Technical Analysis of Zloader Updates

ThreatLabz ⋮ ⋮ 9/21/2025

Technical Analysis

In this section, we will explore the various changes that were introduced in the latest versions of Zloader including new evasion techniques, additional functionality for lateral movement, and modifications to network communication.

Anti-analysis

One notable change to Zloader's functionality involves the required filename that was expected by the malware. Previously, Zloader samples were expected to be run with a specific hardcoded filename. If the actual filename did not match the expected value, that Zloader sample would not run. This design is likely intended to evade automated malware sandbox environments. However, in the most recent versions, the

malware Zloader author introduced two new generic filenames to allow the threat actors that deploy (or update) Zloader with more flexibility. These two generic filenames are `Updater.exe` and `Updater.dll`.

Another significant change that was made to hinder analysis is more obfuscation layers. This level of obfuscation is achieved using different XOR-based integer decoding functions. To simplify the analysis, ThreatLabz used an IDA script to remove these layers of obfuscation as shown in the example below.

```
import idautils

XOR_KEY = 0xAE # CHANGE ACCORDINGLY
FUNCTION_NAME = "Calculate_Int1" # CHANGE ACCORDINGLY

# Iterate through all functions in the IDA database.
for func_addr in Functions():
    func_name = get_func_name(func_addr)
    if func_name.startswith(FUNCTION_NAME):
        print(f"Processing function: {func_name}")

        # Search for cross-references (xrefs) to the function.
        for xref in idautils.XrefsTo(func_addr):
            print(f"\tFound xref at: {hex(xref.frm)}")

            # Grab the DWORD passed and perform a XOR operation on it.
            param = ida_bytes.get_byte(xref.frm-1) # CHANGE ACCORDINGLY
            result = param ^ XOR_KEY
            mov_eax_constant = b'\xB8' + result.to_bytes(4, 'little')
            ida_bytes.patch_bytes(xref.frm, mov_eax_constant)
            set_cmt(xref.frm, FUNCTION_NAME, 0)
```

The figure below illustrates a function that checks Zloader's process integrity level, before and after deobfuscation.

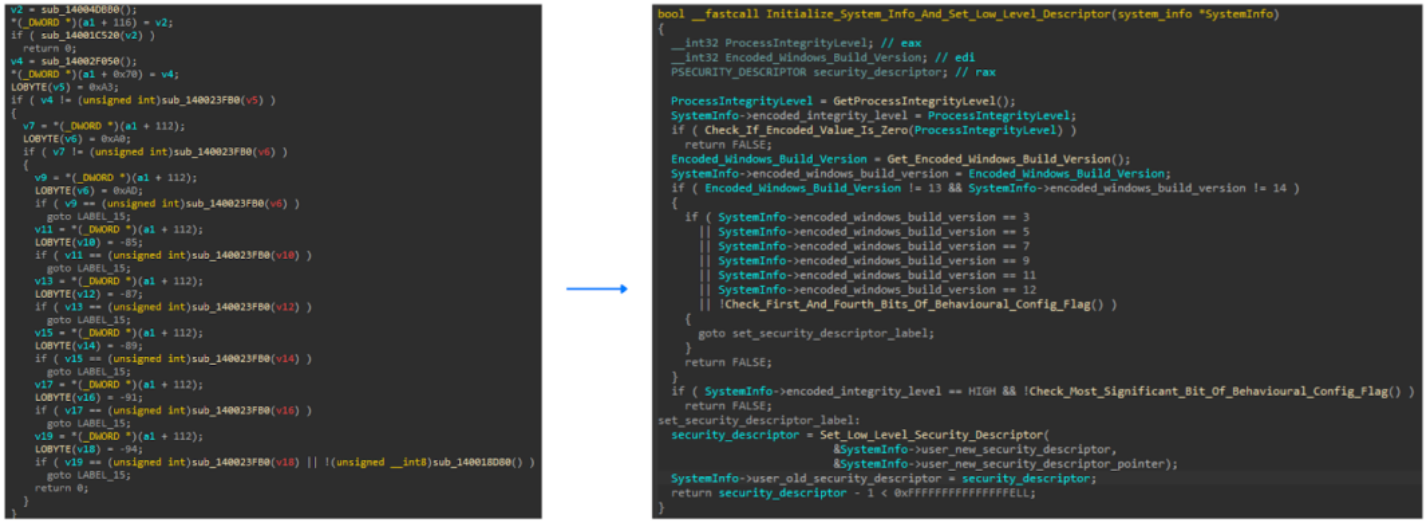

Figure 1: Example of Zloader's new code obfuscation techniques and the same function after deobfuscation.

