

WIN32/INDUSTROYER

aka. Crashoverride

New threat for industrial control systems (ICS)

https://web-assets.esetstatic.com/wls/2017/06/Win32_Industroyer.pdf

Previous ICS threats

- BlackEnergy – developed by Dmyrtro Oleksiuk in 2007. Was used by Sandworm. Distributed as an attachment in emails. Performed DDoS attacks.
- Stuxnet – developed by Equation Group, uncovered in 2010. Attacked Iranian nuclear power station.
- Havex – developed by Energetic Bear, discovered in 2013.

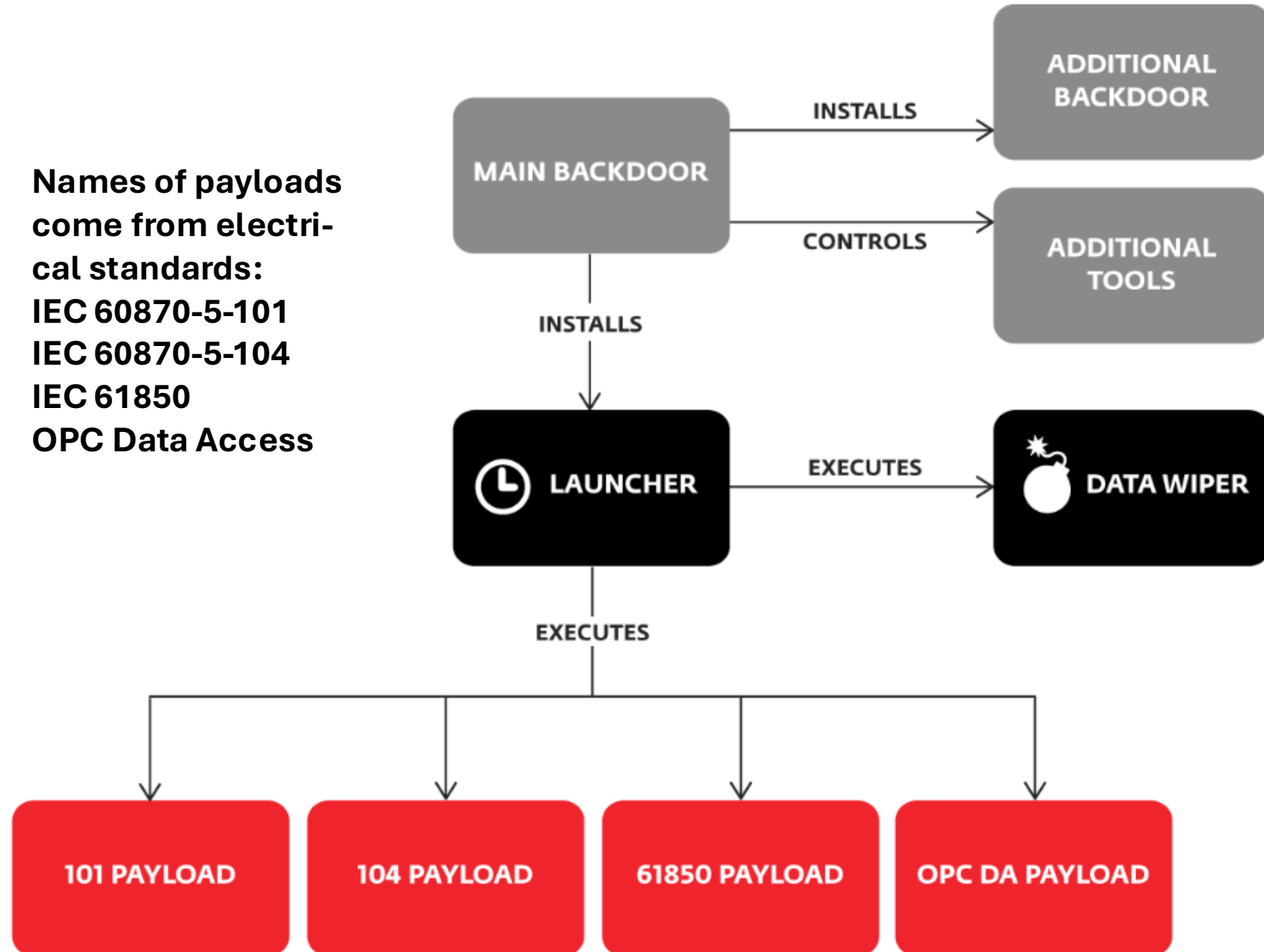
Industroyer was the first one to attack electrical power stations.

