Poison Ivy for Incident Responders

Andreas Schuster

Poison Ivy in the Press

TECHSPOT
TECHNOLOGY NEWS AND ANALYSIS

TECHSPOT | NEWS | PRODUCTS | DOWNLOADS | FORUMS
Search

Home | Reviews | Guides | Product Finder | Forums | Downloads | Drivers | Extras
Login or Sign up | About

Trending | Features | Hardware | Software | Mobile | Gaming | The Web | IT | Apple | Microsoft | Security | More

HOME › NEWS › SECURITY

PoisonIvy RAT used to extract data from chemical and defense firms

By Shawn Knight
On October 31, 2011, 4:00 PM EST

14 | +1 1 | Tweet 17

Symantec Corp has revealed that a coordinated cyber attack targeted at least 48 chemical and defense companies in the US, Bangladesh and the UK. The source of the attack has been traced to a man in China, according to the report.

threatpost
The Kaspersky Lab Security News Service

Sunday, May 20th, 2012

Google™ Custom Sear Search

Apple | Cloud | Compliance | Critical Infrastructure | Cryptography | Governm
Microsoft | Mobile Security | SMB | Social Engineering | Virtualization | Vulne

Home › SMB Security ›

November 3, 2011, 8:17AM

Poison Ivy RAT Still Giving Users a Rash

by Dennis Fisher
Follow @DennisF

Share
1 Comment

The Poison Ivy malware kit is old. It was first seen in 2005, which makes it about 762 years old in Internet years. But that doesn't mean it's no longer useful, as evinced by the data collected by Microsoft in a new report on the tool, which shows that it is still in active use and is turning up on thousands of infected PCs.

HOME | ABOUT | BLOG | RESEARCH | ARCHIVE | CONTACT

Flash Malware Leads to Poison Ivy RAT on Human Rights Site

By shardy Published: July 13, 2011

Tags: Civil Society and NGOs , Cyber Security , human rights , Malware

Digg it! | Facebook

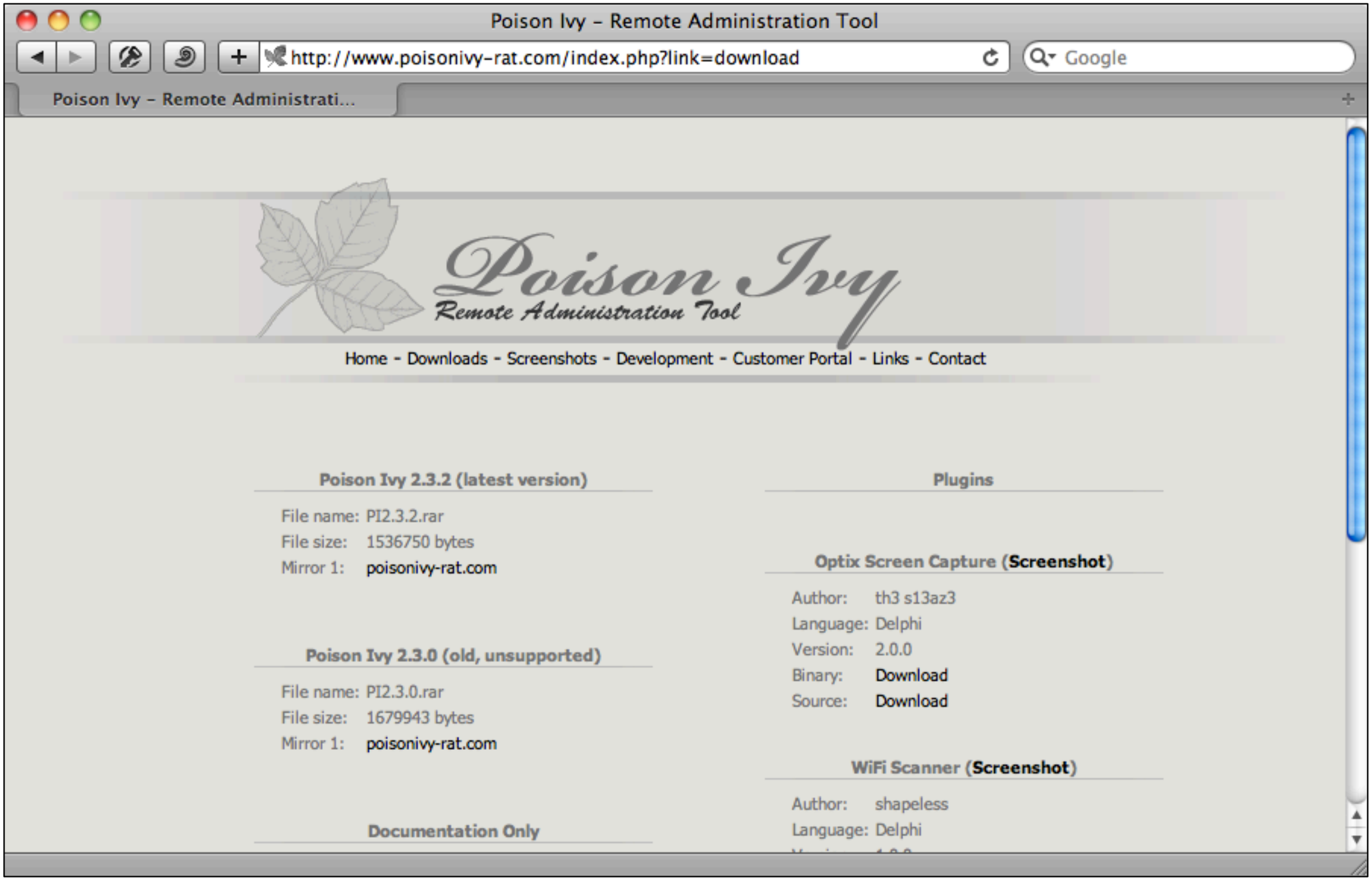
Human rights and civil society organizations face a growing spectrum of online threats, including Internet filtering, website defacements, denial of service attacks, and targeted malware (malicious software) attacks. Such organizations can be particularly vulnerable to these threats due to limited resources and lack of computer network security support.

What is Poison Ivy?

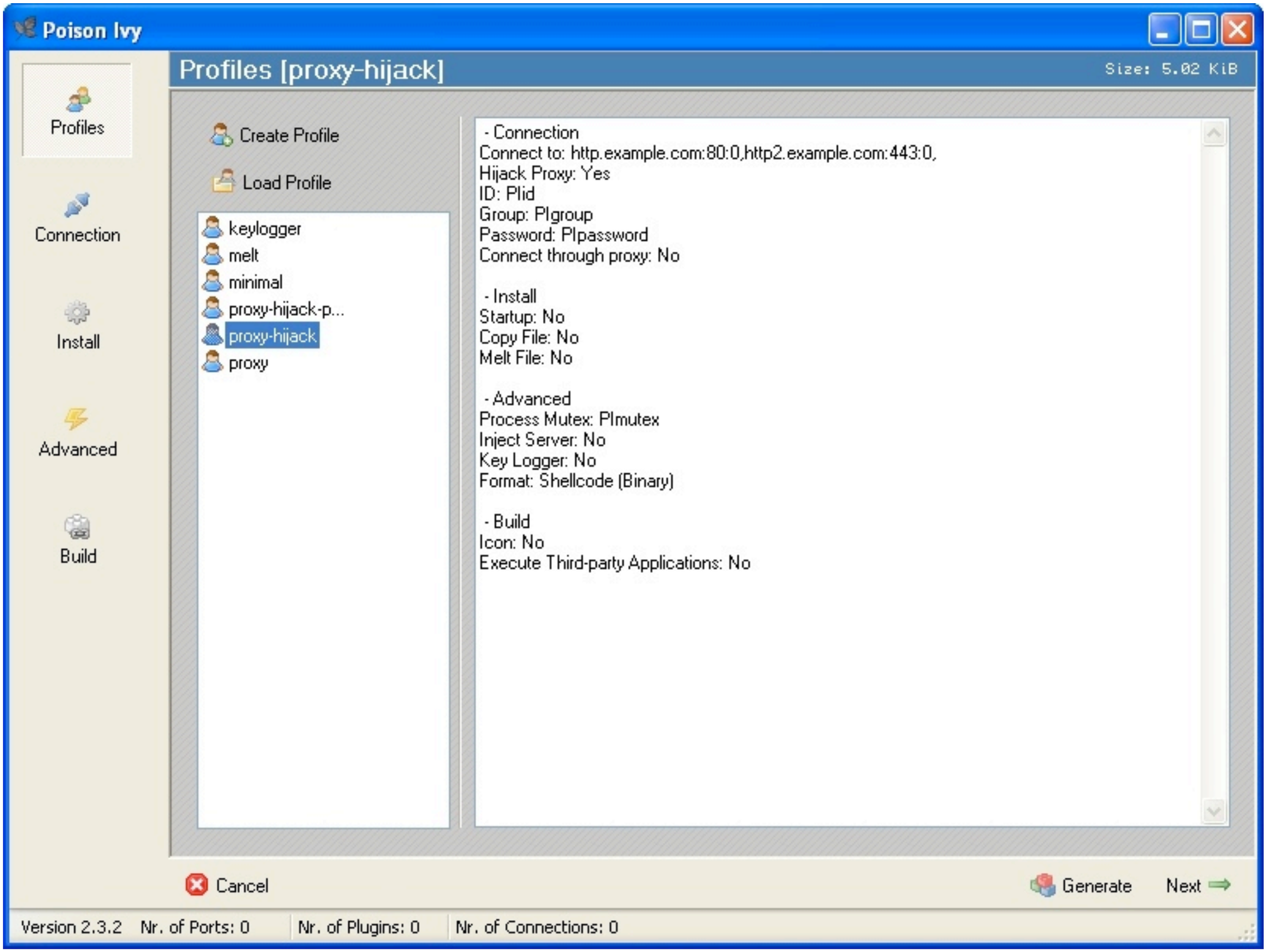
Poison Ivy is a Powerful RAT

- Target platform: Microsoft Windows, 32bit
- System information and manipulation
- Keyword search
- Password collection
- Shell (cmd.exe)
- Surveillance
- Lateral movement: relaying, sharing
- Administration (update, removal)