The process integrity level is important because Zloader will exit if it detects that the process is being executed with high integrity. In modern versions of Windows, most standard processes run with medium integrity. Thus, this new integrity level check is likely another detection mechanism for malware sandboxes, which often run samples with administrator privileges (i.e., high integrity). If Zloader is executed with medium integrity, the malware will be installed in the `%APPDATA%` directory. Otherwise, if Zloader has system integrity, the malware will be installed in the `%PROGRAMFILES%` directory.

The typical integrity levels are shown in the table below:

Integrity Level	Description
Low integrity (SID value: 0x1000)	Restricted processes, usually sandboxed (e.g., web browsers like Chrome/Edge running untrusted content)
Medium integrity (SID value: 0x2000)	Standard user processes
High integrity (SID value: 0x3000)	Administrator privileges
System integrity (SID value: 0x4000)	Processes running as part of the OS kernel or critical system operations (e.g., trusted installers, system services)

Table 1: Summary of Windows process integrity levels.

This behavior stands out because user-mode trojans like Zloader typically require elevated privileges to perform various actions. By avoiding elevated permissions, Zloader sacrifices broader system access for the added benefit of evading malware sandbox detection.

Static configuration

The Zloader static configuration has also undergone minor changes. The TLS Server Name Indication (SNI) and the DNS nameserver, which functions as the command-and-control (C2 server) for Zloader's network communication when using the DNS Tunneling protocol, have been relocated to the end of the C2 domain section.

The DNS servers used for resolving the C2 nameserver were previously stored in network byte order. The DNS servers are now represented using a mini JSON configuration. A description for each JSON key is shown in the table below:

Configuration key	Description
`proto`	Indicates the communication protocol used, such as UDP (DNS), HTTPS (DoH), or TLS (DoT).
`ip`	The resolver IP.
`port`	The resolver port.
`qps`	(Queries Per Second) Indicates the maximum number of DNS queries the resolver can process per second.

Table 2: Mini JSON configuration for the DNS servers used by Zloader's DNS Tunneling protocol.

If a DNS entry equals 127.0.0.1, Zloader ignores the entry and treats it as a placeholder.