General information on Industroyer

- Used (possibly by Sandworm) in cyberattack on Ukraine's power grid on 17th December 2016. It cut fifth of Kyiv's power for one hour
- It's an entire configurable framework for attacking power grids. Attackers had to have broad, specialized knowledge of ICS
- Code is highly obfuscated
- Malware was able to control (electric) switches and circuit breakers. Attackers turned off relays, tried to turn off protective relays and then turn power back on to damage equipment and maybe even injure people. Fortunately, the data packets intended for protective relays were send to the wrong IP address.
- Siemens released an update that fixed these protective relays already in 2015.

High level overview

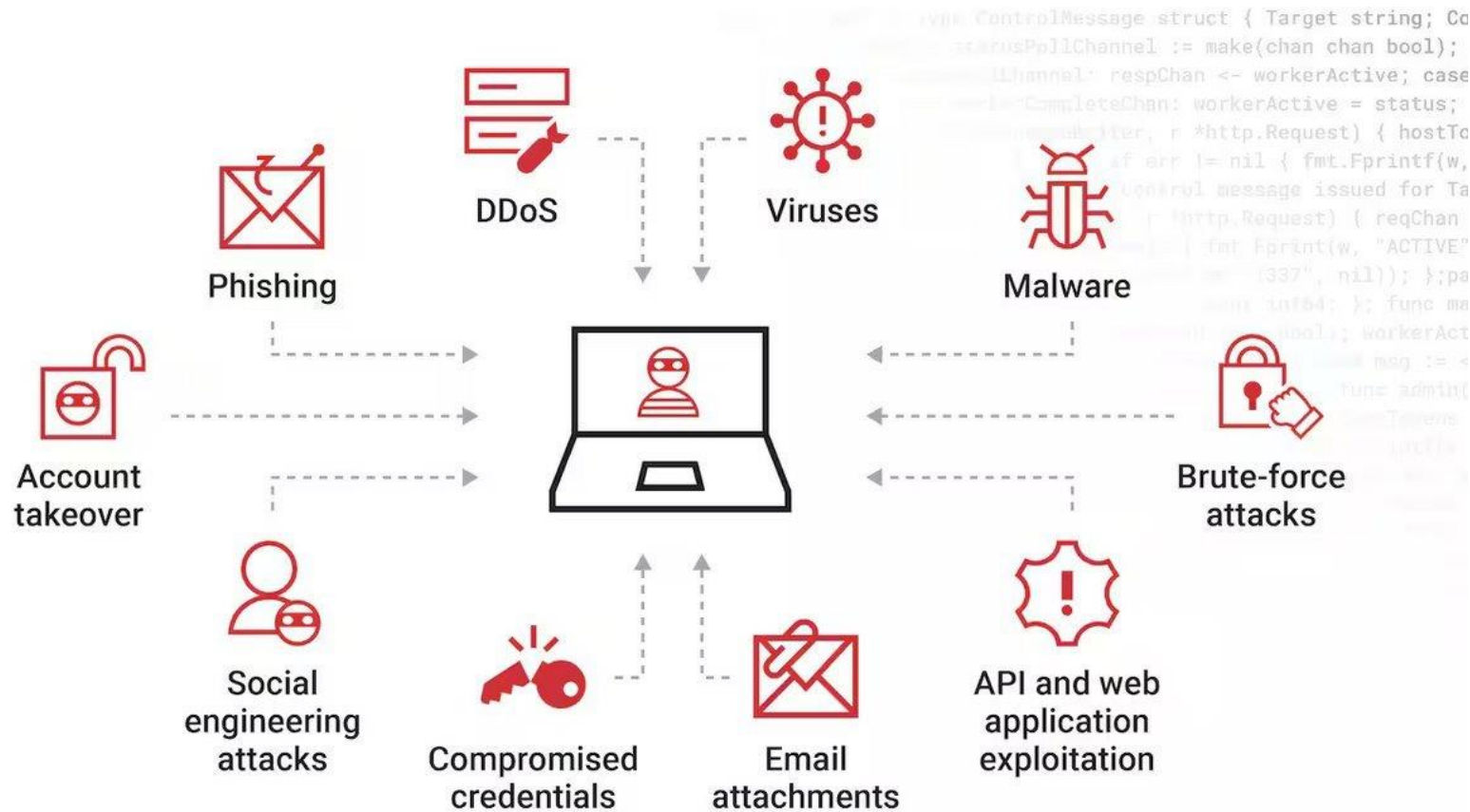
Names of payloads come from electrical standards:
IEC 60870-5-101
IEC 60870-5-104
IEC 61850
OPC Data Access