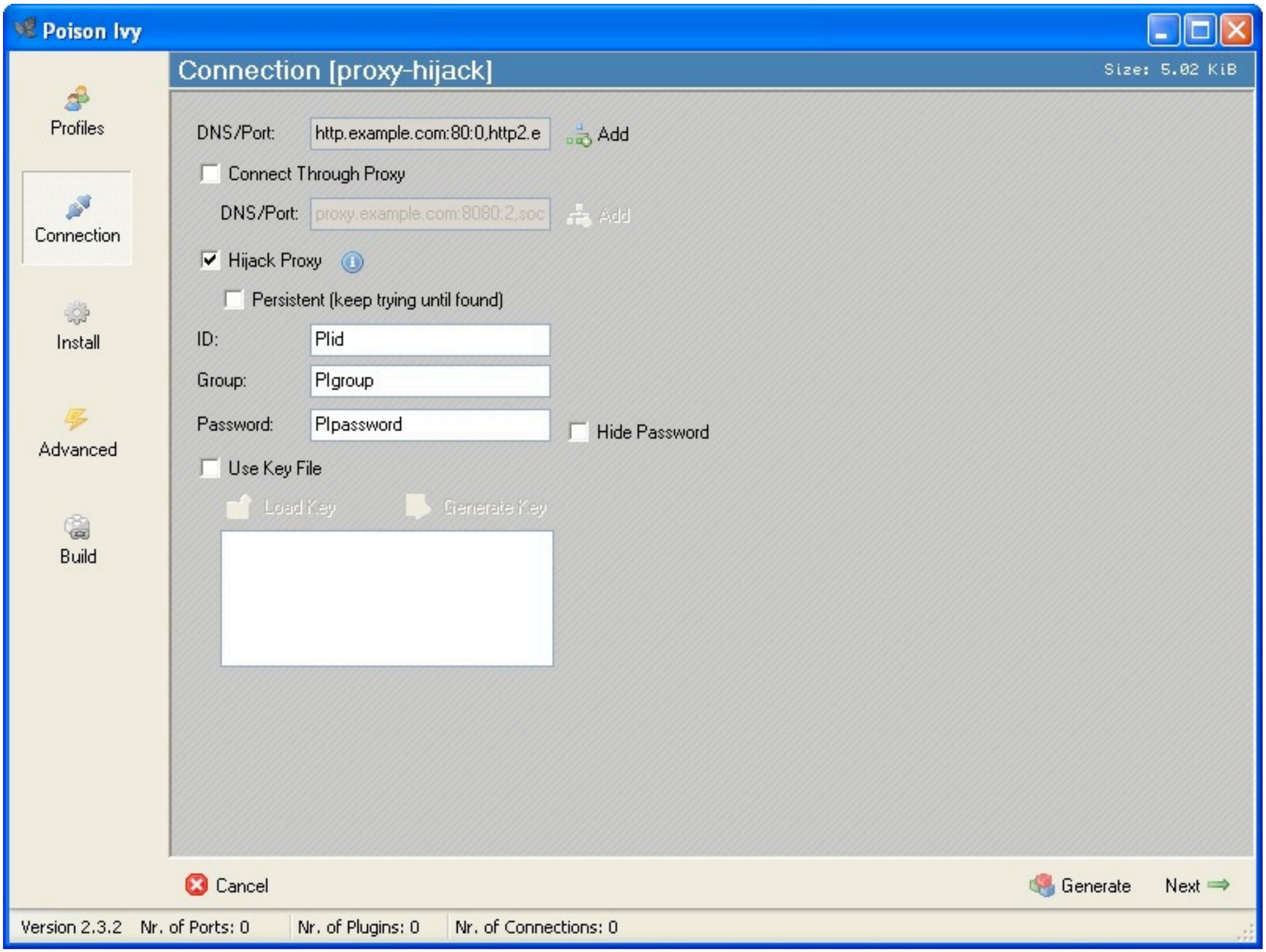
Poison Ivy is Free, but Closed Source



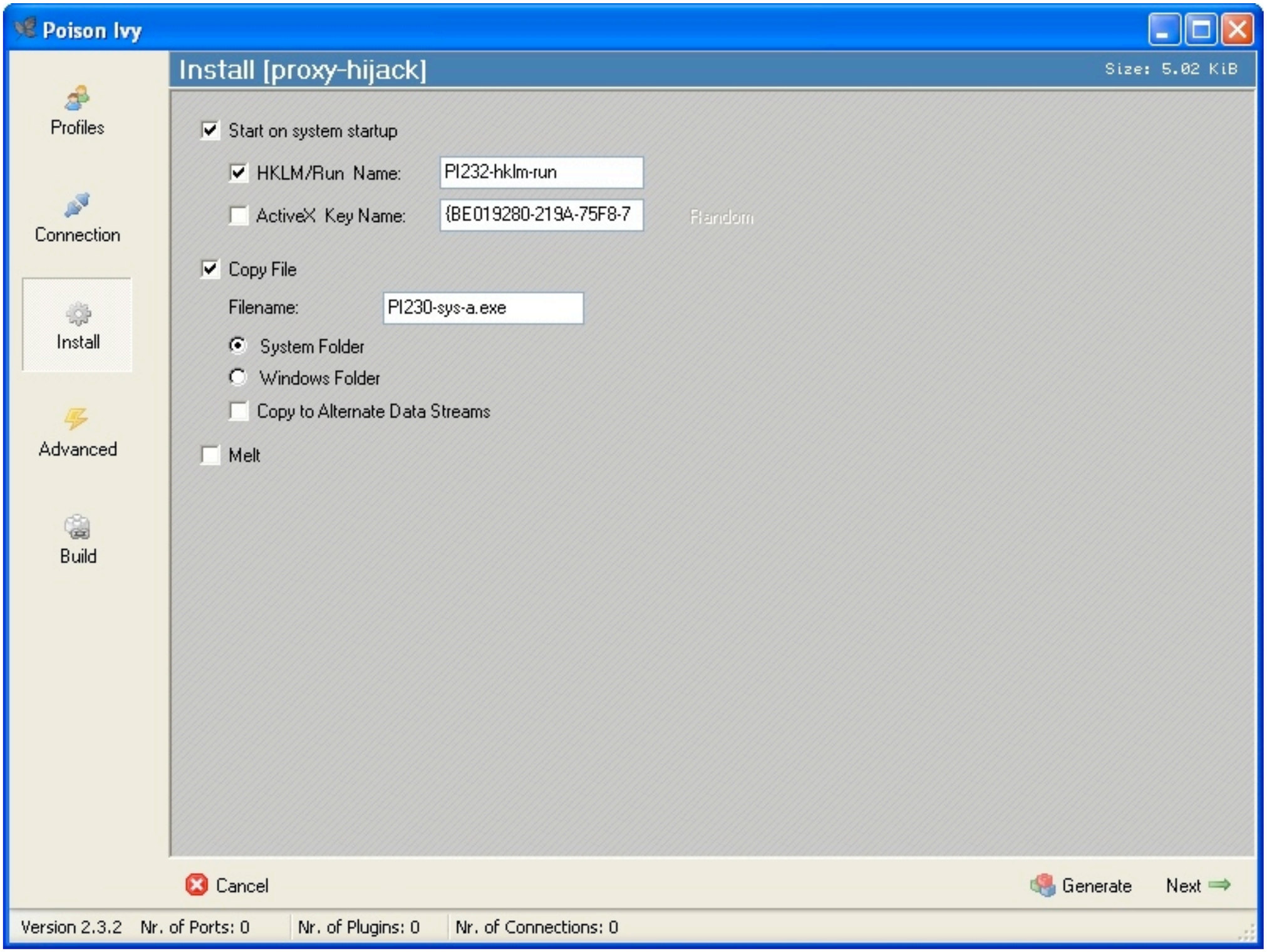
Builder – Step 1



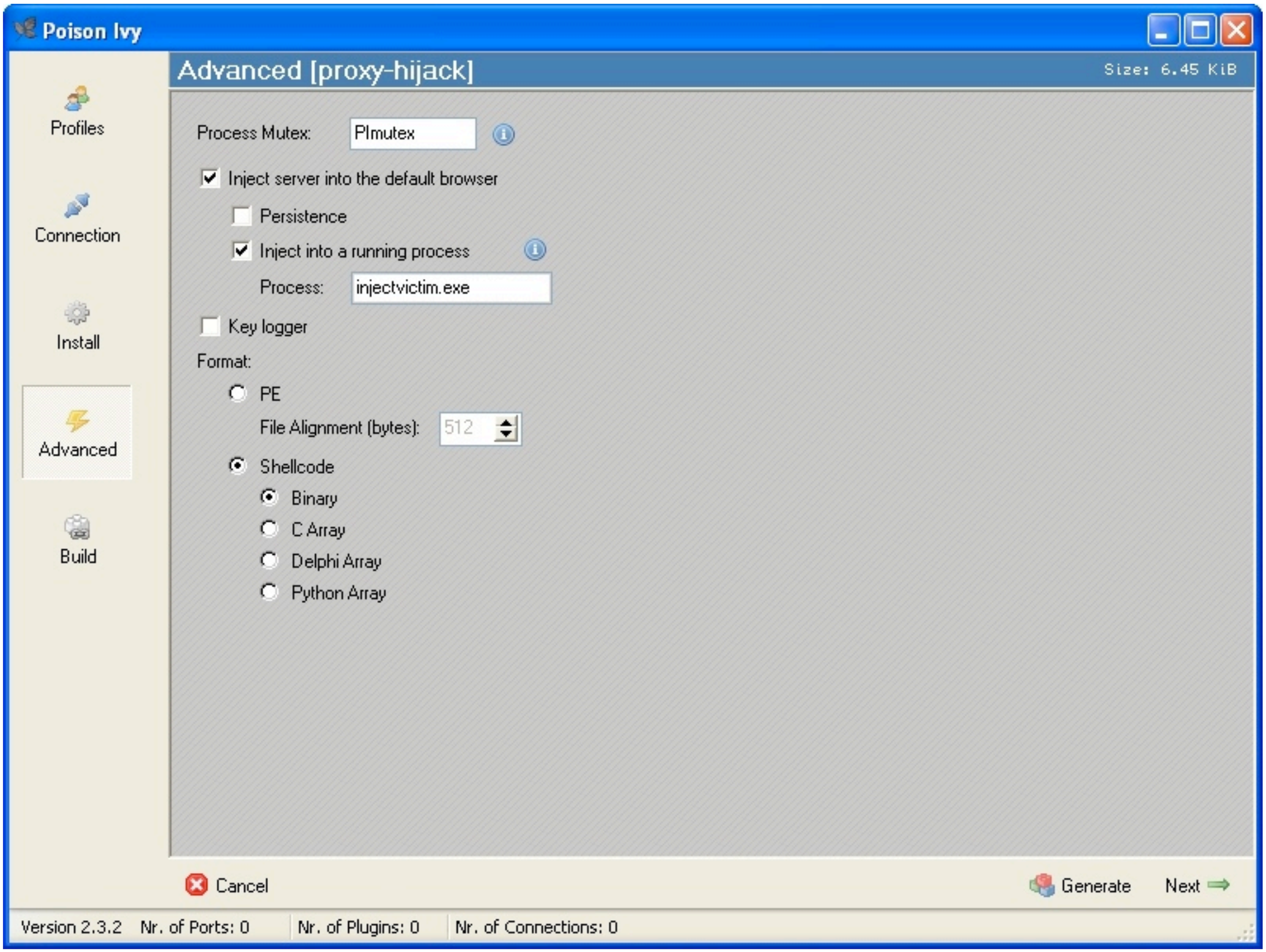
Builder – Step 2



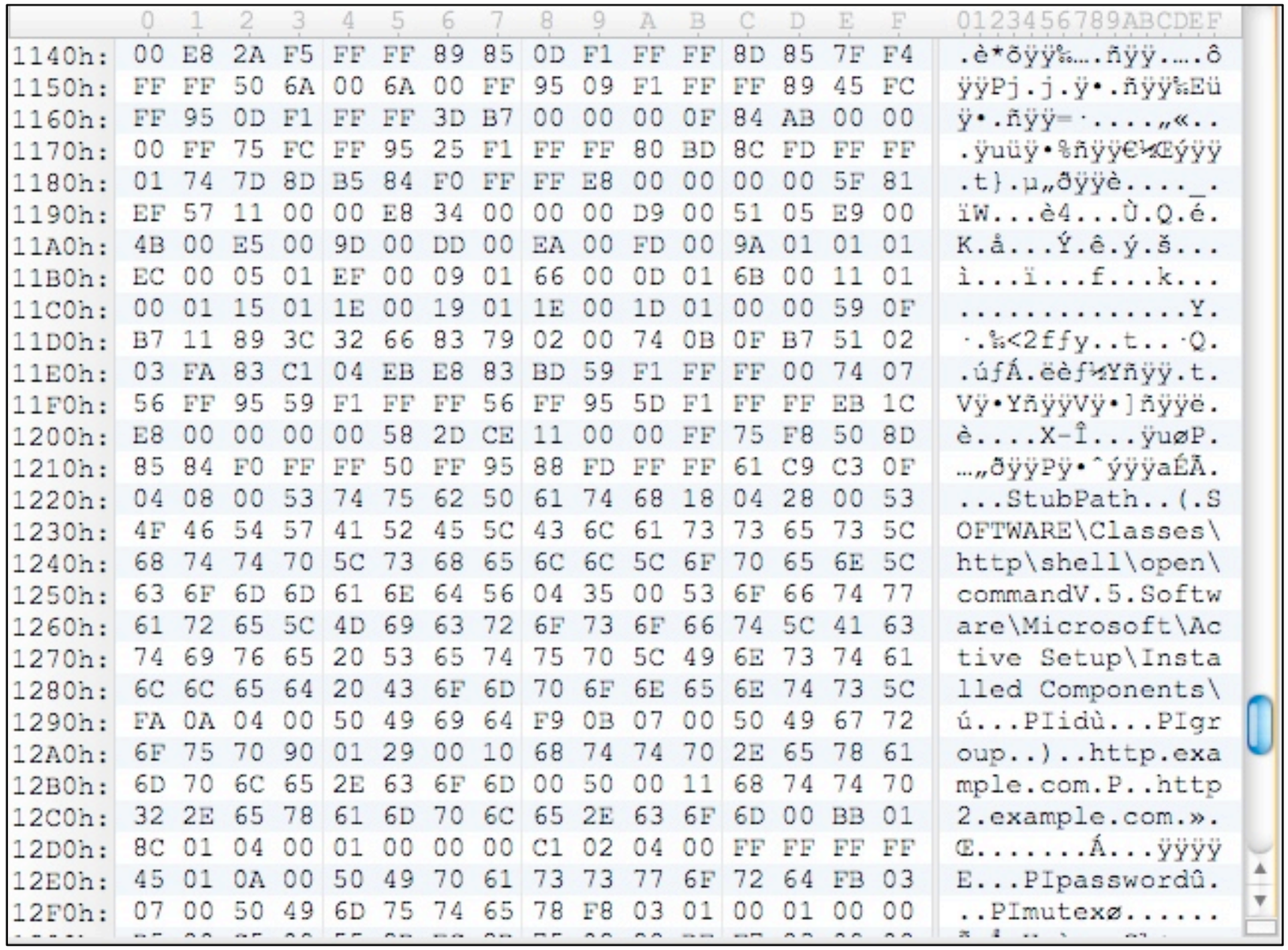
Builder – Step 3



Builder – Step 4

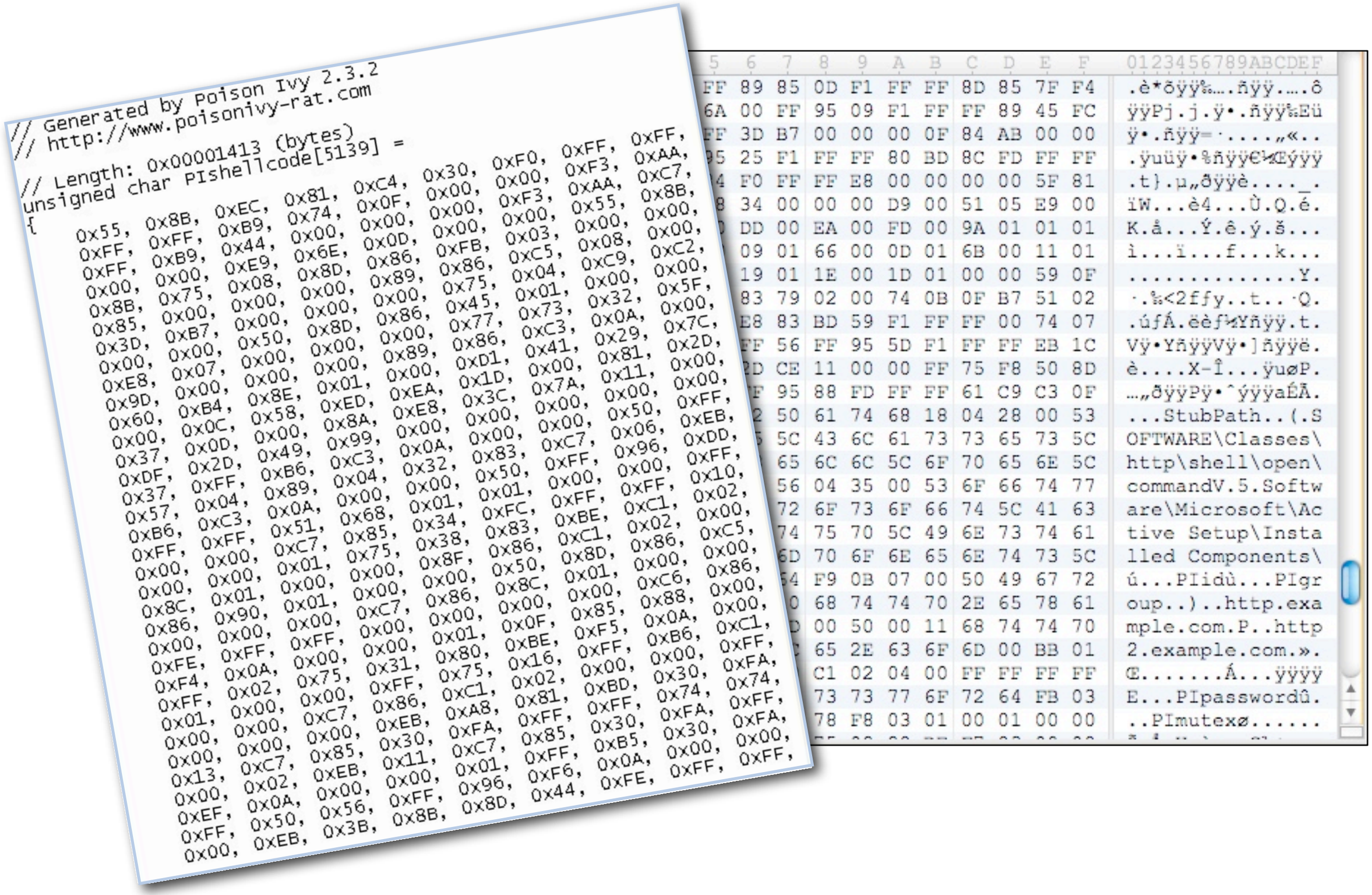


Builder – Output Formats

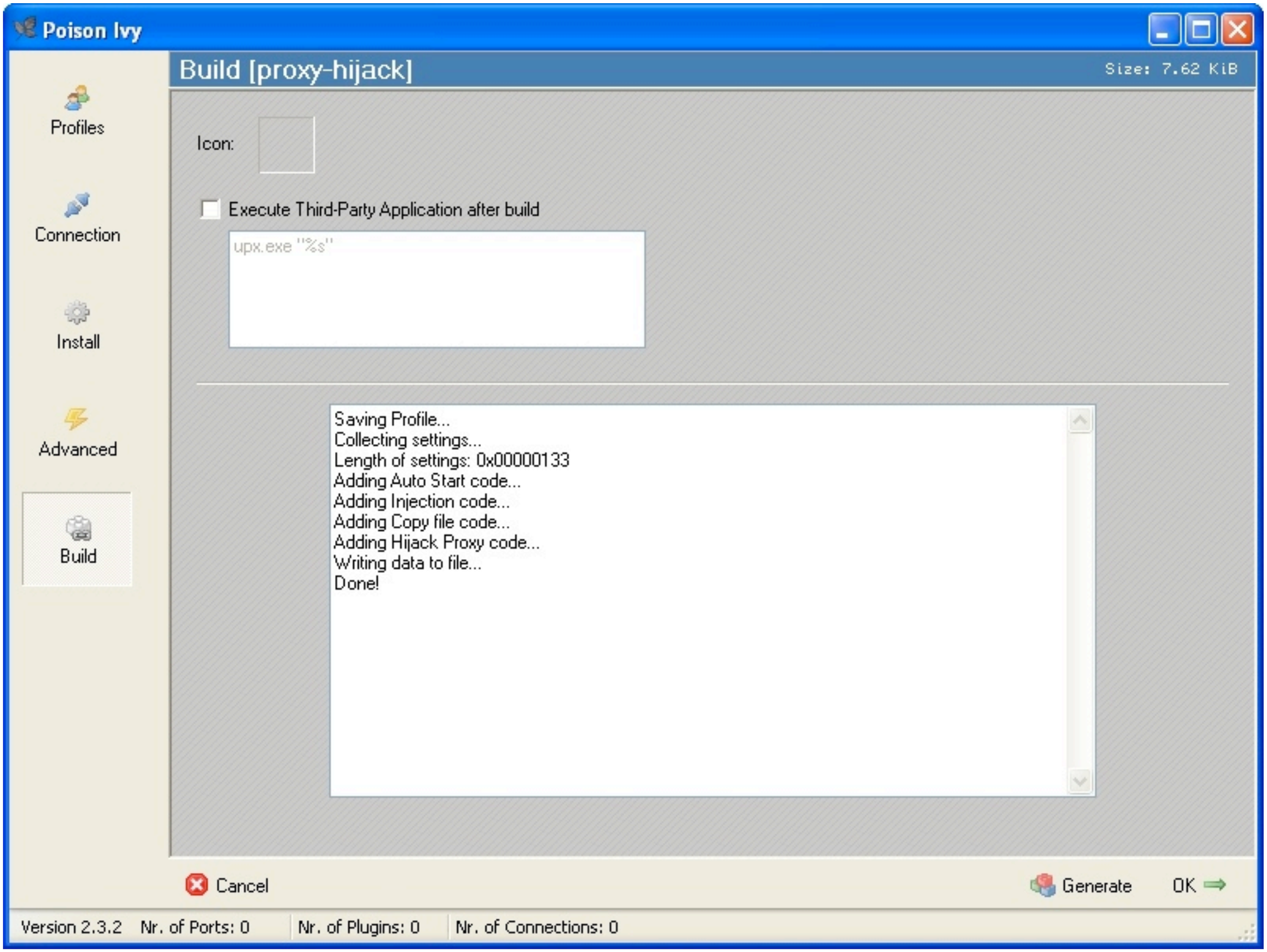


	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
1140h:	00	E8	2A	F5	FF	FF	89	85	0D	F1	FF	FF	8D	85	7F	F4	.è*õÿÿ‰….ñÿÿ.…ô
1150h:	FF	FF	50	6A	00	6A	00	FF	95	09	F1	FF	FF	89	45	FC	ÿÿPj.j.ÿ•.ñÿÿ‰Eü
1160h:	FF	95	0D	F1	FF	FF	3D	B7	00	00	00	0F	84	AB	00	00	ÿ•.ñÿÿ=·....„«..
1170h:	00	FF	75	FC	FF	95	25	F1	FF	FF	80	BD	8C	FD	FF	FF	.ÿuüÿ•%ñÿÿ€½Œýÿÿ
1180h:	01	74	7D	8D	B5	84	F0	FF	FF	E8	00	00	00	00	5F	81	.t}.µ„ðÿÿè...._.
1190h:	EF	57	11	00	00	E8	34	00	00	00	D9	00	51	05	E9	00	ïW...è4...Ù.Q.é.
11A0h:	4B	00	E5	00	9D	00	DD	00	EA	00	FD	00	9A	01	01	01	K.å...Ý.ê.ý.š...
11B0h:	EC	00	05	01	EF	00	09	01	66	00	0D	01	6B	00	11	01	ì...ï...f...k...
11C0h:	00	01	15	01	1E	00	19	01	1E	00	1D	01	00	00	59	0F	..............Y.
11D0h:	B7	11	89	3C	32	66	83	79	02	00	74	0B	0F	B7	51	02	·.‰<2fƒy..t..·Q.
11E0h:	03	FA	83	C1	04	EB	E8	83	BD	59	F1	FF	FF	00	74	07	.úƒÁ.ëèƒ½Yñÿÿ.t.