The figure below shows the modified static configuration, including the new location of the C2 domains, the mini JSON format, and a placeholder entry for an additional DNS server.

```
00000000  00 00 00 00 53 6d 74 31  00 00 00 00 00 00 00 00  |....Smt1........|
00000010  00 00 00 00 00 00 00 00  00 4d 32 00 00 00 00 00  |.........M2.....|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 68 74  |..............ht|
00000030  74 70 73 3a 2f 2f 61 64  73 6d 61 72 6b 73 2e 63  |tps://adsmarks.c|
00000040  6f 6d 2f 00 00 00 00 00  00 00 00 00 00 00 00 00  |om/.............|
00000050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 68  |...............h|
00000070  74 74 70 73 3a 2f 2f 61  64 73 65 6d 61 69 6c 2e  |ttps://adsemail.|
00000080  63 6f 6d 2f 00 00 00 00  00 00 00 00 00 00 00 00  |com/............|
00000090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000000b0  68 74 74 70 3a 2f 2f 66  6f 72 64 6e 73 2f 63 6f  |http://fordns/co|
000000c0  72 70 72 6f 6f 74 2f 20  7e 20 64 6e 73 3a 2f 2f  |rproot/ ~ dns://|
000000d0  64 74 31 2e 61 75 74 6f  6d 6f 74 6f 73 70 6f 72  |dt1.automotospor|
000000e0  74 2e 6e 65 74 00 00 00  00 00 00 00 00 00 00 00  |t.net...........|
000000f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000002b0  00 00 00 00 00 00 00 00  03 00 00 00 31 00 00 00  |............1...|
000002c0  2d 2d 2d 2d 2d 42 45 47  49 4e 20 50 55 42 4c 49  |-----BEGIN PUBLI|
000002d0  43 20 4b 45 59 2d 2d 2d  2d 2d 0a 4d 49 47 66 4d  |C KEY-----.MIGfM|
000002e0  41 30 47 43 53 71 47 53  49 62 33 44 51 45 42 41  |A0GCSqGSIb3DQEBA|
000002f0  51 55 41 41 34 47 4e 41  44 43 42 69 51 4b 42 67  |QUAA4GNADCBiQKBg|
00000300  51 44 43 59 2b 55 46 74  76 55 35 63 6c 74 47 70  |QDCY+UFtvU5cltGp|
00000310  43 45 35 45 46 6c 2b 48  66 62 33 0a 53 38 37 73  |CE5EFl+Hfb3.S87s|
00000320  74 43 4a 64 48 68 53 36  74 75 79 79 61 59 6a 4f  |tCJdHhS6tuyyaYjO|
00000330  74 37 78 49 41 56 33 6b  46 63 36 42 6c 57 78 6b  |t7xIAV3kFc6BlWxk|
00000340  6d 4f 6d 6e 54 57 64 30  71 74 37 47 54 30 6f 2b  |mOmnTWd0qt7GT0o+|
00000350  74 44 32 75 54 66 37 7a  50 66 52 33 0a 74 6b 6d  |tD2uTf7zPfR3.tkm|
00000360  70 33 76 47 58 79 4e 5a  58 6a 52 39 30 6c 77 53  |p3vGXyNZXjR90lwS|
00000370  48 4b 73 32 32 6b 73 66  4f 67 6d 5a 70 4e 64 62  |HKs22ksfOgmZpNdb|
00000380  5a 2b 5a 48 56 6e 34 6f  7a 62 70 45 2f 63 47 58  |Z+ZHVn4ozbpE/cGX|
00000390  7a 7a 6f 2f 6b 39 33 7a  2b 50 36 4a 6b 0a 63 68  |zzo/k93z+P6Jk.ch|
000003a0  58 5a 38 4e 77 46 5a 4d  38 41 52 72 63 6a 65 51  |XZ8NwFZM8ARrcjeQ|
000003b0  49 44 41 51 41 42 0a 2d  2d 2d 2d 2d 45 4e 44 20  |IDAQAB.-----END |
000003c0  50 55 42 4c 49 43 20 4b  45 59 2d 2d 2d 2d 2d 0a  |PUBLIC KEY-----.|
000003d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000004b0  00 00 00 00 5b 7b 22 70  72 6f 74 6f 22 3a 22 75  |....[{"proto":"u|
000004c0  64 70 22 2c 22 69 70 22  3a 22 38 2e 38 2e 38 2e  |dp","ip":"8.8.8.|
000004d0  38 22 2c 22 70 6f 72 74  22 3a 35 33 2c 22 71 70  |8","port":53,"qp|
000004e0  73 22 3a 31 30 30 7d 2c  7b 22 70 72 6f 74 6f 22  |s":100},{"proto"|
000004f0  3a 22 75 64 70 22 2c 22  69 70 22 3a 22 31 32 37  |:"udp","ip":"127|
00000500  2e 30 2e 30 2e 31 22 2c  22 70 6f 72 74 22 3a 30  |.0.0.1","port":0|
00000510  2c 22 71 70 73 22 3a 31  30 7d 5d 00 00 00 00 00  |,"qps":10}].....|
00000520  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000744
```

zscaler | ThreatLabz

Figure 2: Zloader's new static configuration format.

Shell commands

Zloader's interactive shell commands allow a threat actor to execute commands, deploy second-stage malware payloads, run shellcode, exfiltrate data, as well as identify and terminate specific processes. The latest version of Zloader adds a new set of LDAP functions to improve network discovery and expand lateral movement capabilities. The new functions are outlined in the table below.

Command	Description
`ldap_bind_s`	Authenticates and binds to the LDAP server.
`ldap_err2string`	Converts an LDAP error code into a human-readable string.
`ldap_first_attribute`	Retrieves the first attribute of an LDAP entry.
`ldap_first_entry`	Retrieves the first entry from an LDAP search result.
`ldap_get_values`	Retrieves the values associated with a specific attribute from an LDAP entry.
`ldap_init`	Initializes a connection to the LDAP server.
`ldap_memfree`	Releases allocated memory used by LDAP functions.
`ldap_next_attribute`	Retrieves the next attribute from an LDAP entry.
`ldap_next_entry`	Retrieves the next entry from an LDAP search result.
`ldap_search_s`	Performs a synchronous search on the LDAP server.
`ldap_set_option`	Sets various options for an LDAP session (e.g., timeout or protocol version).
`ldap_value_free`	Releases memory used for an array of attribute values.
`ldap_search`	Performs an asynchronous search on the LDAP server.