Vector of attack

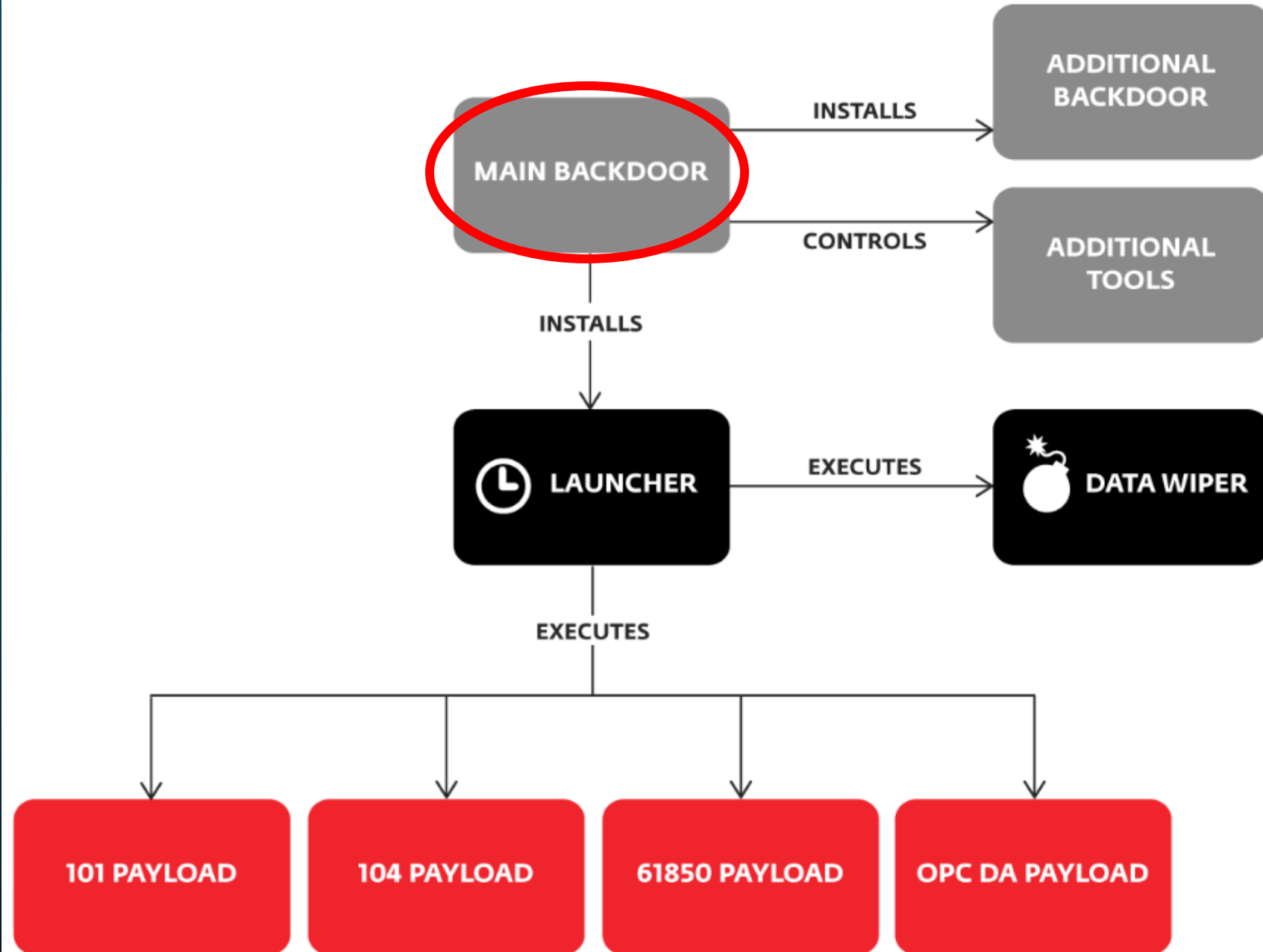
Firstly, the malware had to find a way inside the power grid company's network. Vector of attack is not known, but it could've been any popular one.