11F0h:	56	FF	95	59	F1	FF	FF	56	FF	95	5D	F1	FF	FF	EB	1C	Vÿ•YñÿÿVÿ•]ñÿÿë.
1200h:	E8	00	00	00	00	58	2D	CE	11	00	00	FF	75	F8	50	8D	è....X-Î...ÿuøP.
1210h:	85	84	F0	FF	FF	50	FF	95	88	FD	FF	FF	61	C9	C3	0F	…„ðÿÿPÿ•ˆýÿÿaÉÃ.
1220h:	04	08	00	53	74	75	62	50	61	74	68	18	04	28	00	53	...StubPath..(.S
1230h:	4F	46	54	57	41	52	45	5C	43	6C	61	73	73	65	73	5C	OFTWARE\Classes\
1240h:	68	74	74	70	5C	73	68	65	6C	6C	5C	6F	70	65	6E	5C	http\shell\open\
1250h:	63	6F	6D	6D	61	6E	64	56	04	35	00	53	6F	66	74	77	commandV.5.Softw
1260h:	61	72	65	5C	4D	69	63	72	6F	73	6F	66	74	5C	41	63	are\Microsoft\Ac
1270h:	74	69	76	65	20	53	65	74	75	70	5C	49	6E	73	74	61	tive Setup\Insta
1280h:	6C	6C	65	64	20	43	6F	6D	70	6F	6E	65	6E	74	73	5C	lled Components\
1290h:	FA	0A	04	00	50	49	69	64	F9	0B	07	00	50	49	67	72	ú...PIidù...PIgr
12A0h:	6F	75	70	90	01	29	00	10	68	74	74	70	2E	65	78	61	oup..)..http.exa
12B0h:	6D	70	6C	65	2E	63	6F	6D	00	50	00	11	68	74	74	70	mple.com.P..http
12C0h:	32	2E	65	78	61	6D	70	6C	65	2E	63	6F	6D	00	BB	01	2.example.com.».
12D0h:	8C	01	04	00	01	00	00	00	C1	02	04	00	FF	FF	FF	FF	Œ.......Á...ÿÿÿÿ
12E0h:	45	01	0A	00	50	49	70	61	73	73	77	6F	72	64	FB	03	E...PIpasswordû.
12F0h:	07	00	50	49	6D	75	74	65	78	F8	03	01	00	01	00	00	..PImutexø.....