Table 3: New LDAP functions added to Zloader's interactive shell.

Network communication

The latest versions of Zloader have removed the Domain Generation Algorithm (DGA), which was rarely used in previous versions. In addition to this change, several other important updates have been introduced to Zloader's DNS tunnel encryption, together with new support for the WebSockets protocol. These updates are explored in the following sections.

DNS C2 traffic

The DNS C2 protocol, previously described in our Zloader 2.9.4.0 blog, has undergone significant changes in the latest iterations. In older versions, Zloader relied on TLS encryption for payloads in DNS queries and responses. However, the current implementation replaces this with Base32 encoding layered on top of a custom encryption algorithm. The comparison figure below highlights the differences between the old and new Zloader DNS C2 messages.

Old Zloader DNS tunnel queries

```
135 Standard query response 0x0001 A cdn.7768f00f0100000002000000000.0a000101.ns1.brownswer.com A 8.8.8.8
110 Standard query 0x0002 A cdn.7768f00f0200000003000000000.ns1.brownswer.com
126 Standard query response 0x0002 A cdn.7768f00f0200000003000000000.ns1.brownswer.com A 8.8.8.8
279 Standard query 0x0003 A cdn.7768f00f0300000004000300000.160303009d010000990303672db16382e1fc1943834fcfd029791c315afec8.8a1af42b54f4eadfaccfa0b600002ac02cc02bc030c02f009f009ec024c023...
271 Standard query 0x0004 A cdn.7768f00f0400000004cc03000000.35002f000a0100004600000000b000900000666f72646e73000a0008000600.1d00170018000b00020100000d00140012040105010201040305030203020...
295 Standard query response 0x0003 A cdn.7768f00f0300000004000300000.160303009d010000990303672db16382e1fc1943834fcfd029791c315afec8.8a1af42b54f4eadfaccfa0b600002ac02cc02bc030c02f009f009e...
287 Standard query response 0x0004 A cdn.7768f00f0400000004cc03000000.35002f000a0100004600000000b000900000666f72646e73000a0008000600.1d00170018000b00020100000d00140012040105010201040305030...
110 Standard query 0x0005 A cdn.7768f00f0500000005000000000.ns1.brownswer.com
126 Standard query response 0x0005 A cdn.7768f00f0500000005000000000.ns1.brownswer.com A 11.8.8.8
110 Standard query 0x0006 A cdn.7768f00f0600000005000000000.ns1.brownswer.com
126 Standard query response 0x0006 A cdn.7768f00f0600000005000000000.ns1.brownswer.com A 11.8.8.8
110 Standard query 0x0007 A cdn.7768f00f0700000005000000000.ns1.brownswer.com
126 Standard query response 0x0007 A cdn.7768f00f0700000005000000000.ns1.brownswer.com A 11.8.8.8
110 Standard query 0x0008 A cdn.7768f00f0800000005000000000.ns1.brownswer.com
126 Standard query response 0x0008 A cdn.7768f00f0800000005000000000.ns1.brownswer.com A 11.8.8.8
110 Standard query 0x0009 A cdn.7768f00f0900000005000000000.ns1.brownswer.com
126 Standard query response 0x0009 A cdn.7768f00f0900000005000000000.ns1.brownswer.com A 11.8.8.8
110 Standard query 0x000a A cdn.7768f00f0a00000005000000000.ns1.brownswer.com
126 Standard query response 0x000a A cdn.7768f00f0a00000005000000000.ns1.brownswer.com A 11.8.8.8
110 Standard query 0x000b A cdn.7768f00f0b00000005000000000.ns1.brownswer.com
126 Standard query response 0x000b A cdn.7768f00f0b00000005000000000.ns1.brownswer.com A 11.8.8.8
```

