(a:byte(2), 0x63), bit.bxor(a:byte(3), 0x3e)}

    -- decode first 16 bytes after "<c>" and make sure they're valid Base64
    b = "<c>"
    for i = 4, 20, 1 do
       b = b .. string.char(bit.bxor(a:byte(i), key[((i - 1) % 3) + 1]))
    end

    data = string.match(b, "^<c>[A-Za-z0-9+/]*$")
```

Lua Logging – Lua Output

```
if data then

    azpost = azpost + 1
    hexkey = string.format("\\x%02x\\x%02x\\x%02x", key[1], key[2], key[3])

    for i = 21, 768, 1 do
       b = b .. string.char(bit.bxor(a:byte(i), key[((i - 1) % 3) + 1]))
    end

    D = HttpGetRequestBody()
    d = ""
    for i = 1, #D, 1 do
        d = d .. D[i]
    end

    -- strip possible leading 0x00 from request
    if string.sub(d, 1, 3) == "\0\0\0" then
       d = string.sub(d, 4)
    end

    -- decode request
    e = ""
    for i = 1, #d, 1 do
       e = e .. string.char(bit.bxor(d:byte(i), key[((i - 1) % 3) + 1]))
    end
```

Lua Logging – Lua Output

```
        file:write(ts .. " " .. h .. " [**] Decoded AZORult request - Key: " .. (hexkey) .. "
Request: " .. hex_esc(e) .. " [**] " .. srcip .. ":" .. sp .. " -> " ..  dstip .. ":" .. dp .. "\n")
        file:write(ts .. " " .. h .. " [**] Decoded AZORult response - Key: " .. (hexkey) .. "
Response: " .. hex_esc(b) .. " [**] " .. srcip .. ":" .. sp .. " -> " ..  dstip .. ":" .. dp .. "\n")
        file:flush()

        data = string.match(b, "^<c>([A-Za-z0-9+/]+=*)</c>")
        if not data then
            -- header might be truncated and missing "</c>" end
            data = string.match(b, "^<c>([A-Za-z0-9+/]+=*)$")
        end

        if data then
            data = data .. string.rep("=", (3 - ((#data + 3) % 4)))
            decoded = base64_d(data)
            file:write(ts .. " " .. h .. " [**] Decoded AZORult header - Key: " .. (hexkey) .. "
Header: " .. hex_esc(decoded) .. " [**] " .. srcip .. ":" .. sp .. " -> " ..  dstip .. ":" .. dp ..
"\n")
            file:flush()
        end
    end
end
```

suricata.yaml

```
# Configure the type of alert (and other) logging you would like.
outputs:
...
  # Lua Output Support - execute lua script to generate alert and event
  # output.
  # Documented at:
  # https://suricata.readthedocs.io/en/latest/output/lua-output.html
  - lua:
      enabled: yes
      #scripts-dir: /etc/suricata/lua-output/
      scripts:
      - suri-log-azorult.lua
```

Lua Logging – azorult.log