Builder – Output Formats

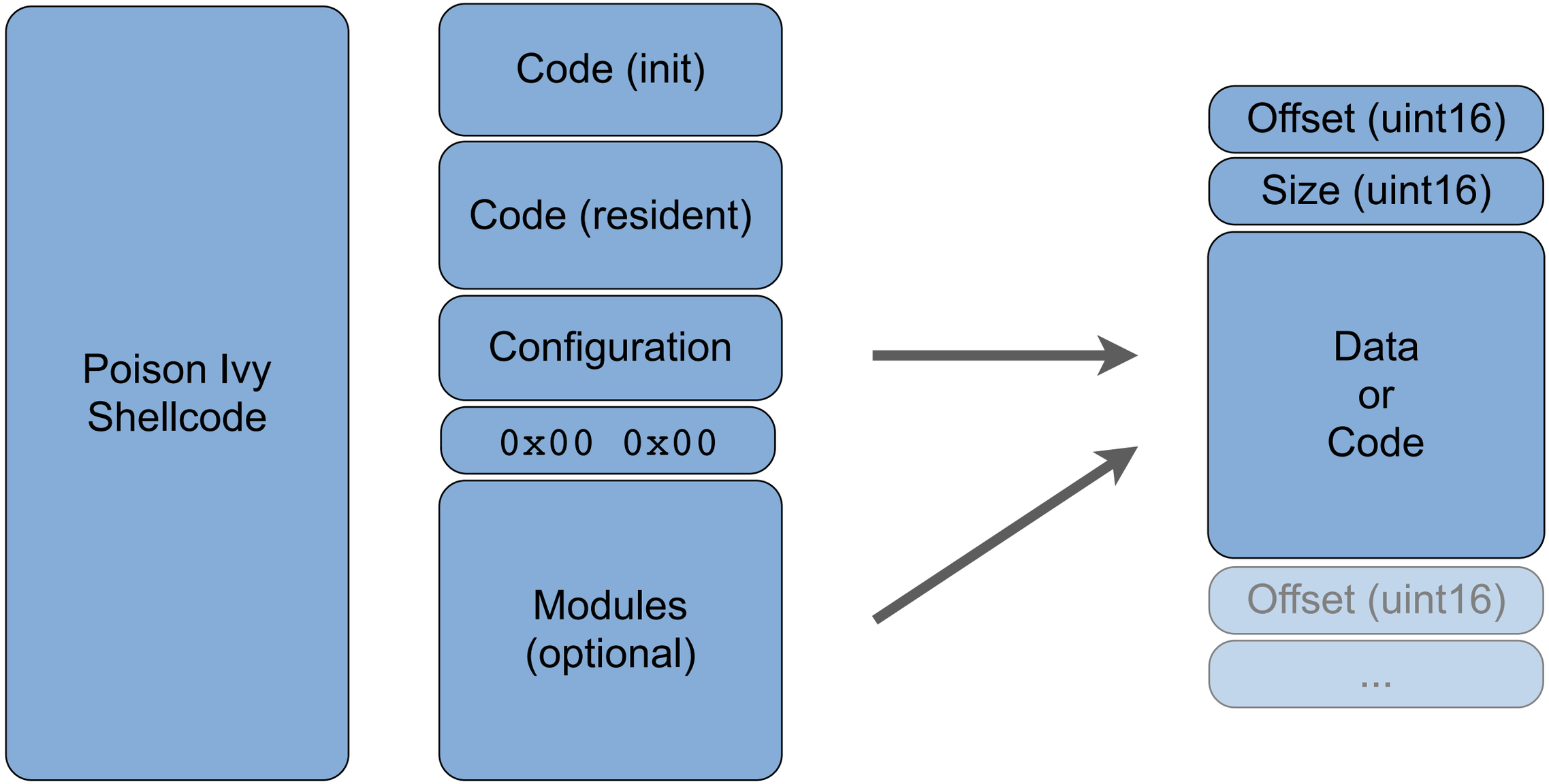


Builder – Step 5



On Disk

File Structure



Configuration Section

- Data types:
  - Boolean values (uchar 0x01)
  - Integers (int32)
  - Strings
  - Host entry:
    - host name (string)
    - protocol (uchar: 0=direct, 1=SOCKS, 2=HTTP)
    - port (ushort)

- Offsets can be identified with a meaning, e.g.:
  - 0x3fa: Activate keylogger (boolean)
  - 0x3fb: Mutex (string, max. 20 characters)
  - 0xafa: Id (string, max. 255 characters)
  - 0xd0e: ptr Keylogger_setup()

A Simple Decoder

```
143
144 // ============================================================================
145 //
146 // Parser loop
147 //
148
149 local uint64 start = GetCursorPos();
150 local uint32 items = 0;
151 FSeek(start);
152
153 // parse config variables
154 while ((! FEof()) && (ReadUShort(FTell()) != 0))
155 {
156     PIITEM      item;
157     items++;
158 };
159
160 FSkip(2);
161
162 // parse code modules
163 items = 0;
164 while ((! FEof()) && (ReadUShort(FTell()) != 0))
165 {
166     MODULE      module;
167     items++;
168 };
169
```

A Simple Decoder

```
143
144 // ============================================================================
145 //
146 // Parser loop
147 //
148
149 local uint64 start = GetCursorP
150 local uint32 items = 0;
151 FSeek(start);
152
153 // parse config variables
154 while ((! FEof()) && (ReadUShor
155 {
156     PIITEM      item;
157     items++;
158 };
159
160 FSkip(2);
161
162 // parse code modules
163 items = 0;
164 while ((! FEof()) && (ReadUShor
165 {
166     MODULE      module;
167     items++;
168 };
169
```

Name	Value
▶ struct PIITEM item[3]	Id: 8-17
▼ struct PIITEM item[4]	
enum PIOFFSET offset	Hosts (190h)
uint16 size	18
▶ struct HOSTENTRY host	domain.rm6.org:80 (direct)
▶ struct PIITEM item[5]	
▶ struct PIITEM item[6]	
▶ struct PIITEM item[7]	Secret: admin
▶ struct PIITEM item[8]	Persistence by ActiveSetup: YES
▶ struct PIITEM item[9]	ActiveSetup GUID: {56F89C06-FB85-B973-C7A0-DFA94FA985D3}
▶ struct PIITEM item[10]	Mutex: 8-17
▶ struct PIITEM item[11]	Use proxy: YES
▶ struct PIITEM item[12]	
▶ struct PIITEM item[13]	Copy to file: msvces.exe
▶ struct PIITEM item[14]	Copy to dir: %WINDIR%\System32
▶ struct PIITEM item[15]	Melt original file: YES
▶ struct PIITEM item[16]	Inject: YES
▶ struct MODULE module[0]	Table ofs 0f5h, size 662
▶ struct MODULE module[1]	Table ofs 0d5h, size 197
▶ struct MODULE module[2]	Table ofs d04h, size 360
▶ struct MODULE module[3]	Table ofs 0d1h, size 60
▶ struct MODULE module[4]	Table ofs 0c5h, size 155
▶ struct MODULE module[5]	Table ofs 0f1h, size 579
▶ struct MODULE module[6]	Table ofs d00h, size 522

Keylogger Module

- Registers hook procedure through SetWindowsHookEx API.
- WH_JOURNALRECORD: process input messages that were posted to system message queue.
- Handler routine logs
  - time stamp in SYSTEMTIME format
  - title of active window
  - character, or key name
- Log file: trojan file name, minus the „exe“
  - eg. C:\Windows\mytrojan.exe
  - becomes C:\Windows\mytrojan.