Method of gaining high privileges is also not known. (maybe mimikatz?)



Common types of attack vectors

High level overview



Main backdoor

- Attacker had to have admin privileges, to run it as Windows Service
- C&C server uses it to control other components via HTTPS and Tor
- Can be set to be active only on specific hours, although all samples tested were set to work 24/7, so the code looked weird...

```
while ( 1 )
{
    do
    {
        Sleep(dwMilliseconds);
        GetLocalTime(&SystemTime);
    }
    while ( SystemTime.wHour >= 24u );
    c2_connect_and_execute_cmd(&dwMilliseconds);
}
```

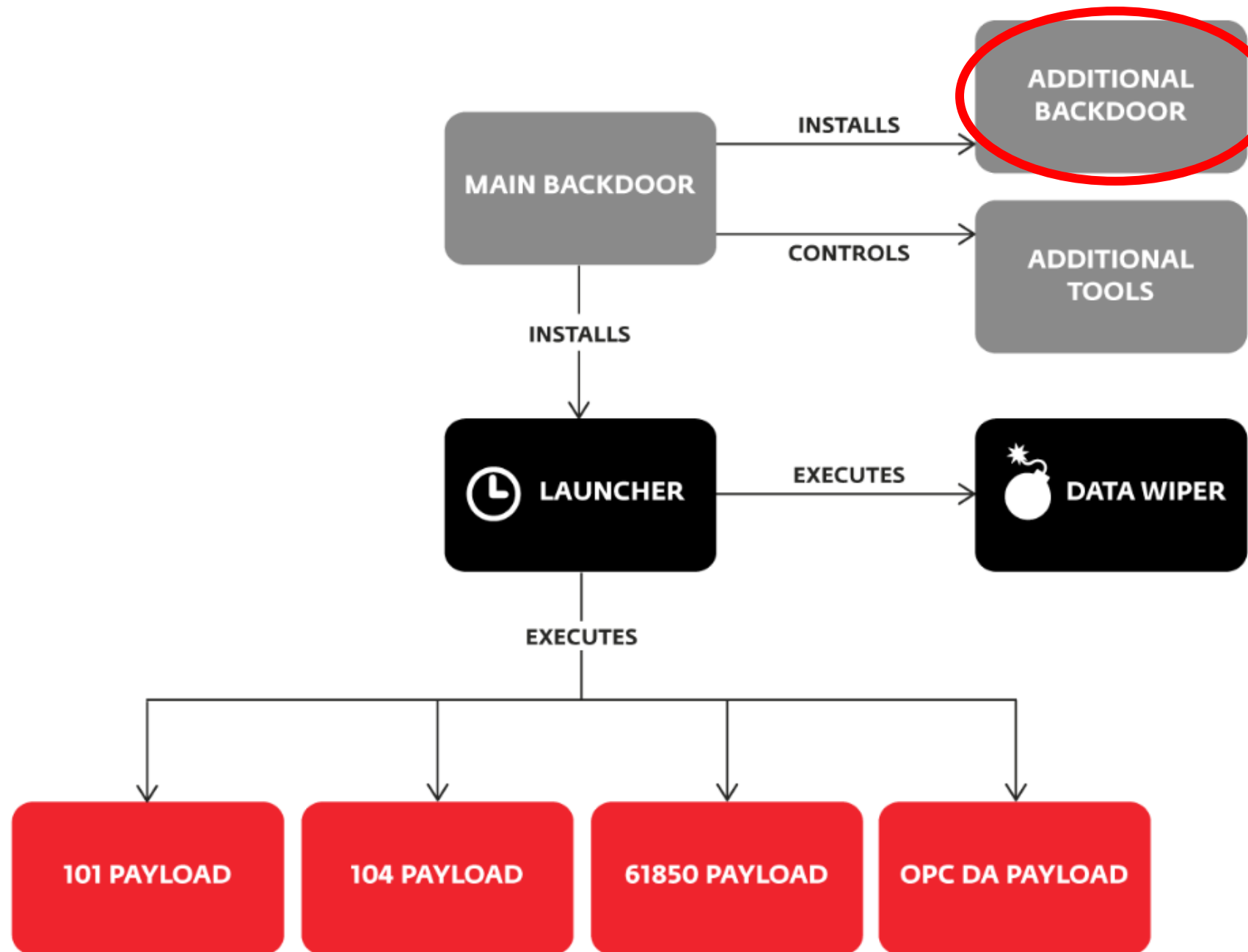
Main backdoor commands

Command ID	Purpose
0	Execute a process
1	Execute a process under a specific user account. Credentials for the account are supplied by the attacker
2	Download a file from C&C server
3	Copy a file

.text:00403FD2	main_func proc near		; CODE XREF: WinMain(x,x,x,x)+14↑p
.text:00403FD2			; .text:004038C4↑p
.text:00403FD2	call	\$+5	
.text:00403FD7			
.text:00403FD7	loc_403FD7:		; CODE XREF: main_func+57↓j