```
10/12/2018-21:35:24.572822 bitdotz[.]top [**] Found possible AZORult POST : POST /index.php HTTP/1.0
[**] 10.10.12.102:49185 -> 5.101.67[.]143:80
10/12/2018-21:35:24.572822 bitdotz[.]top [**] Decoded AZORult request - Key: \x03\x55\xae Request:
%34%32F%30%33%34%38E%2D%36%39BD%30%38%30%30%2D%33%35%34E%30C%38%34%2D%32%30%33%39BC%38D%2D%35%35%36%3
1%31%32AC [**] 10.10.12.102:49185 -> 5.101.67[.]143:80
10/12/2018-21:35:24.572822 bitdotz[.]top [**] Decoded AZORult response - Key: \x03\x55\xae Response:
<c>LSsrKy0tLS0rLQ0KRglwYXNzd29yZHMJJVVTRVJQUk9GSUxFJVwJKnBhc3MqLnR4dCwqcHdkKi50eHQsKmlkKi50eHQsKmJ0Yy
oudHh0LCpjb2luKi50eHQsKndhbGwqLnR4dCwqbG9naW4qLnR4dAkxMDAJLQkrCQ0KTAlodHRwOi8vc2FmaS5jby56YS93aW5udHg
uZXhlCSsJKg0KSQkxMDQuMjQ1LjE0NS4yNTpDQQ0K</c><n>\xa9\x15Y,\xa5\x16\x1dv\xa1\x0b\x1db\xa7\x17U,\xab\x0
a^r\xa7\x09U,\xa4T\x1d0\xe5U\x1ee\xa4\x09\x0aL\x92\xf50\x02\xc8e0\x05\xc8e0\xfe7e0\xb9\xc8e0\x01\xc8e
0A\xc8e0\x01\xc8e0\x01\xc8e0\x01\xc8e0\x01\xc8e0\x01\xc8e0\x01\xc8e0\x01\xc8e0\x01\xc8e0\xb9\xc8e0\x0
f\xd7\xdf>\x01|l\xfd … [**] 10.10.12.102:49185 -> 5.101.67[.]143:80
10/12/2018-21:35:24.572822 bitdotz[.]top [**] Decoded AZORult header - Key: \x03\x55\xae Header: -
+++----+-
\x0d\x0aF\x09passwords\x09%USERPROFILE%\\x09*pass*.txt,*pwd*.txt,*id*.txt,*btc*.txt,*coin*.txt,*wall*
.txt,*login*.txt\x09100\x09-
\x09+\x09\x0d\x0aL\x09hxxp://safi[.]co[.]za/winntx.exe\x09+\x09*\x0d\x0aI\x09104.245.145.25:CA\x0d\x0
a [**] 10.10.12.102:49185 -> 5.101.67[.]143:80
```

AZORult – Lua Script (logging version)

```
function common(a, d, verbose)

    if #a < 256 then
        return 0
    end

    -- cleartext starts "<c>" followed by Base64 string so deduce 3-byte XOR key
    key = {bit.bxor(a:byte(1), 0x3c), bit.bxor(a:byte(2), 0x63), bit.bxor(a:byte(3), 0x3e)}

    -- decode first 16 bytes after "<c>" and make sure they're valid Base64
    b = "<c>"
    for i = 4, 20, 1 do
       b = b .. string.char(bit.bxor(a:byte(i), key[((i - 1) % 3) + 1]))
    end

    data = string.match(b, "^<c>[A-Za-z0-9+/]*$")
```

AZORult – Lua Script (logging version)

```
    if data then
        hexkey = string.format("\\x%02x\\x%02x\\x%02x", key[1], key[2], key[3])
        if verbose==1 then
            print("Decoded AZORult XOR key: " .. hexkey)
        else
            ScFlowvarSet("CnC/AZORult/decoded/xor_key", #"CnC/AZORult/decoded/xor_key", hexkey,
#hexkey)
        end

        for i = 21, 768, 1 do
           b = b .. string.char(bit.bxor(a:byte(i), key[((i - 1) % 3) + 1]))
        end

        decoded = hex_esc(b)
        if verbose==1 then
            print("Decoded AZORult : " .. decoded)
        else
            ScFlowvarSet("CnC/AZORult/decoded/download", #"CnC/AZORult/decoded/download", decoded,
#decoded)
        end
```

AZORult – Lua Script (logging version)

```
        data = string.match(b, "^<c>([A-Za-z0-9+/]+=*)</c>")
        if not data then
            -- header might be truncated and missing "</c>" end
            data = string.match(b, "^<c>([A-Za-z0-9+/]+=*)$")
        end

        if data then
            data = data .. string.rep("=", (3 - ((#data + 3) % 4)))

            decoded = hex_esc(base64_d(data))
            if verbose==1 then
                print("Decoded AZORult header : " .. decoded)
            else
                ScFlowvarSet("CnC/AZORult/decoded/header", #"CnC/AZORult/decoded/header", decoded,
#decoded)
            end
        end
```

AZORult – Lua Script (logging version)