On the Wire

Authentication

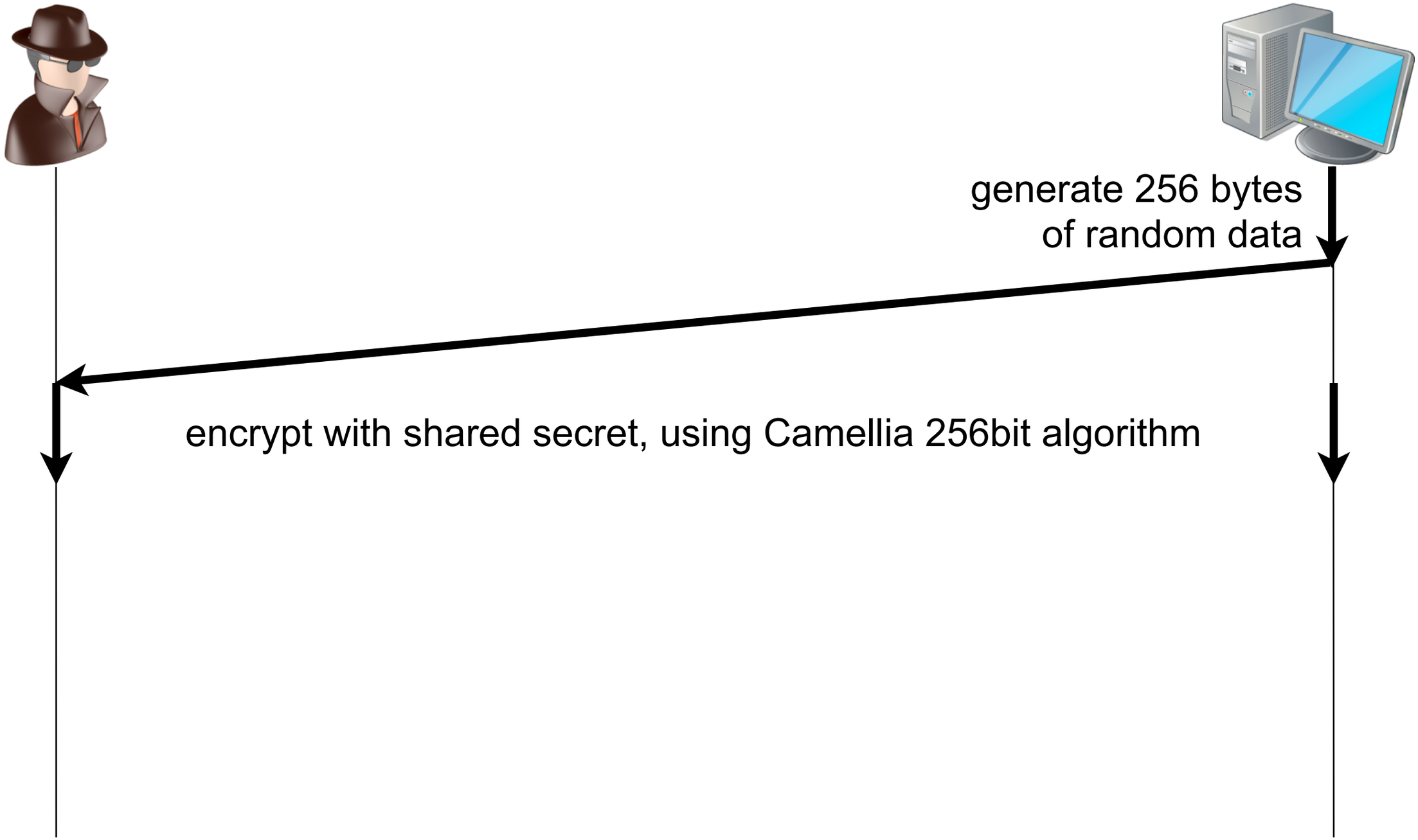
Authentication

generate 256 bytes
of random data

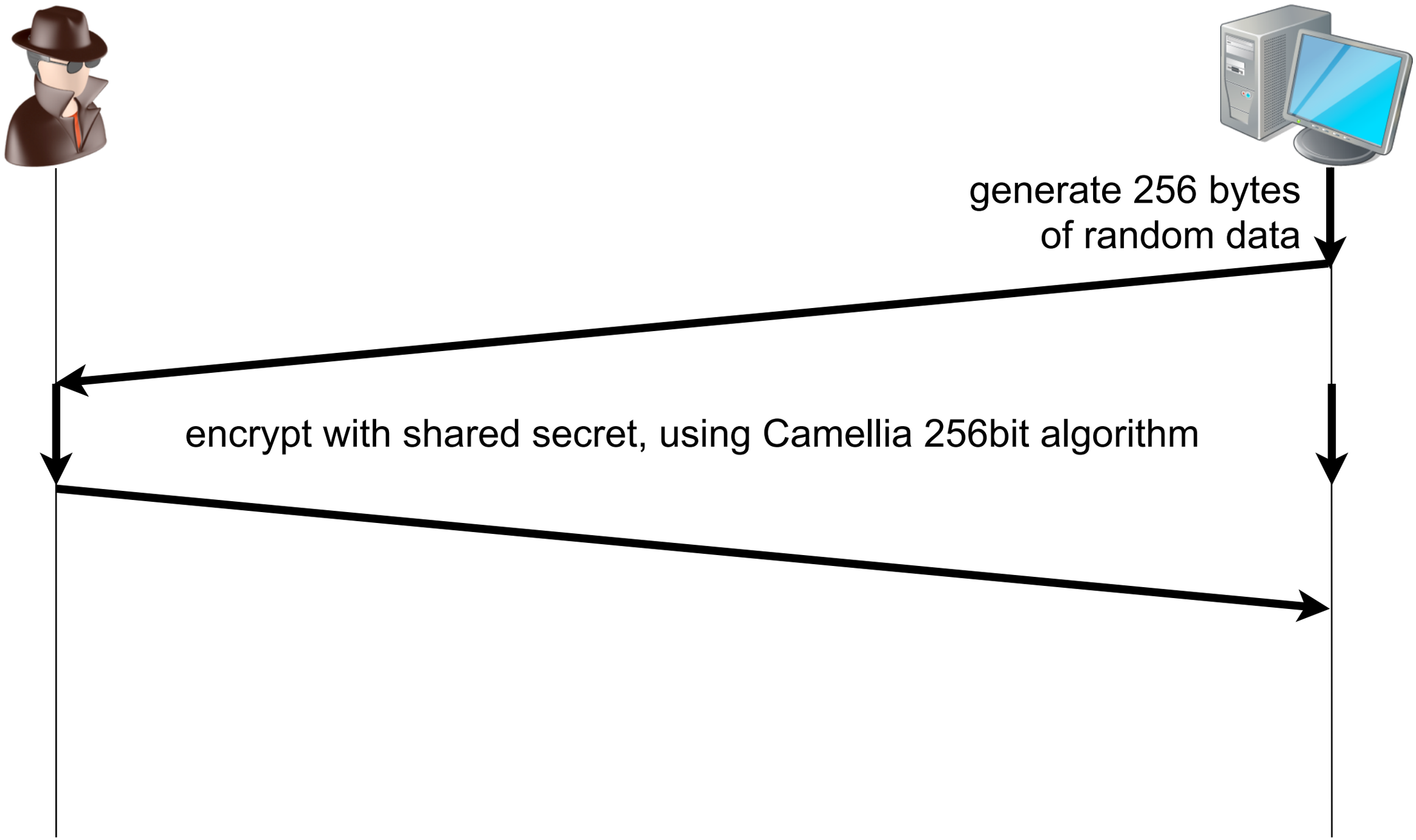
Authentication

generate 256 bytes
of random data

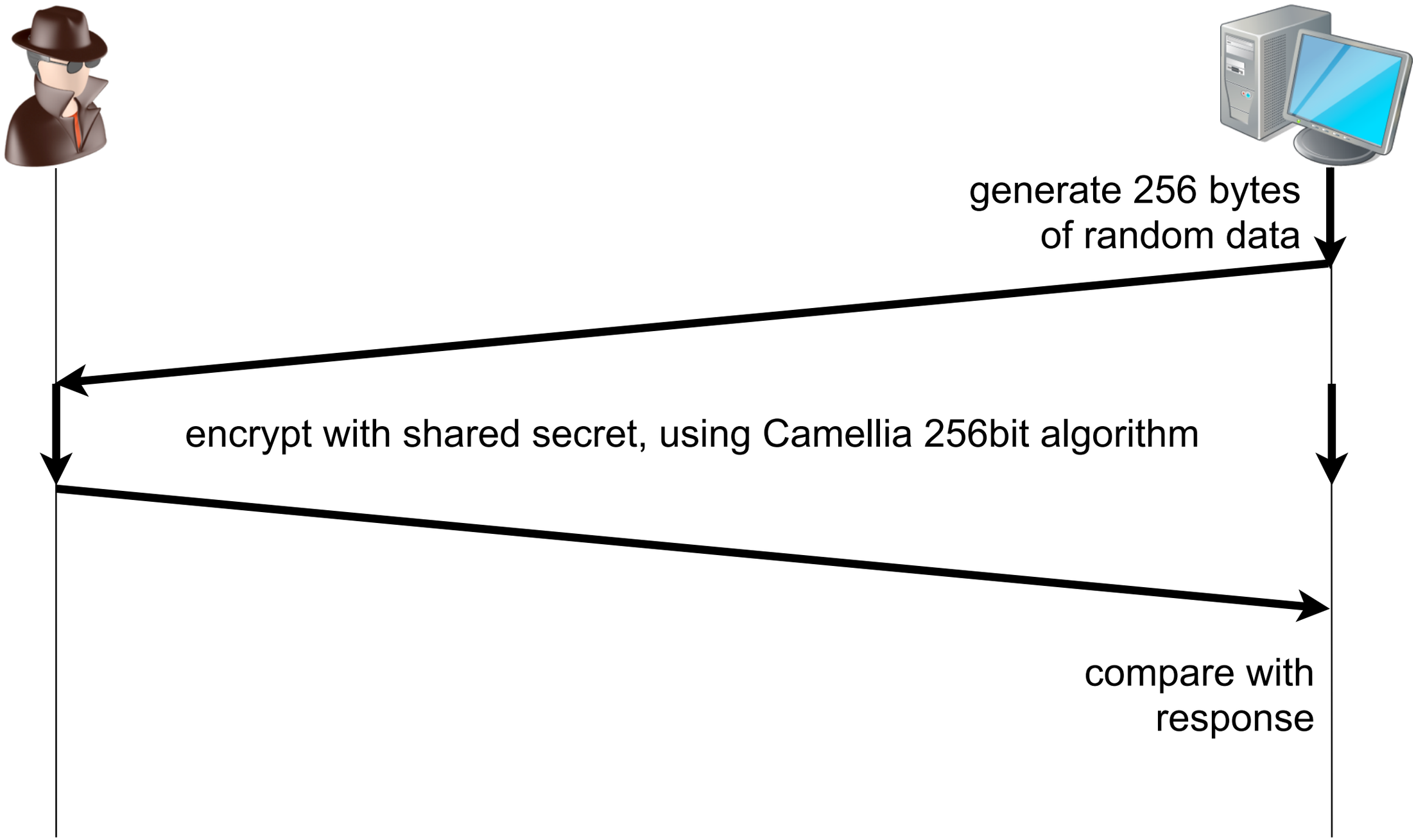
Authentication



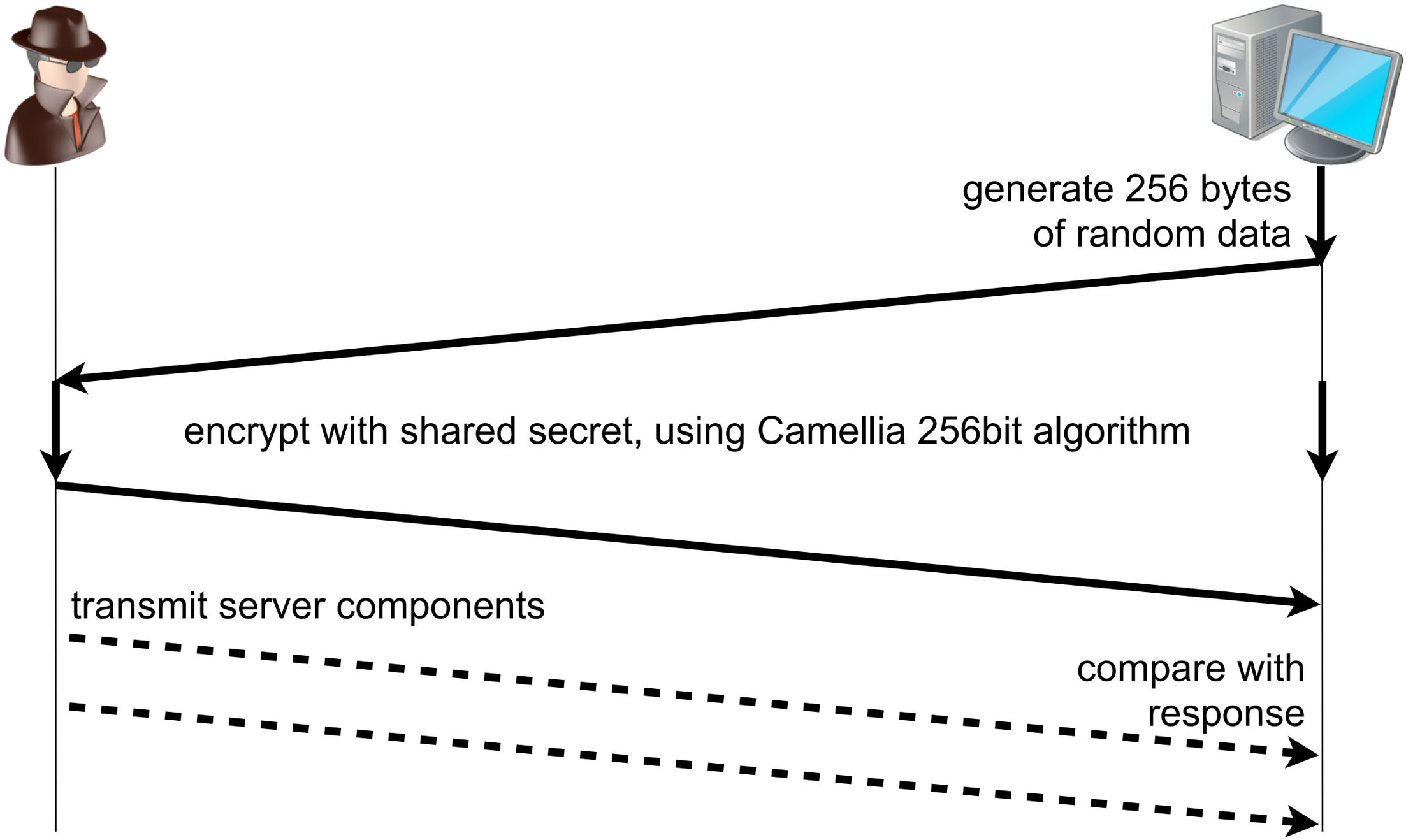
Authentication



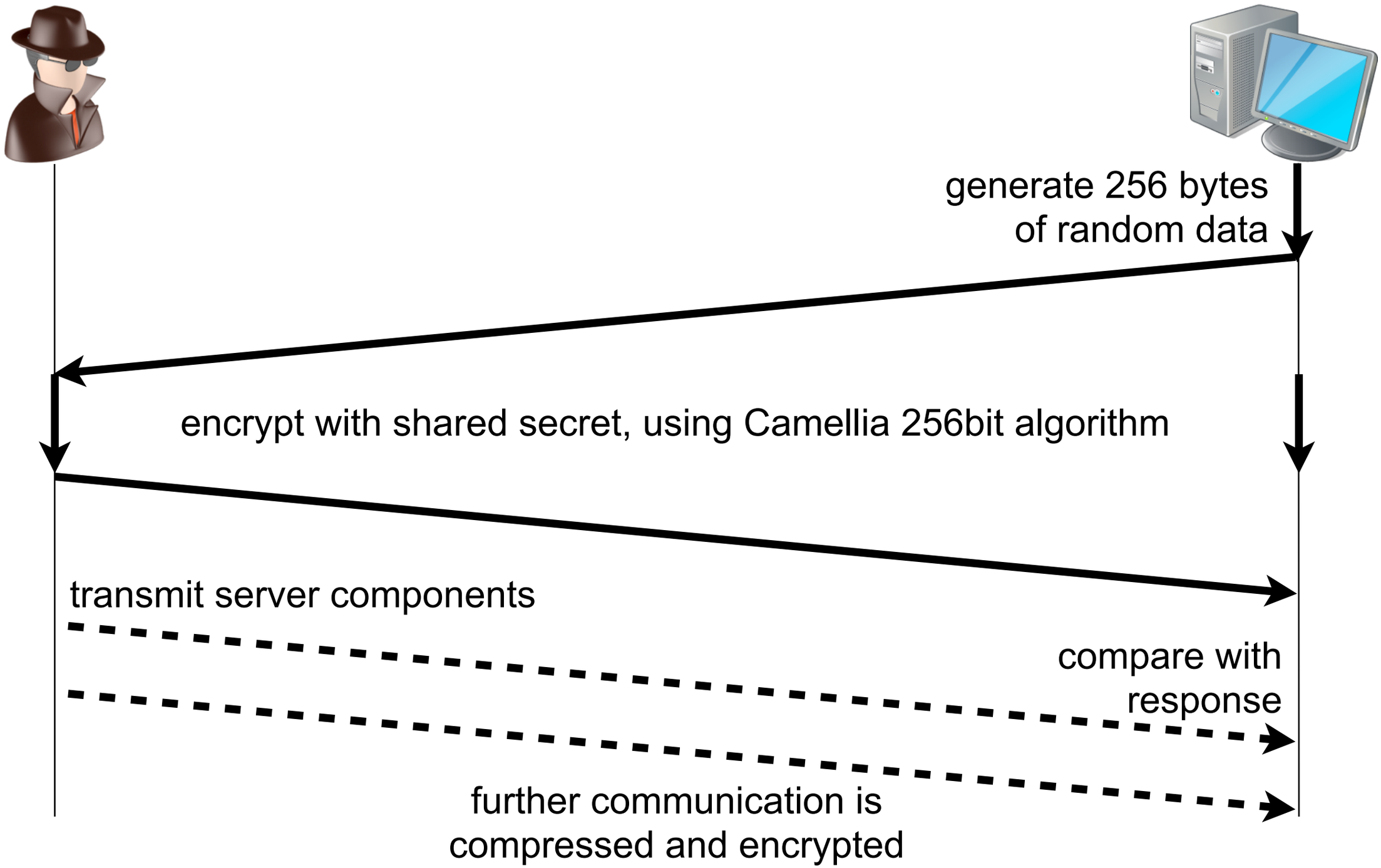
Authentication



Authentication



Authentication



An Impossible Challenge

```
$ dd if=/dev/zero of=challenge.bin bs=256 count=1
$ hexdump -Cv challenge.bin
00000000  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
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 00  |................|
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

The Characteristic Response Pattern