New Zloader DNS tunnel queries

```
133 Standard query response 0x0002 A cdn.2pbsvwi.dwdu3hyafe6gubzkhvvqiky.dt1.automotosport.net A 209.34.203.219
228 Standard query 0x0003 A cdn.c7spqsi.3elz7jwfxhvfdqv25bjmroy.sx33ubhgs2et5nokte62pt6dooa6roie.5ke4oyolwovtfjg7tmtpfg6gptr3dzyxuxjyamfi.whiympx4tget3kgjrzp4f6edhw65taza47yq.dt1.automoto...
230 Standard query 0x0004 A cdn.c7spqsi.3elz7jwcxhvfdqv25bjmroy.udli4jfoqpfgf5mk4zmixvece6wnfglzt3cj4mhrt2hcnpgtte2kzna.vtlyo75j3kpdjm4xtatlvxunh3clnjj3xhe5i5i.uplzwnfizltvxsvq.dt1.automo...
244 Standard query response 0x0003 A cdn.c7spqsi.3elz7jwfxhvfdqv25bjmroy.sx33ubhgs2et5nokte62pt6dooa6roie.5ke4oyolwovtfjg7tmtpfg6gptr3dzyxuxjyamfi.whiympx4tget3kgjrzp4f6edhw65taza47yq.dt1...
246 Standard query response 0x0004 A cdn.c7spqsi.3elz7jwcxhvfdqv25bjmroy.udli4jfoqpfgf5mk4zmixvece6wnfglzt3cj4mhrt2hcnpgtte2kzna.vtlyo75j3kpdjm4xtatlvxunh3clnjj3xhe5i5i.uplzwnfizltvxsvq.d...
117 Standard query 0x0005 A cdn.efxfksi.54ltamhrxfd4p452jdepnoy.dt1.automotosport.net
133 Standard query response 0x0005 A cdn.efxfksi.54ltamhrxfd4p452jdepnoy.dt1.automotosport.net A 65.93.102.41
239 Standard query 0x0006 A cdn.nlahrxi.usfr7gv7evvg3pzgnbxlwjy.usxjugg56yqela5t2wjjsrsowt4wkxsl5tlbdk3jyqjdje2tc5y5s.vsaxzhz4rxsejojspn2mxm6o2izxuxisj54rs.moynv2cjkyw6grexylx2f7uqla2ne.d...
233 Standard query 0x0007 A cdn.nlahrxi.usfr7gv6evvg3pzgnbxlwjy.a6fudnh7cizzz6ckenbf4cft7cottzjoxkgwncva.bcvbndqmo6y4xyp3qtvwjf6whsergeeu6gi3xvior7aotohc76y655gpwkbte.jot2hfwutr2a.dt1.aut...
235 Standard query 0x0008 A cdn.nlahrxi.usfr7gvrevvg3pzgnbxlwjy.ta7l3wfbcn4lnkkkc6bkxebh7jivkxcodcpup3jsb4dn6jq.rfluakua6mc7zxsviv4id4dejtsdgl56b3iy67kf6qci74klze.yqqgtwrdkdhax5sfq4.dt1.a...
247 Standard query 0x0009 A cdn.nlahrxi.usfr7gvqevvg3pzgnbxlwjy.stmf6rtzvtmzad3jxb57z2mksqahfe2e4x7mmnnbpjvptnz7jbkq.ishrezjlbits4yranwlhmsb46ungtsp2zazgg37742yh4wqtjwrj6ti.mjurroy32g3jcx...
163 Standard query 0x000a A cdn.nlahrxi.usfr7gvtevvg3pzgnfxlwjy.o5kxfhoy6imck3f3lauua4cz4absllc3.osky4jp57kgq.dt1.automotosport.net
 76 Standard query 0x8d96 A tse1.mm.bing.net
255 Standard query response 0x0006 A cdn.nlahrxi.usfr7gv7evvg3pzgnbxlwjy.usxjugg56yqela5t2wjjsrsowt4wkxsl5tlbdk3jyqjdje2tc5y5s.vsaxzhz4rxsejojspn2mxm6o2izxuxisj54rs.moynv2cjkyw6grexylx2f7...
249 Standard query response 0x0007 A cdn.nlahrxi.usfr7gv6evvg3pzgnbxlwjy.a6fudnh7cizzz6ckenbf4cft7cottzjoxkgwncva.bcvbndqmo6y4xyp3qtvwjf6whsergeeu6gi3xvior7aotohc76y655gpwkbte.jot2hfwutr2...
184 Standard query response 0x8d96 A tse1.mm.bing.net CNAME mm-mm.bing.net.trafficmanager.net CNAME ax-0001.ax-msedge.net A 150.171.28.10 A 150.171.27.10
251 Standard query response 0x0008 A cdn.nlahrxi.usfr7gvrevvg3pzgnbxlwjy.ta7l3wfbcn4lnkkkc6bkxebh7jivkxcodcpup3jsb4dn6jq.rfluakua6mc7zxsviv4id4dejtsdgl56b3iy67kf6qci74klze.yqqgtwrdkdhax5s...
263 Standard query response 0x0009 A cdn.nlahrxi.usfr7gvqevvg3pzgnbxlwjy.stmf6rtzvtmzad3jxb57z2mksqahfe2e4x7mmnnbpjvptnz7jbkq.ishrezjlbits4yranwlhmsb46ungtsp2zazgg37742yh4wqtjwrj6ti.mjurr...
179 Standard query response 0x000a A cdn.nlahrxi.usfr7gvtevvg3pzgnfxlwjy.o5kxfhoy6imck3f3lauua4cz4absllc3.osky4jp57kgq.dt1.automotosport.net A 213.112.200.98
117 Standard query 0x000b A cdn.5w4npba.epkl5yslpterkr35zilee7q.dt1.automotosport.net
133 Standard query response 0x000b A cdn.5w4npba.epkl5yslpterkr35zilee7q.dt1.automotosport.net A 140.223.176.229
```