```
        e = ""
        for i = 1, #d, 1 do
           e = e .. string.char(bit.bxor(d:byte(i), key[((i - 1) % 3) + 1]))
        end

        decoded = hex_esc(e)
        if verbose==1 then
            print("Decoded AZORult request : " .. decoded)
        else
            ScFlowvarSet("CnC/AZOrult/decoded/request", #"CnC/AZOrult/decoded/request", decoded,
#decoded)
        end

        return 1

    end
    return 0
end
```

suricata.yaml

```
# Configure the type of alert (and other) logging you would like.
outputs:
...
  # Extensible Event Format (nicknamed EVE) event log in JSON format
  - eve-log:
      enabled: yes
      filetype: regular #regular|syslog|unix_dgram|unix_stream|redis
      filename: eve.json
...
      types:
...
        # Metadata event type. Triggered whenever a pktvar is saved
        # and will include the pktvars, flowvars, flowbits and
        # flowints.
        - metadata
```

Lua Logging – (parsed) eve-json.log


```
"CnC/AZORult/decoded/xor_key": "\\x03\\x55\\xae"
"CnC/AZORult/decoded/download":
"<c>LSsrKy0tLS0rLQ0KRglwYXNzd29yZHMJJVVTRVJQUk9GSUxFJVwJKnBhc3MqLnR4dCwqcHdkKi50eHQsKmlkKi50eHQsKmJ0Y
youdHh0LCpjb2luKi50eHQsKndhbGwqLnR4dCwqbG9naW4qLnR4dAkxMDAJLQkrCQ0KTAlodHRwOi8vc2FmaS5jby56YS93aW5udH
guZXhlCSsJKg0KSQkxMDQuMjQ1LjE0NS4yNTpDQQ0K</c><n>\\xa9\\x15Y,\\xa5\\x16\\x1dv\\xa1\\x0b\\x1db\\xa7\\x
17U,\\xab\\x0a^r\\xa7\\x09U,\\xa4T\\x1d0\\xe5U\\x1ee\\xa4\\x09\\x0aL\\x92\\xf50\\x02\\xc8e0\\x05\\xc8
e0\\xfe7e0\\xb9\\xc8e0\\x01\\xc8e0A\\xc8e0\\x01\\xc8e0\\x01\\xc8e0\\x01\\xc8e0\\x01\\xc8e0\\x01\\xc8e
0\\x01\\xc8e0\\x01\\xc8e0\\x01\\xc8e0\\xb9\\xc8e0\\x0f\\xd7\\xdf>\\x01|l\\xfd …
"CnC/AZORult/decoded/request":
"%34%32F%30%33%34%38E%2D%36%39BD%30%38%30%30%2D%33%35%34E%30C%38%34%2D%32%30%33%39BC%38D%2D%35%35%36%
31%31%32AC"
"CnC/AZORult/decoded/header": "-+++----+-
\\x0d\\x0aF\\x09passwords\\x09%USERPROFILE%\\\\x09*pass*.txt,*pwd*.txt,*id*.txt,*btc*.txt,*coin*.txt,
*wall*.txt,*login*.txt\\x09100\\x09-
\\x09+\\x09\\x0d\\x0aL\\x09hxxp://safi[.]co[.]za/winntx.exe\\x09+\\x09*\\x0d\\x0aI\\x09104.245.145.25
:CA\\x0d\\x0a"
```

Potential Issues – Use of libraries

- Sometimes needed, especially for things like cryptography and compression algorithms which can be very cumbersome to implement
  - RC4 and XTEA are easy – Serpent and AES are not …
