Lobshot

research.openanalysis.net/lobshot/bot/hvnc/triage/2023/07/16/lobshot.html

OALABS Research July 16, 2023

Background

LOBSHOT is the internal name given to an hVNC based bot by the Elastic Security Labs team. The public name of this malware is currently unknown. The primary capability of the malware is acting as both a stealer and hidden VNC.

References

Elastic Security Labs discovers the LOBSHOT malware

Samples

e4ea88887753a936eaf3361dcc00380b88b0c210dcbde24f8f7ce27991856bf6

Analysis

DarkVNC Overlap

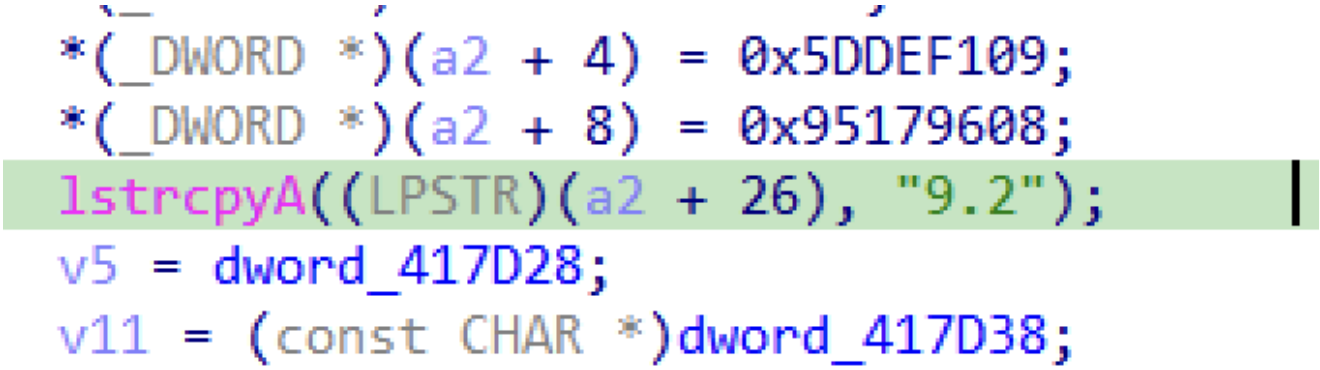
On VT the sample is being flagged as "DarkVNC" by Kaspersy this is possibly just an overlap with hVNC code from DarkVNC.

C2 Request

According to Elastic the C2 request contains some hardcoded data that can be used as a signature to hunt for the samples.

> Searching for the above mov instruction paired with the first DWORD of the hardcoded value **(C7 06 25 56 0A DC)** shows over 550 samples in VirusTotal within the last year. With some of the first samples showing up in late July 2022. The prevalence of these hardcoded values shows that it has been actively used and under development for a long period of time, and will likely continue to be used in the future.

There appears to also be a version number at the top of this struct



Config Extractor

This has been copied from the original Elastic Github

```
# Copyright Elasticsearch B.V. and/or licensed to Elasticsearch B.V. under
# one or more contributor license agreements. Licensed under the Elastic License 2.0;
# you may not use this file except in compliance with the Elastic License 2.0.
```

Notes

- The original sample used a convoluted way to obtain a static seed value used for the string decryption in other versions they realized this was silly and just hard coded a value
- There are different seed values for different variants

```
import pefile
import re

IP_ADDRESS_REGEX = r"^([0-9]{1,3}\.){3}[0-9]{1,3}$"
PORT_REGEX = r"^[0-9]{1,5}$"


def is_ascii(s):
    return all((c < 128 and c > 39) for c in s)



def perform_extraction_file(file_path):
    rdata = parse_file(file_path)
    if not rdata:
        raise RuntimeError(".rdata section not found")

    candidates = generate_candidates(rdata)
    strings = get_encrypted_strings(candidates)
    #print(strings)
    for s in strings:
        if re.search(IP_ADDRESS_REGEX, s):
            result_ip = s
        if re.search(PORT_REGEX, s):
            result_port = int(s)
    if not result_ip:
        raise RuntimeError("Configuration unsuccessful, could not extract IP/Port
\n")
    print(f"{result_ip}:{result_port}")


def string_decryption(encrypted_data, seed):
    buffer = bytearray([0 for _ in range(len(encrypted_data) // 2)])

    flag = False
    z = 0
    index = 0
    index2 = 2

    for i, x in enumerate(encrypted_data):
        try:
            y = encrypted_data[index + 1] - 0x61
            index += 2

            if flag:
                buffer[i] = seed ^ z ^ (y | (16 * encrypted_data[index2] - 16)) &
0xFF
                index2 += 2
            else:
                flag = True
                z = y | (16 * (x - 1)) & 0xFF
        except (IndexError, ValueError):
```

```
            continue

    buffer = buffer[1:]
    return buffer


def parse_file(file_path):
    with open(file_path, "rb") as data:
        data = data.read()
        pe = pefile.PE(data=data)
        for section in pe.sections:
            if b"rdata" in section.Name:
                rdata = section.get_data()
                return rdata
        else:
            print(".rdata section not found in {}\n".format(file_path.name))
            return None


def generate_candidates(section_data):
    candidates: list[bytes] = list()
    blocks = section_data.split(b"\x00")
    blocks = [x for x in blocks if x != b""]
    for block in blocks:
        if len(block) > 3 and not b"\\" in block:
            candidates.append(block)
    return candidates


def get_encrypted_strings(candidates):
    string_table = []
    seeds = 'abcdefghijklmnopqrstuvwxyz'
    seeds += seeds.upper()
    for s in seeds:
        seed = ord(s)
        tmp_out = []
        for string in candidates:
            decrypted_string = string_decryption(string, seed)
            decrypted_string = decrypted_string.replace(b'\x00', b'')
            if len(decrypted_string) < 3 or not is_ascii(decrypted_string):
                continue
            tmp_out.append(decrypted_string.decode('utf-8'))
        if 'connect' in tmp_out or 'msedge.exe' in tmp_out or 'kernel32.dll' in
tmp_out:
            string_table = tmp_out
            break
    return string_table


def decrypt_candidates(candidates):
    result_ip = None
    result_port = None
```

```
    for string in candidates:
        decrypted_string = string_decryption(string, ord('S'))
        print(decrypted_string)
        if re.search(IP_ADDRESS_REGEX, decrypted_string):
            result_ip = decrypted_string
        if re.search(PORT_REGEX, decrypted_string):
            result_port = int(decrypted_string)
    if not result_ip:
        raise RuntimeError("Configuration unsuccessful, could not extract IP/Port
\n")

    return result_ip.decode('utf-8'), result_port


def display_results(result):
    print("FILE: {}".format(result[2].name))
    print("IP: {}".format(result[0].decode("utf-8")))
    print("Port: {}\n".format(result[1]))


FILE_PATH = '/tmp/e4ea88887753a936eaf3361dcc00380b88b0c210dcbde24f8f7ce27991856bf6'
perform_extraction_file(FILE_PATH)

95.217.125.200:443

chr(0x53)

'S'

import os
# assign directory
directory = '/tmp/lobshots'

# iterate over files in
# that directory
for filename in os.listdir(directory):
    f = os.path.join(directory, filename)
    # checking if it is a file
    if os.path.isfile(f):
        try:
            print(f)
            perform_extraction_file(f)
        except:
            print('\t fail!')
```