zscaler | ThreatLabz

Figure 3: Example DNS C2 message comparison between the old and new versions of Zloader.

The Zloader DNS C2 message format is now the following:

```
[prefix].[session_key].[header].[payload].[zloader_nameserver_domain]
```

zscaler | ThreatLabz

Figure 4: Zloader DNS tunneling protocol message format.

A new *session* key field has been introduced that contains a random DWORD, which is used throughout the communication exchange. The session key field is used to generate the final key, which is then used to decode the query's header and payload. The final key is computed by applying an XOR operation between the Base32-encoded DWORD in the session key and a hardcoded DWORD embedded in the malware binary, which may vary between samples and instances of Zloader. Once the final key is generated, the following algorithm is used to decode the header and payload:

```
def decode_sections(bytes_array, key):
    result = bytearray()
    for byte in bytes_array:
        # XOR uses the last byte of the key, then rotates and increments.
        last_byte = key & 0xFF
        result.append(byte ^ last_byte)
        key = ((key > 24) & 0xFF)
        key = (key & 0xFFFFFF00) | ((key + 1) & 0xFF)
    return result
```

The examples in the figure below show the final structure and decoded outputs of the DNS requests:

Query

```
00000000  63 64 6e 2e 63 37 73 70  71 73 69 2e 33 65 6c 7a  |cdn.c7spqsi.3elz|
00000010  37 6a 77 66 78 68 76 66  64 71 76 32 35 62 6a 6d  |7jwfxhvfdqv25bjm|
00000020  72 6f 79 2e 73 78 33 33  75 62 68 67 73 32 65 74  |roy.sx33ubhgs2et|
00000030  35 6e 6f 6b 74 65 36 32  70 74 36 64 6f 6f 61 36  |5nokte62pt6dooa6|
00000040  72 6f 69 65 2e 35 6b 65  34 6f 79 6f 6c 77 6f 76  |roie.5ke4oyolwov|
00000050  74 66 6a 67 37 74 6d 74  70 66 67 36 67 70 74 72  |tfjg7tmtpfg6gptr|
00000060  33 64 7a 79 78 75 78 6a  79 61 6d 66 69 2e 77 68  |3dzyxuxjyamfi.wh|
00000070  69 79 6d 70 78 34 74 67  65 74 33 6b 67 6a 72 7a  |iympx4tget3kgjrz|
00000080  70 34 66 36 65 64 68 77  36 35 74 61 7a 61 34 37  |p4f6edhw65taza47|
00000090  79 71 2e 64 74 31 2e 61  75 74 6f 6d 6f 74 6f 73  |yq.dt1.automotos|
000000a0  70 6f 72 74 2e 6e 65 74                           |port.net|
```

Query properties

- Type: A
- Class: IN
- Address: 65.240.236.31
- Computed XOR Key: **0xb7e84fc5**