- What can we assume is installed on the users' Suricata system?
- Perhaps use well-known distro such as Ubuntu LTS versions
- Still need to tell users which packages need to be installed
- Is it reasonable to expect users to compile their own?
- Need to be careful about thread-safety

Lua - A few gotchas

- Indices are counted from 1 not 0
- Regular expressions are limited
  - e.g. no “/(…){n, m}/”
- Integer arithmetic is not 32-bit
  - May need extra library
- “\x22” hex code string escapes don’t work in version 5.1
  - Use “\34” decimal code instead

Suggested Uses

- Flowvar capture
  - Could use scripts as additions to existing rules (always returning “1” if we don’t wish to change the detection logic)
  - Easy to add to existing Lua detection scripts too
  - PCRE can also be used to return flowvars
- Lua Output
  - Useful for automated analysis of stored PCAPs (e.g. as part of sandbox analyses)
  - E.g. extract additional files for re-submission to the sandbox
    - Particularly decoded Windows executables

Other Targets

- Formbook
- Loki Bot
- Ursnif /ISFB
  - (given list of known Serpent cryptographic keys to try)
- Danabot

References/Further Reading

- Quote about Lua from a programming language expert
  https://stackoverflow.com/questions/2100902/is-lua-based-primarily-on-well-established-programming-language-ideas
- AZORult PCAP used in the example analysis
  http://malware-traffic-analysis.net/2018/10/12/index.html
  “2018-10-12-Hookads-campaign-Fallout-EK-pcaps.zip” (second file in this zip)
- Any.Run - public submissions (filter on "azorult")
  https://app.any.run/submissions

References/Further Reading

- Proofpoint Threat Insight Blog

  https://www.proofpoint.com/us/threat-insight/post/threat-actors-using-legitimate-paypal-accounts-to-distribute-chthonic-banking-trojan

  https://www.proofpoint.com/us/threat-insight/post/new-version-azorult-stealer-improves-loading-features-spreads-alongside

- More blogs linked in MalPedia

  https://malpedia.caad.fkie.fraunhofer.de/details/win.azorult

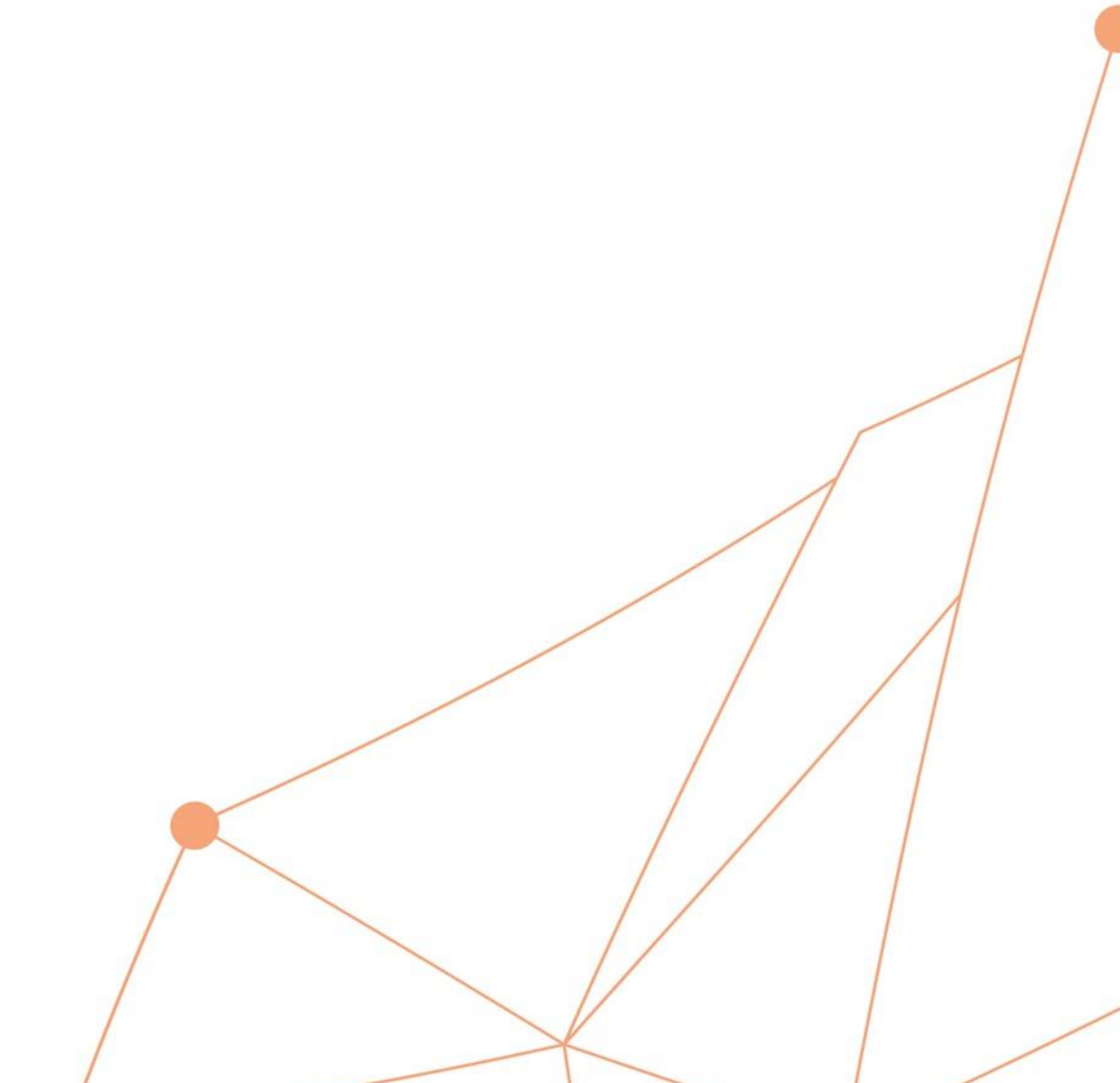
- Another example of Lua Output for automated analysis (I'm not alone ☺)

  https://www.trustwave.com/Resources/SpiderLabs-Blog/Decoding-Hancitor-Malware-with-Suricata-and-Lua/

proofpoint®

Addendum

If there's time …



Formbook – Lua Script (logging version)