/tmp/lobshots/1560ff3abacb50dc796f76eada2f8d8c020fa3e4a1f57b806029f44fca5682ae
193.149.176.112:443
/tmp/lobshots/b937eb4c88479119655b956f5be7e96c87fee72fa1d705028fe915719c7e8718
168.100.10.88:443
/tmp/lobshots/e14ee6302076a2bb9e5634407500757319d5de9c45305ec6269120b7283b24cf
64.190.113.123:443
/tmp/lobshots/7195799e2783d90b4f1ab326207c286064ca3557eb68a0dd04e65760781ba6bf
64.52.80.210:443
/tmp/lobshots/92dc319c346eccbffd091e87aef346751bf2af21d104010f1658639d7cd7b09f
64.52.80.210:443
/tmp/lobshots/61221cd73093dc61d224f979b680433d33f4cca6e14be6a79d71028b778d3668
64.52.80.210:443
/tmp/lobshots/0100d00ec16886288a5acd559dc055052578a607a265eaa3bf349c1140a00489
64.52.80.210:443
/tmp/lobshots/2db2248ee80a26252576c2463f11ced5e6fa855cb8a5c929af135f45d9fbd862
146.70.86.228:443
/tmp/lobshots/88502f27ab03c34af7ceda2bb6fecda42ae227e74e8a5e52346db749e200d134
91.235.136.155:443
/tmp/lobshots/48f61315302ba4cac52f949a173981982d22bebc367866b929461f19738927d9
64.52.80.210:443
/tmp/lobshots/8cbae065a70d589fb78723a2c7dc5e903694196b82cea9ee7a53bac9750d5ad1
64.52.80.210:443
/tmp/lobshots/5532ea44f1f30f2e8a36e8d0ab5da7f9f5bfb15cd25a5100b8810490a2482039
66.85.74.142:443
/tmp/lobshots/8309535e526cf7eab791082b6f1ae6961739d2826779f9e920c3ed684c6a28f1
179.43.162.5:443
/tmp/lobshots/85364b9503ec20a8801bfd20db136c072d4ab7506f8afca81faf4b72567ee43d
64.190.113.123:443
/tmp/lobshots/77767acf212e727f9c1c9a07535980ce37d6e9a2c1431cfde81bf3d9eaf6fb05
64.190.113.123:443
/tmp/lobshots/ae221670729038f92398b7fe4e08928ea6ebc0c1d006c63c8a3bac2e30770c2b
77.91.84.186:443
/tmp/lobshots/7e0e7def3be7f9bc7793bf155935d16a9db631dd99b71ab51295338315129cce
64.190.113.123:443
/tmp/lobshots/921db56e4de5605b3759de43727f62be0f4c158a2837cf08ff376c427b85bec8
193.149.180.212:443
/tmp/lobshots/2c89fb34f8dd0094d988bd5460596141209e2b6afc15ef0288d8128cad70cc2e
91.103.252.54:443
/tmp/lobshots/17291fc45dc342a6ce4309dd4b77cd094c564fad14c212bca827770daf7207d4
64.52.80.210:443
/tmp/lobshots/15ec54cd2b2605ec8395645fe545204a89ddfe6fef656c98c0578006184d0228
168.100.8.124:443
/tmp/lobshots/22011ef253be77a08e930c391e61afd9234435216d3ff451b71834661e4858ed
45.61.136.124:443
/tmp/lobshots/e4ea88887753a936eaf3361dcc00380b88b0c210dcbde24f8f7ce27991856bf6
95.217.125.200:443
/tmp/lobshots/5ab55b3fb492e568db48cd0eac2770621d99f1f72cfe317f6601cd2447074b4f
193.149.176.112:443
/tmp/lobshots/209270bcd107207466f516789d4bfe677694cda5614e4abebaa26f2a827b293c
146.70.86.228:443