Decoded output

```
00000000  1c af 76 f6 03 00 00 00  05 00 03 00 00 00 2e 50  |..v............P|
00000010  4f 53 54 20 2f 63 6f 72  70 72 6f 6f 74 2f 20 48  |OST /corproot/ H|
00000020  54 54 50 2e 2f 31 2e 31  0d 0a 41 63 63 65 70 74  |TTP./1.1..Accept|
00000030  3a 20 2a 2f 2a 0d 0a 43  6f 6e 6e 65 63 2e 74 69  |: */*..Connec.ti|
00000040  6f 6e 3a 20 63 6c 6f 73  65 0d 0a 43 6f 6e 74 65  |on: close..Conte|
00000050  6e 74 2d 4c                                       |nt-L            |
```

Query

```
00000000  63 64 6e 2e 6e 6c 61 68  72 78 69 2e 75 73 66 72  |cdn.nlahrxi.usfr|
00000010  37 67 76 37 65 76 76 67  33 70 7a 67 6e 62 78 6c  |7gv7evvg3pzgnbxl|
00000020  77 6a 79 2e 75 73 78 6a  75 67 67 35 36 79 71 65  |wjy.usxjugg56yqe|
00000030  6c 61 35 74 32 77 6a 6a  73 72 73 6f 77 74 34 77  |la5t2wjjsrsowt4w|
00000040  6b 78 73 6c 35 74 6c 62  64 6b 33 6a 79 71 6a 64  |kxsl5tlbdk3jyqjd|
00000050  6a 65 32 74 63 35 79 35  73 2e 76 73 61 78 7a 68  |je2tc5y5s.vsaxzh|
00000060  7a 34 72 78 73 65 6a 6f  6a 73 70 6e 32 6d 78 6d  |z4rxsejojspn2mxm|
00000070  36 6f 32 69 7a 78 75 78  69 73 6a 35 34 72 73 2e  |6o2izxuxisj54rs.|
00000080  6d 6f 79 6e 76 32 63 6a  6b 79 77 36 67 72 65 78  |moynv2cjkyw6grex|
00000090  79 6c 78 32 66 37 75 71  6c 61 32 6e 65 2e 64 74  |ylx2f7uqla2ne.dt|
000000a0  31 2e 61 75 74 6f 6d 6f  74 6f 73 70 6f 72 74 2e  |1.automotosport.|
000000b0  6e 65 74                                          |net             |
```

Query properties

- Computed XOR Key: **0x23686bb8**

Decoded output

```
00000000  1c af 76 f6 06 00 00 00  05 00 03 00 00 00 2e 1c  |..v.............|
00000010  8a aa 74 64 d3 4a 28 39  10 be fc 22 61 22 db 45  |..td.J(9..."a".E|
00000020  4d 33 3b 51 ff 7f da d7  ee 7d 46 2c 78 67 02 19  |M3;Q.....}F,xg..|
00000030  2e 14 a5 15 f3 85 a8 8e  29 03 14 10 1b 70 94 a2  |........)....p..|
00000040  bd 8f 52 11 62 f2 50 77  2e db 94 b3 84 f0 73 47  |..R.b.Pw......sG|
```

Figure 5: Showcases the final structure and decoded outputs of the DNS requests.

The purpose of switching from TLS-based encryption to a custom algorithm may be due to the fact that the TLS messages can easily be identified in DNS traffic due to their well defined structure. Thus, this change was likely made to better evade network-based signatures.

After decryption, the Zloader DNS tunnel header is identical to previous versions as shown below:

```
struct zloader_dns_tunnel_header {
  unsigned int session_id;          // Randomly generated.
  unsigned int sequence_num;        // Incremented per packet.
  byte msg_type;                    // 1-9
  byte reserved;                    // Reserved
  unsigned int generic_var;         // Varies by msg_type
};
```

Once all components of the payload have been sent or received, the data format structure aligns with Zloader's HTTPS communications. The payload is first encrypted using the Zeus `VisualEncrypt` algorithm, followed by encryption with a randomly generated 32-byte (256-bit) RC4 key. Finally, the RC4 key itself is encrypted with a hardcoded 1,024-bit RSA public key.

WebSocket support

In the latest versions, Zloader introduced WebSockets that can be used to upgrade the HTTP connection with the following hardcoded header:

```
GET %s HTTP/1.1\
Host: %s\
Connection: Upgrade
Pragma: no-cache
Cache-Control: no-cache
User-Agent: %s
Upgrade: websocket
Origin: %s
Sec-WebSocket-Version: 13
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: %s
Sec-WebSocket-Key: %s
```

The introduction of WebSockets in Zloader may be designed to further blend in with legitimate web-based traffic to bypass network-based detections.