```
$ nc suspect.example.org 80 <challenge.bin >response.bin
$ hexdump -Cv -n 256 response.bin
00000000  35 e1 06 6c cd 15 87 3e  ee f8 51 89 66 b7 0f 8b  |5..l...>..Q.f...|
00000010  35 e1 06 6c cd 15 87 3e  ee f8 51 89 66 b7 0f 8b  |5..l...>..Q.f...|
00000020  35 e1 06 6c cd 15 87 3e  ee f8 51 89 66 b7 0f 8b  |5..l...>..Q.f...|
00000030  35 e1 06 6c cd 15 87 3e  ee f8 51 89 66 b7 0f 8b  |5..l...>..Q.f...|
00000040  35 e1 06 6c cd 15 87 3e  ee f8 51 89 66 b7 0f 8b  |5..l...>..Q.f...|
00000050  35 e1 06 6c cd 15 87 3e  ee f8 51 89 66 b7 0f 8b  |5..l...>..Q.f...|
00000060  35 e1 06 6c cd 15 87 3e  ee f8 51 89 66 b7 0f 8b  |5..l...>..Q.f...|
00000070  35 e1 06 6c cd 15 87 3e  ee f8 51 89 66 b7 0f 8b  |5..l...>..Q.f...|
00000080  35 e1 06 6c cd 15 87 3e  ee f8 51 89 66 b7 0f 8b  |5..l...>..Q.f...|
00000090  35 e1 06 6c cd 15 87 3e  ee f8 51 89 66 b7 0f 8b  |5..l...>..Q.f...|
000000a0  35 e1 06 6c cd 15 87 3e  ee f8 51 89 66 b7 0f 8b  |5..l...>..Q.f...|
000000b0  35 e1 06 6c cd 15 87 3e  ee f8 51 89 66 b7 0f 8b  |5..l...>..Q.f...|
000000c0  35 e1 06 6c cd 15 87 3e  ee f8 51 89 66 b7 0f 8b  |5..l...>..Q.f...|
000000d0  35 e1 06 6c cd 15 87 3e  ee f8 51 89 66 b7 0f 8b  |5..l...>..Q.f...|
000000e0  35 e1 06 6c cd 15 87 3e  ee f8 51 89 66 b7 0f 8b  |5..l...>..Q.f...|
000000f0  35 e1 06 6c cd 15 87 3e  ee f8 51 89 66 b7 0f 8b  |5..l...>..Q.f...|
```

Build a Dictionary

```perl
#!/usr/bin/perl

# This script builds a dictionary for PoisonIvy's Camellia 256bit encryption.
# Usage: poison-dictgen.pl <wordlist >dictionary

use Crypt::Camellia;

# Adjust challenge to your liking.
my $challenge = pack("H32", "00000000000000000000000000000000");

while(<>) {
        chomp;
        my $secret = $_;
        my $key = $secret . chr(0) x (32 - length($secret));

        my $cipher = Crypt::Camellia->new($key);
        my $response = unpack("H16", $cipher->encrypt($challenge));

        printf("%s:%s\n", $secret, $response);
}
```

Guess the Password

```
123:0a900097709a7bc55d8ea3c08b6e67e8
1234:c7be4cc90ea8284ac4b3f6676dbf8d18
12345:6285b591e66c82b6f2416898d6d4abb0
123456:b8daf20e599f34c672bcc98f564088b9
a:9b4aaa9b45e75b7db053ce9a5489f543
aa:82f3bd21c7597b5c138d5a93c061d626
aaa:4eb98cd3503736a0ce4ae30c8b8b04cf
aaaa:8f85170f5ee10f47fbf3e637c2ce1203
aaaaaa:c937b1d5edfa59ffdecf1ac0c34c5ba2
admin:35e1066ccd15873eeef8518966b70f8b
admin1:18d03c27edaa4d845051a808525f2ef1
computer:8260040cd108a24d7ab2265c687bfd37
password:f0ccc0a81c857d24813b08af14e18da2
zaphod:c507d32c412f0100d2f989928f783dfe
zebra:48be2d2bdb15a505e71400fa43bc878b
```

- „admin“ is Poison Ivy‘s default password and it is being used frequently.

A Better Challenge

```
$ cat challenge.bin
GET / HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:2.0.1) Gecko/20100101 Firefox/4.0
If-None-Match: "686897696a7c8"
Accept-Encoding: deflate,gzip2
If-None-Match: "57249921be599"
Accept: text/html,application/xhtml+xml,application/xml;q=1
```

A Better Challenge

```
$ cat challenge.bin
GET / HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:2.0.1) Gecko/20100101 Firefox/4.0
If-None-Match: "686897696a7c8"
Accept-Encoding: deflate,gzip2
If-None-Match: "57249921be599"
Accept: text/html,application/xhtml+xml,application/xml;q=1
```

```
00000000  47 45 54 20 2f 20 48 54  54 50 2f 31 2e 31 0d 0a  |GET / HTTP/1.1..|
00000010  55 73 65 72 2d 41 67 65  6e 74 3a 20 4d 6f 7a 69  |User-Agent: Mozi|
00000020  6c 6c 61 2f 35 2e 30 20  28 57 69 6e 64 6f 77 73  |lla/5.0 (Windows|
00000030  20 4e 54 20 36 2e 31 3b  20 72 76 3a 32 2e 30 2e  | NT 6.1; rv:2.0.|
00000040  31 29 20 47 65 63 6b 6f  2f 32 30 31 30 30 31 30  |1) Gecko/2010010|
00000050  31 20 46 69 72 65 66 6f  78 2f 34 2e 30 20 0d 0a  |1 Firefox/4.0 ..|
00000060  49 66 2d 4e 6f 6e 65 2d  4d 61 74 63 68 3a 20 22  |If-None-Match: "|
00000070  36 38 36 38 39 37 36 39  36 61 37 63 38 22 0d 0a  |686897696a7c8"..|
00000080  41 63 63 65 70 74 2d 45  6e 63 6f 64 69 6e 67 3a  |Accept-Encoding:|
00000090  20 64 65 66 6c 61 74 65  2c 67 7a 69 70 32 0d 0a  | deflate,gzip2..|
000000a0  49 66 2d 4e 6f 6e 65 2d  4d 61 74 63 68 3a 20 22  |If-None-Match: "|
000000b0  35 37 32 34 39 39 32 31  62 65 35 39 39 22 0d 0a  |57249921be599"..|
000000c0  41 63 63 65 70 74 3a 20  74 65 78 74 2f 68 74 6d  |Accept: text/htm|
000000d0  6c 2c 61 70 70 6c 69 63  61 74 69 6f 6e 2f 78 68  |l,application/xh|
000000e0  74 6d 6c 2b 78 6d 6c 2c  61 70 70 6c 69 63 61 74  |tml+xml,applicat|
000000f0  69 6f 6e 2f 78 6d 6c 3b  71 3d 31 20 0d 0a 0d 0a  |ion/xml;q=1 ....|
```

A Better Challenge

```
$ cat challenge.bin
GET / HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:2.0.1) Gecko/20100101 Firefox/4.0
If-None-Match: "686897696a7c8"
Accept-Encoding: deflate,gzip2
If-None-Match: "57249921be599"
Accept: text/html,application/xhtml+xml,application/xml;q=1
```

```
00000000  a4 40 b9 55 fc 60 17 90  ea 01 67 eb 56 87 02 79  |.@.U.`....g.V..y|
00000010  2e 70 2e b0 4a 24 6d 00  d1 3d ae 3d 67 71 ef ca  |.p..J$m..=.=gq..|
00000020  17 bd 25 4f 02 c5 6d 88  33 8b 31 1a 0c 37 eb 03  |..%O..m.3.1..7..|
00000030  87 30 9b f7 a4 ce 70 57  3f 44 52 eb d1 f7 d3 9e  |.0....pW?DR.....|
00000040  4b f7 63 0f b3 dc 58 75  60 dd 09 78 77 36 58 b9  |K.c...Xu`..xw6X.|
00000050  23 6e 83 cf ea 26 3b c2  dd 29 42 2c ba 97 3f 77  |#n...&;..)B,..?w|
00000060  f9 1a a3 29 41 34 05 3f  30 aa 75 59 f1 78 e5 e6  |...)A4.?0.uY.x..|
00000070  39 dc 77 73 76 98 49 a3  42 9b 9a dc b1 fd b7 3b  |9.wsv.I.B......;|
00000080  8a 9a 16 6f 1c b9 60 19  3e a2 65 f7 be 3c 98 13  |...o..`.>.e..<..|
00000090  24 6f 6c 57 2a ba 77 57  30 5b e0 79 f8 dc 29 cf  |$olW*.wW0[.y..).|
000000a0  f9 1a a3 29 41 34 05 3f  30 aa 75 59 f1 78 e5 e6  |...)A4.?0.uY.x..|
000000b0  06 56 9a 90 7b 44 91 41  ae d5 a8 a7 f6 4c f9 e5  |.V..{D.A.....L..|
000000c0  80 bb b0 39 90 77 1d 3e  e0 3c 19 d7 8b 9f 10 37  |...9.w.>.<.....7|
000000d0  94 93 bf 06 27 8f 39 78  a3 e1 fd b2 2b 13 68 4d  |....'.9x....+.hM|
000000e0  60 59 9e 05 a8 98 41 9c  ba c5 f4 e9 81 f7 d3 f9  |`Y....A.........|
000000f0  89 4f b3 2e 45 ac bd c5  fc ba d4 db da da 55 e4  |.O..E.........U.|
```

In Memory

Common Memory Allocation Code

```
mov     esi, [ebp+data_area]
push    PAGE_EXECUTE_READWRITE ; flProtect
push    MEM_COMMIT or MEM_RESERVE ; flAllocation
push    [ebp+Size]          ; usually less than 0x1000 bytes
push    0                   ; lpBaseAddress - let system decide
push    [ebp+hProcess]
call    [esi+VirtualAllocEx]
push    eax

; copy code/data into new allocation
lea     edi, [ebp+lpNumberOfBytesWritten]
push    edi
push    [ebp+Size]
push    [ebp+lpBuffer]
push    eax                 ; lpBaseAddress of new allocation
push    [ebp+hProcess]
call    [esi+WriteProcessMemory]
```

Volatility and Vadinfo

```
$ vol.py -f /samples/winXPPro-228037b9.vmem vadinfo
Volatile Systems Volatility Framework 2.0

************************************************************************
 Pid:   1500
 VAD node @ff3d06c0 Start 5b860000 End 5b8b3fff Tag Vad
 Flags: ImageMap
 Commit Charge: 4 Protection: 7
 ControlArea @80f0eec8 Segment e166ec28
 Dereference list: Flink 00000000, Blink 00000000
 NumberOfSectionReferences:          1 NumberOfPfnReferences:          31
 NumberOfMappedViews:                8 NumberOfUserReferences:          9
 WaitingForDeletion Event:  00000000
 Flags: Accessed, File, HadUserReference, Image
 FileObject @80f0eeec FileBuffer @ e166d630, Name: \WINDOWS\system32\netapi32.dll
 First prototype PTE: e166ec68 Last contiguous PTE: fffffffc
 Flags2: Inherit
 File offset: 00000000
 ...
 VAD node @8100bae8 Start 00160000 End 00160fff Tag VadS
 Flags: MemCommit, PrivateMemory
 Commit Charge: 1 Protection: 6