.text:00403FD7			; main_func+5F↓j
.text:00403FD7	add	esp, 4	
.text:00403FDA	push	ebp	; lpOverlapped
.text:00403FDB	mov	ebp, esp	
.text:00403FDD	cmp	edx, 142F9F9Ah	<- machine binding or obfuscation
.text:00403FE3	jz	short loc 404023	
.text:00403FE5	push	ecx	
.text:00403FE6	push	ecx	
.text:00403FE7	mov	eax, [ebp+10h]	
.text:00403FEA	mov	dword_416190, eax	
.text:00403FEF	mov	eax, [ebp+8]	
.text:00403FF2	mov	dword_416194, eax	
.text:00403FF7	mov	eax, [ebp+0Ch]	
.text:00403FFA	cmp	edx, 0B5B93EF3h	<- machine binding or obfuscation
.text:00404000	jz	short loc 404023	
.text:00404002	mov	lpOverlapped, eax	
.text:00404007	mov	[ebp-8], eax	
.text:0040400A	lea	eax, [ebp-8]	
.text:0040400D	push	eax	; lpServiceStartTable
.text:0040400E	mov	dword ptr [ebp-4], offset ServiceMain	
.text:00404015	call	ds:StartServiceCtrlDispatcherW	<- Persistence,
.text:0040401B	xor	al, al	windows service
.text:0040401D	mov	esp, ebp	connecting to Service
.text:0040401F	pop	ebp	Control Manager
.text:00404020	retn		

Peek inside main backdoor's machine code.