```
local bit = require("bit")
local gcrypt = require("luagcrypt")

gcrypt.check_version()

function init(args)
    local needs = {}
    needs["http.uri"] = tostring(true)
    return needs
end

function to_hex(s)
    local hex = string.gsub(s, ".", function(c)
        return string.format("%02x", string.byte(c))
    end)
    return hex
end

function sha1(s)
    local hash = gcrypt.Hash(gcrypt.MD_SHA1)
    hash:write(s)
    return hash:read()
end
```

Formbook – Lua Script (logging version)

```
function le_to_be(k)
    local i, j
    k2 = ""
    for i = 1, #k, 4 do
        for j = 3, 0, -1 do
             k2 = k2 .. string.char(string.byte(k, i + j))
        end
    end
    return k2
end

function base64_d(a)
-- requires a to be a valid base64 string
    local b, i, keyStr
    keyStr = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/="
    b = ""
    for i = 1, #a, 4 do
    …
    end
    return b
end
```

Formbook – Lua Script (logging version)

```
function rc4(s, K)
  local S={}, i, t, m, x

  x = ""
  for i = 0, 255, 1 do
    S[i + 1] = i
  end

  t, m = 0, 0
  for i = 0, 255, 1 do
    m = (m + K[t + 1] + S[i + 1]) % 256
    S[i + 1], S[m + 1] = S[m + 1], S[i + 1]
    t = (t + 1) % #K
  end

  t, m = 0, 0
  for i =1, #s, 1 do
    t = (t + 1) % 256
    m = (m + S[t + 1]) % 256
    S[t + 1], S[m + 1] = S[m + 1], S[t + 1]
    x = x .. string.char(bit.bxor(string.byte(s, i), S[((S[t + 1] + S[m + 1]) % 256) + 1]))
  end
  return x
end
```

Formbook – Lua Script (logging version)