 VAD node @8100ab40 Start 00020000 End 00020fff Tag VadS
 Flags: MemCommit, PrivateMemory
 Commit Charge: 1 Protection: 4
```

Volatility and Malfind

```
$ vol.py -f /samples/winXPPro-228037b9.vmem malfind --dump-dir tmp
Volatile Systems Volatility Framework 2.0

Name                 Pid    Start      End        Tag       Hits   Protect
IEXPLORE.EXE         1500   0x00160000 0x160fff00 VadS      0      PAGE_EXECUTE_READWRITE
Dumped to: tmp/IEXPLORE.EXE.44e5978.00160000-00160fff.dmp
0x00160000   55 8b ec 81 c4 fc ef ff ff 56 57 c7 85 fc ef ff    U........VW.....
0x00160010   ff 00 10 00 00 8b 7d 08 e8 08 00 00 00 77 69 6e    ......}.......win
0x00160020   69 6e 65 74 00 ff 97 9d 00 00 00 68 de 79 77 b7    inet.......h.yw.
0x00160030   50 50 ff 97 dd 00 00 00 8d 8d fc ef ff ff 51 8d    PP............Q.
0x00160040   b5 00 f0 ff ff 56 6a 26 6a 00 ff d0 85 c0 0f 84    .....Vj&j.......
0x00160050   ae 00 00 00 8b 76 04 56 ff 97 f0 0a 00 00 85 c0    .....v.V........
0x00160060   0f 84 9c 00 00 00 91 8b 55 10 33 c0 83 c0 01 3b    ........U.3....;
0x00160070   c1 75 04 33 c0 eb 1f 80 3c 30 3d 75 ef 33 c0 83    .u.3....<0=u.3..

Disassembly:
00160000: 55                                  PUSH EBP
00160001: 8bec                                MOV EBP, ESP
00160003: 81c4fceffffff                       ADD ESP, 0xffffeffc
00160009: 56                                  PUSH ESI
0016000a: 57                                  PUSH EDI
0016000b: c785fceffffff00100000               MOV DWORD [EBP-0x1004], 0x1000
00160015: 8b7d08                              MOV EDI, [EBP+0x8]
00160018: e808000000                          CALL 0x160025
```

A Simple Signature

Offset 0x40f

Size 8

„StubPath“

Offset 0x418

Size 40

„SOFTWARE
\Classes
\http\shell\open
\command“

Config

RAM

A Simple Signature

Offset 0x40f

Size 8

„StubPath“

Offset 0x418

Size 40

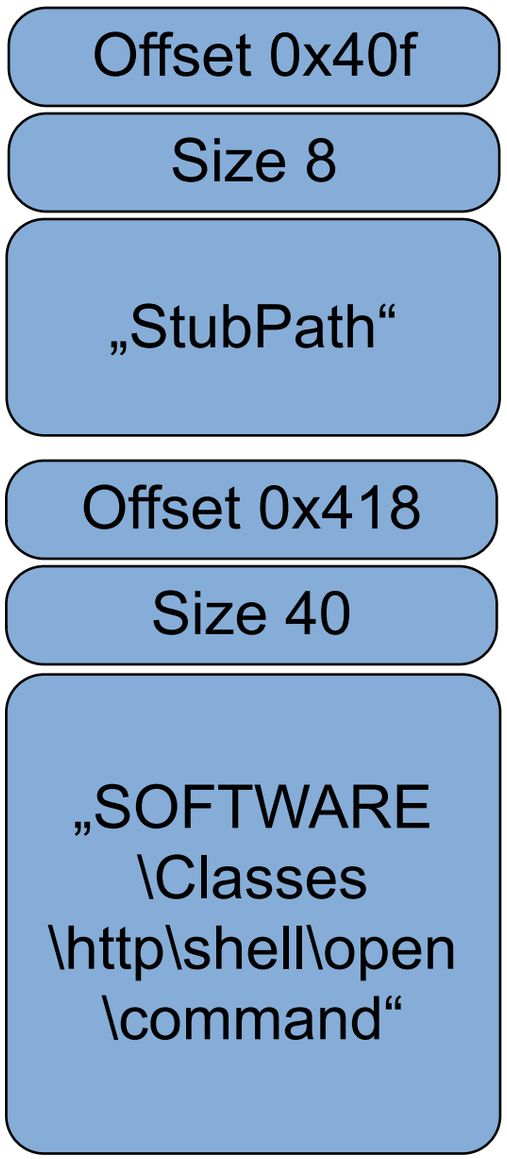
„SOFTWARE
\Classes
\http\shell\open
\command“

Config

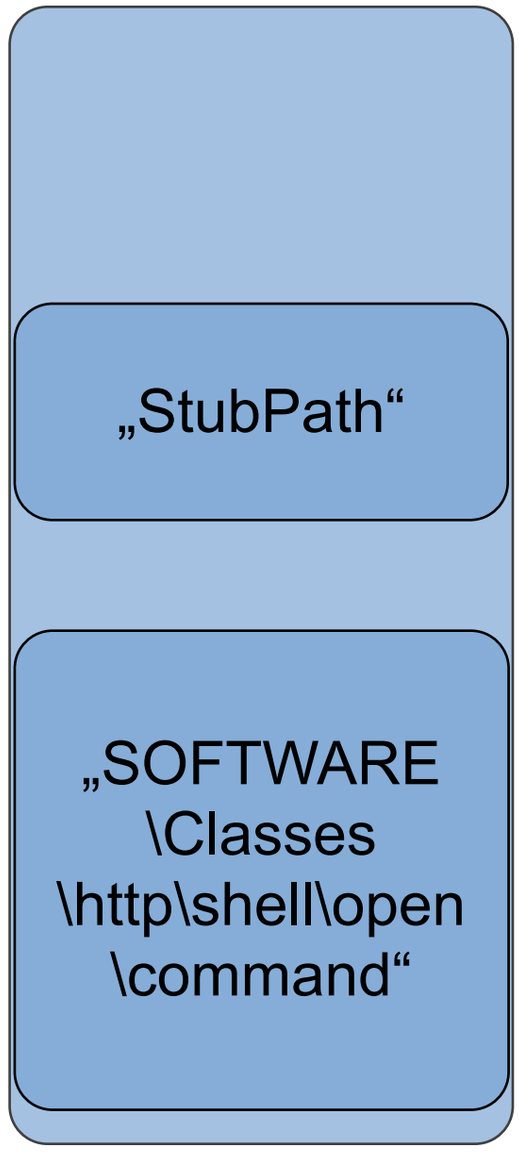
„StubPath“

RAM

A Simple Signature

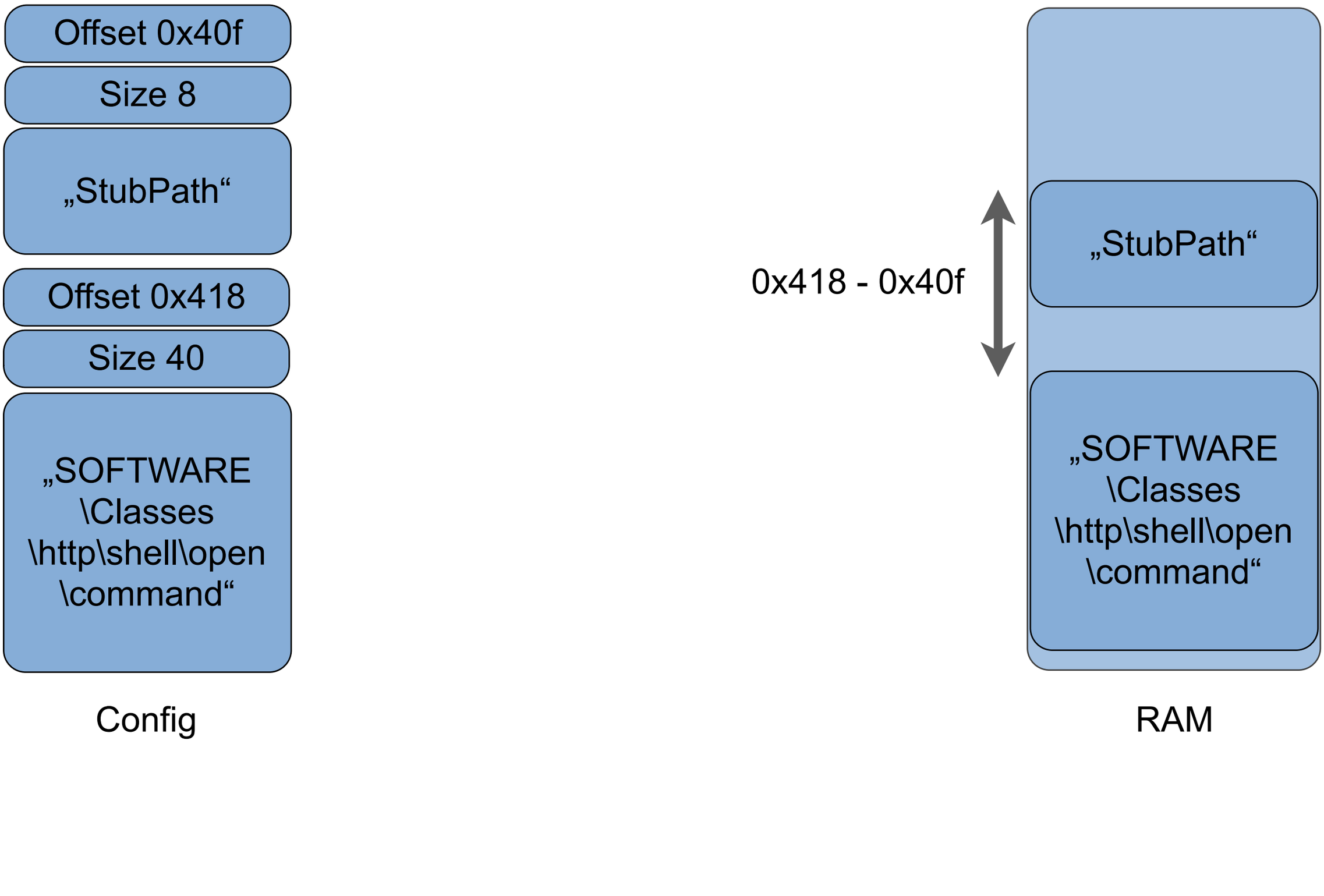


Config

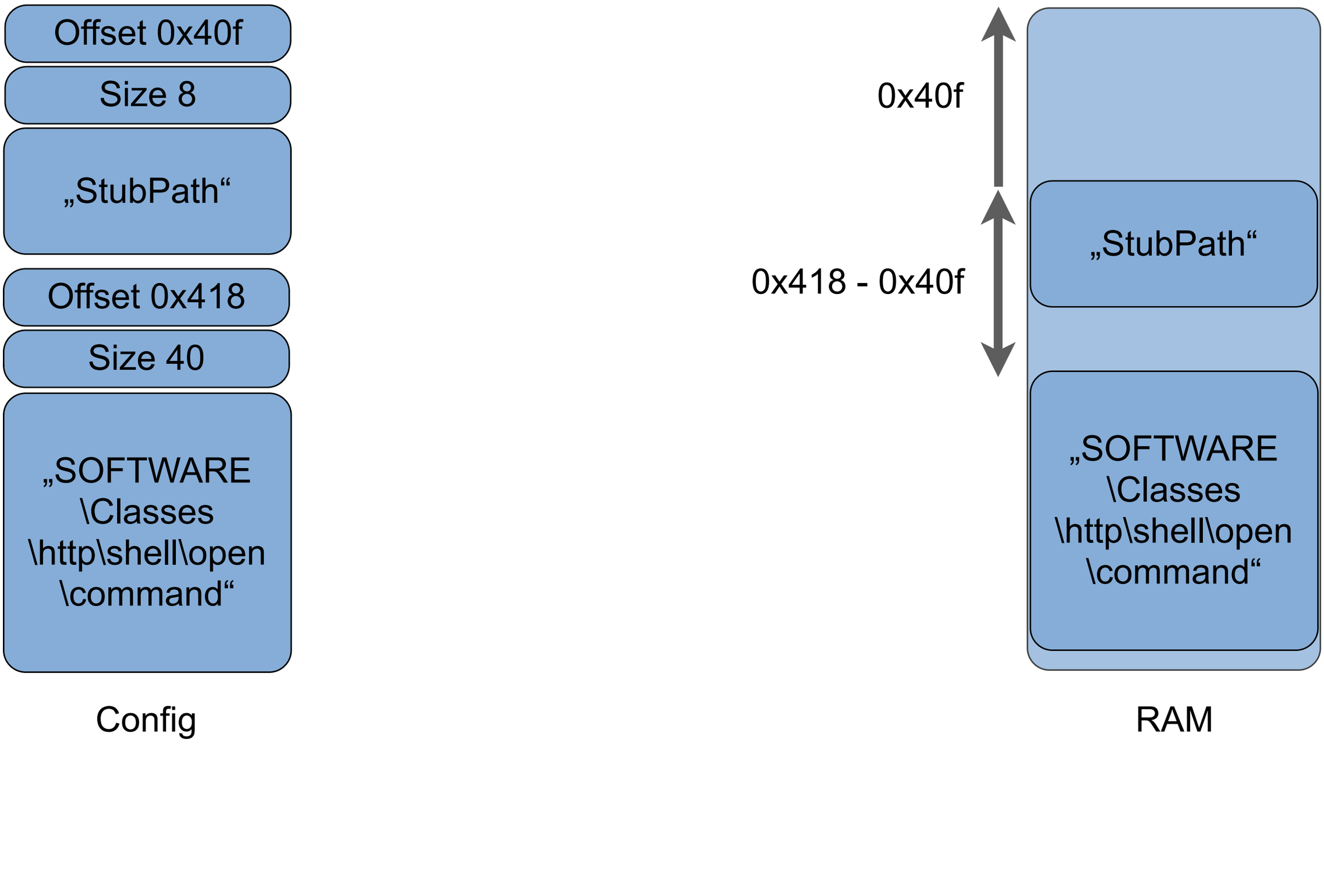


RAM

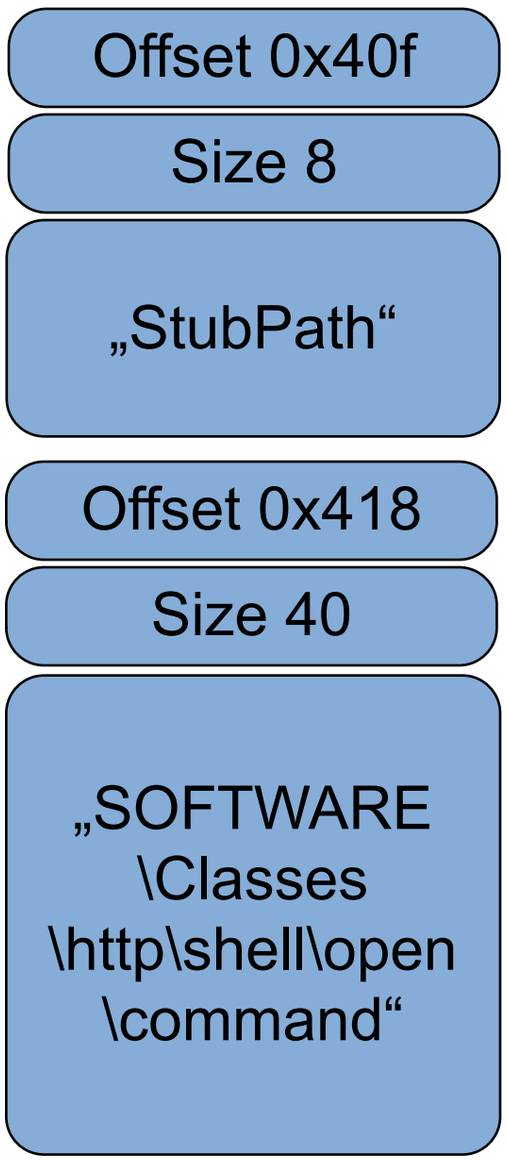
A Simple Signature



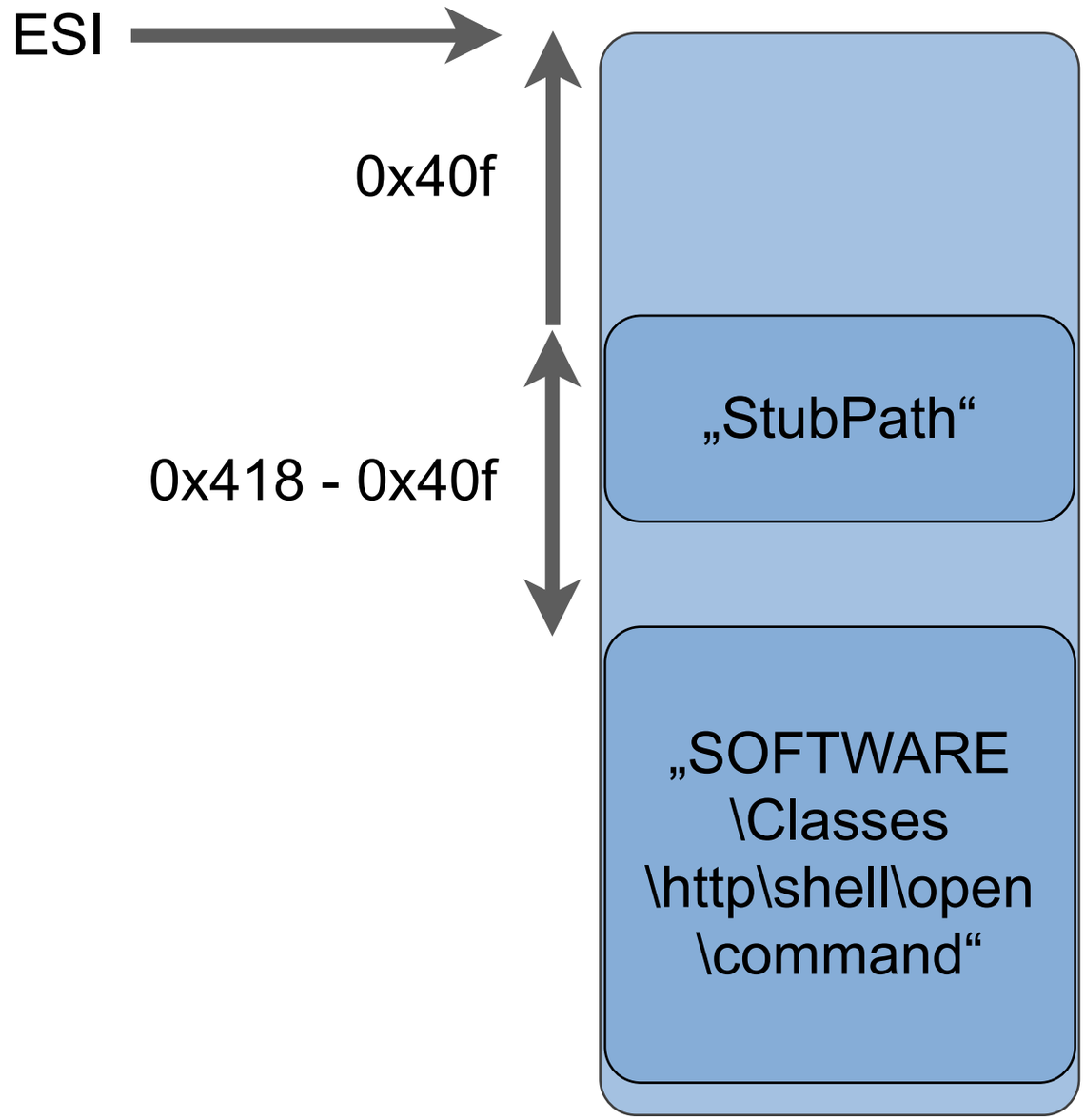
A Simple Signature



A Simple Signature



Config



RAM

A Simple Signature

- Signature based on string constants:
  - at *ESI* + 0x40f: „StubPath“
  - at *ESI* + 0x418: „SOFTWARE\Classes\http\shell\open\command“
  - at *ESI* + 0x456: „Software\Microsoft\Active Setup\Installed Components\“

- Method:
  - search for strings
  - verify proper distance
  - calculate config/variables base address (ESI)
  - parse remaining variables

- This signature is easy to break!
  For a robust signature, search for important code blocks and deduce location of config/variables area.

Volatility and PoisonIvyScan