High level overview



Additional backdoor

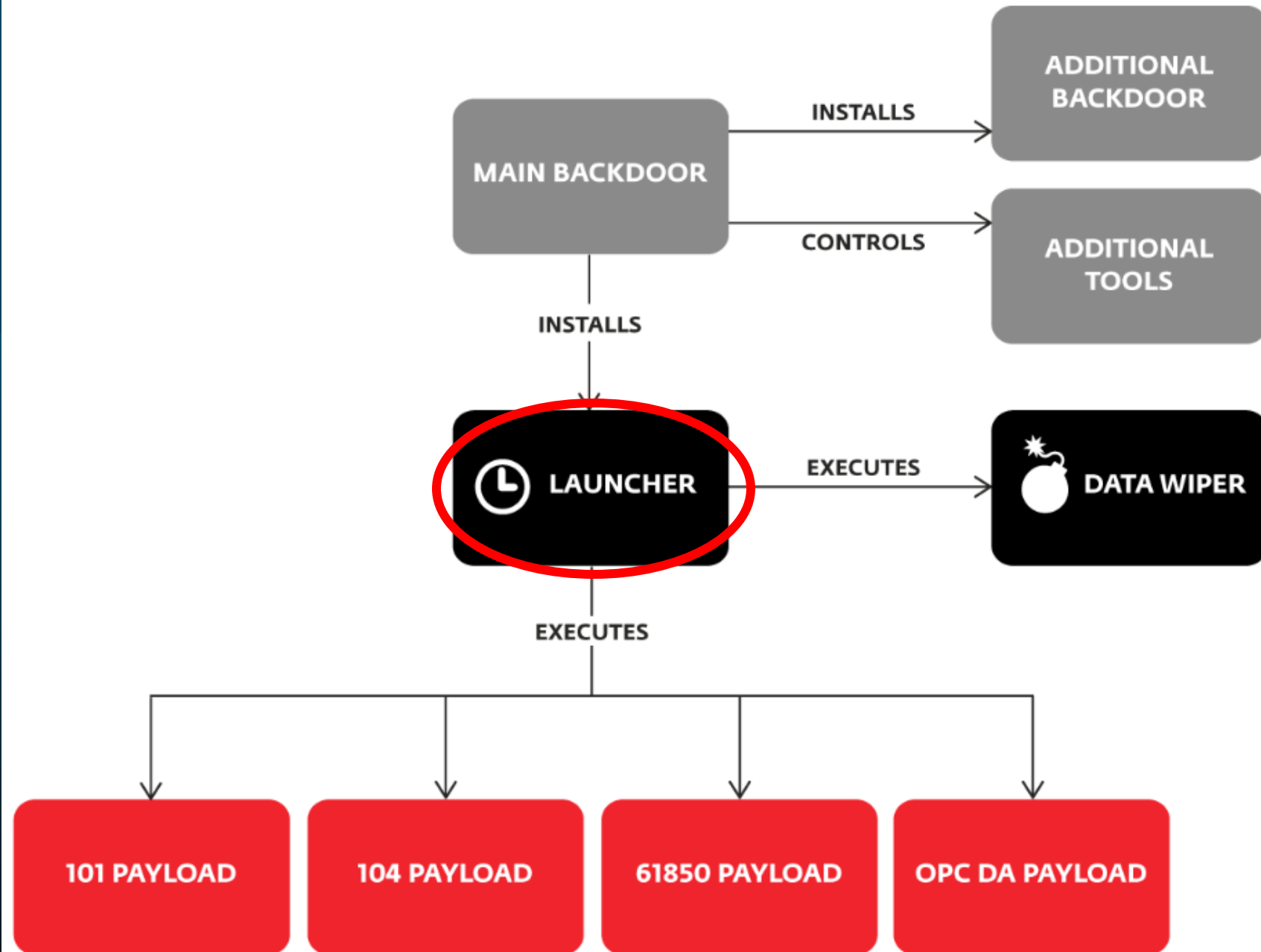
Safety feature (for attackers), in case main backdoor is deleted or disabled.