```
function common(u, h, verbose)
    if verbose == 1 then print("Checking URI " .. u .. " and Host " .. h) end
    c2_path = string.match(u,"^(.*/)[^/]*%?[A-Za-z0-9_-]+=");
    data = string.match(u,"%?[A-Za-z0-9_-]+=([A-Za-z0-9+/_-]+[.=]*)")
    if data then
        data = string.gsub(data, "_", "/")
        data = string.gsub(data, "-", "+")
        data = string.gsub(data, "%.", "=")
        data = data .. string.rep("=", (3 - ((#data + 3) % 4)))
        data_bin = base64_d(data)
        key = le_to_be(sha1(h .. c2_path))
        K = {string.byte(key, 1, #key)}
        decoded = rc4(data_bin, K)
        if #decoded > 10 and string.find(decoded, "^[%w%p%s]+$") then
            if verbose == 1 then
                print("Decoded Formbook checkin " .. decoded .. " from URI")
            else
                ScFlowvarSet("CnC/formbook/decoded/checkin", #"CnC/formbook/decoded/checkin",
decoded, #decoded)
            end
            return 1
        end
    end
    return 0
end
```

Formbook – Lua Script (logging version)

```
-- return match via table
function match(args)
    local u = tostring(args["http.uri"])
    local h = HttpGetRequestHost()
    if h == nil then h = "<hostname unknown>" end
    return common(u, h, 0)
end

function run()
    local h = arg[1]
    local u = arg[2]
    common(u, h, 1)
end
```

Lua Logging – (parsed) eve-json.log

- Formbook PCAP
  http://malware-traffic-analysis.net/2018/02/26/index2.html


```
"CnC/formbook/decoded/checkin": "FBNG:502F93453.8:Windows 7 Home Premium x86:Y2hhcmxlbmUucGFya3M="

"CnC/formbook/decoded/post": "\\x0d\\x00\\x0a\\x00O\\x00u\\x00t\\x00l\\x00o\\x00o\\x00k\\x00
\\x00R\\x00e\\x00c\\x00o\\x00v\\x00e\\x00r\\x00y\\x00\\x0d\\x00\\x0a\\x00c\\x00l\\x00s\\x00i\\x00d\\x
00\\x09\\x00{\\x00E\\x00D\\x004\\x007\\x005\\x004\\x001\\x004\\x00-\\x00B\\x000\\x00D\\x006\\x00-
\\x001\\x001\\x00D\\x002\\x00-\\x008\\x00C\\x003\\x00B\\x00-
\\x000\\x000\\x001\\x000\\x004\\x00B\\x002\\x00A\\x006\\x006\\x007\\x006\\x00}\\x00“
"CnC/formbook/decoded/post":
"FBIMG502F9345\\xff\\xd8\\xff\\xe0\\x00\\x10JFIF\\x00\\x01\\x01\\x01\\x00`\\x00`\\x00\\x00\\xff\\xdb\
\\x00C\\x00\\x08\\x06\\x06\\x07\\x06\\x05\\x08\\x07\\x07\\x07\\x09\\x09\\x08\\x0a\\x0c\\x14\\x0d\\x0c\
\\x0b\\x0b\\x0c\\x19\\x12\\x13\\x0f\\x14\\x1d\\x1a\\x1f\\x1e\\x1d\\x1a\\x1c\\x1c $.'
\",#\\x1c\\x1c(7),01444\\x1f'9=82<.342\\xff\\xdb\\x00C\\x01\\x09\\x09\\x09\\x0c\\x0b\\x0c\\x18\\x0d\\
x0d\\x182!\\x1c!2222222"
```