```
$ vol.py -f /samples/winXPPro-228037b9.vmem poisonivyscan
Volatile Systems Volatility Framework 2.0
Name                 PID    Data VA
-------------------- ------ ----------
explorer.exe            984 0x02630000
```

Volatility and PoisonIvyScan

```
$ vol.py -f /samples/winXPPro-228037b9.vmem poisonivyscan
Volatile Systems Volatility Framework 2.0
Name                 PID    Data VA
-------------------- ------ ----------
explorer.exe            984 0x02630000
```

Volatility and PoisonIvyScan

```
$ vol.py -f /samples/winXPPro-228037b9.vmem poisonivyscan
Volatile Systems Volatility Framework 2.0
Name                 PID    Data VA
-------------------- ------ ----------
explorer.exe            984 0x02630000

$ vol.py -f /samples/winXPPro-228037b9.vmem poisonivyconfig
Volatile Systems Volatility Framework 2.0
------------------------------------------------------------------------
Process: explorer.exe (984)

Infection:
          PoisonIvy has ADMIN privileges!
          Version: 231
          Base VA: 0x013e0000
          Extra VA: 0x02410000
          Data VA: 0x02630000
          Mutex: PImutex
          Original file: C:\Documents and Settings\user\Desktop\full.exe
          Melt original file: ON

Command and Control:
          Host 01: http.example.com:80
          Host 02: http2.example.com:443
          Key (from file): 0x0711b691b42f81d38db2b40b3ff371d4
                           0x63545fcfb5932be1f644cf4269a7c711
          Id: PIid
          Group: PIgroup
```

Volatility and PoisonIvyScan

```
Keylogger:
        Keylogger: ON
        Keylogger TID: 456
        Keylogger Setup: 0x025b0000
        Keylogger Routine: 0x025d0000
        Keylogger logfile: C:\WINDOWS\system32:PI230-sys-a.

Copy file:
        Copy routine: 0x025e0000
        Destination: %WINDIR%\System32:PI230-sys-a.exe

Persistence:
        Active Setup: ON
        Active Setup key: Software\Microsoft\Active Setup\
                Installed Components\{BE019280-219A-75F8-7B01-78AC04665EFD}
        Active Setup name: StubPath
        Setup routine: 0x013f0000
        HKLM Run: ON
        HKLM Run name: PI232-hklm-run
        Setup routine: 0x02400000
```

Volatility and PoisonIvyScan

```
Injector:
        Inject into other processes: ON
        Persistently: ON
        Injector TID: 840
        Injector Routine: 0x02600000
        Target process name: injectvictim.exe
        Target default browser: ON

Proxy:
        Use Proxy: ON
        Persistently: ON
        Host 01: proxy.example.com:8080 (HTTP)
        Host 02: socks.example.com:3460 (SOCKS)
```

References

References

- Related work:
  - Saade, Kurc, and Stewart: „Threat Report: Poison Ivy“. Microsoft Malware Protection Center. October 2011. www.microsoft.com/en-us/download/details.aspx?id=27871
  - Brown: „Detecting Poison Ivy“. McAfee. March 3, 2011. http://hbgary.com/attachments/detectingpoisonivy.pdf
  - Dereszowski, Andrej: „Targeted attacks: From being a victim to counter attacking“. SIGNAL 11. March 15, 2010. http://www.signal11.eu/en/research/articles/targeted_2010.pdf

References

- Poison Ivy file analysis:
  - Sweetscape 010 Editor:
    http://www.sweetscape.com/010editor/
  - Template:
    http://computer.forensikblog.de/files/010_templates/PoisonIvyConfig.bt
    http://r.forens.is/010pi
- Memory analysis:
  - Volatility Memory Analysis Framework:
    http://code.google.com/p/volatility/
  - Malfind by Michael Hale Ligh:
    http://mhl-malware-scripts.googlecode.com/files/malfind.py
  - PoisonIvyScanner Plugin for Volatility:
    http://computer.forensikblog.de/files/volatility_plugins/poison_ivy.py
    http://r.forens.is/volpi

Questions?

Thank You for Your Attention!

Andreas Schuster
a.schuster@yendor.net
http://computer.forensikblog.de/en/