It's a trojanized version of Windows Notepad. It's fully functional but additionally executes malicious code (in form of a shellcode), which connects to a **different** C&C server and downloads payload.

.text:01004AD5	lea	eax, [ebp+var_50]	.text:01004AD5	lea	eax, [ebp+var_50]
.text:01004AD8	push	eax	.text:01004AD8	push	eax
.text:01004AD9	lea	eax, [ebp+h]	.text:01004AD9	lea	eax, [ebp+h]
.text:01004ADC	push	eax	.text:01004ADC	push	eax
.text:01004ADD	push	0B0h	.text:01004ADD	push	0B0h
.text:01004AE2	push	hWnd	.text:01004AE2	push	hWnd
.text:01004AE8	mov	stru_100A680.lStructSize, 58h	.text:01004AE8	mov	stru_100A680.lStructSize, 58h
.text:01004AF2	mov	stru_100A680.hwndOwner, edx	.text:01004AF2	mov	stru_100A680.hwndOwner, edx
.text:01004AF8	mov	stru_100A680.nMaxFile, 104h	.text:01004AF8	mov	stru_100A680.nMaxFile, 104h
.text:01004B02	mov	stru_100A500.lStructSize, 28h	.text:01004B02	mov	stru_100A500.lStructSize, 28h
.text:01004B0C	mov	stru_100A500.hwndOwner, edx	.text:01004B0C	mov	stru_100A500.hwndOwner, edx
.text:01004B12	call	esi ; SendMessageW	.text:01004B12	call	esi ; SendMessageW
.text:01004B14	push	[ebp+var_50]	.text:01004B14	pusha	
.text:01004B17	push	[ebp+h]	.text:01004B15	pushf	<- start
.text:01004B1A	push	0B1h	.text:01004B16	neg	ebx
.text:01004B1F	push	hWnd	.text:01004B18	shr	eax, 1
.text:01004B25	call	esi ; SendMessageW	.text:01004B1B	dec	ebx
.text:01004B27	push	ebx	.text:01004B1C	mov	eax, 17B200AFh
.text:01004B28	push	ebx	.text:01004B21	mov	edi, 71CFC28h
.text:01004B29	push	0B7h	.text:01004B26	or	edi, dword_10095C7
.text:01004B2E	push	hWnd	.text:01004B2C	xor	esi, 1C779E91h
.text:01004B34	call	esi ; SendMessageW	.text:01004B32	xor	eax, eax
.text:01004B36	push	ebx	.text:01004B34	dec	edi
.text:01004B37	call	ds:GetKeyboardLayout	.text:01004B35	rol	esi, 5
.text:01004B3D	and	ax, 3FFh	.text:01004B38	and	esi, edi
.text:01004B41	cmp	ax, 11h	.text:01004B3A	and	esi, edi
.text:01004B45	jnz	short loc_1004B58	.text:01004B3C	rol	edx, 6
.text:01004B47	push	1	.text:01004B3F	neg	eax
.text:01004B49	push	1	.text:01004B41	xor	esi, eax
.text:01004B4B	push	0D8h	.text:01004B43	neg	ebx
.text:01004B50	push	hWnd	.text:01004B45	shr	ebx, 5
.text:01004B56	call	esi ; SendMessageW	.text:01004B48	mov	ecx, 5E95422h

Figure 4. Comparison between original Notepad binary code (at the left) and backdoored binary code.

High level overview



Launcher

Separate executable responsible for launching the payloads and the Data wiper component at specific dates, which were: 17th and 20th December of 2016.

Starts two highest priority threads:

1. Loads a payload DLL (name provided inside main backdoor's command, not hardcoded)
2. Waits 1 or 2 hours and loads Data wiper

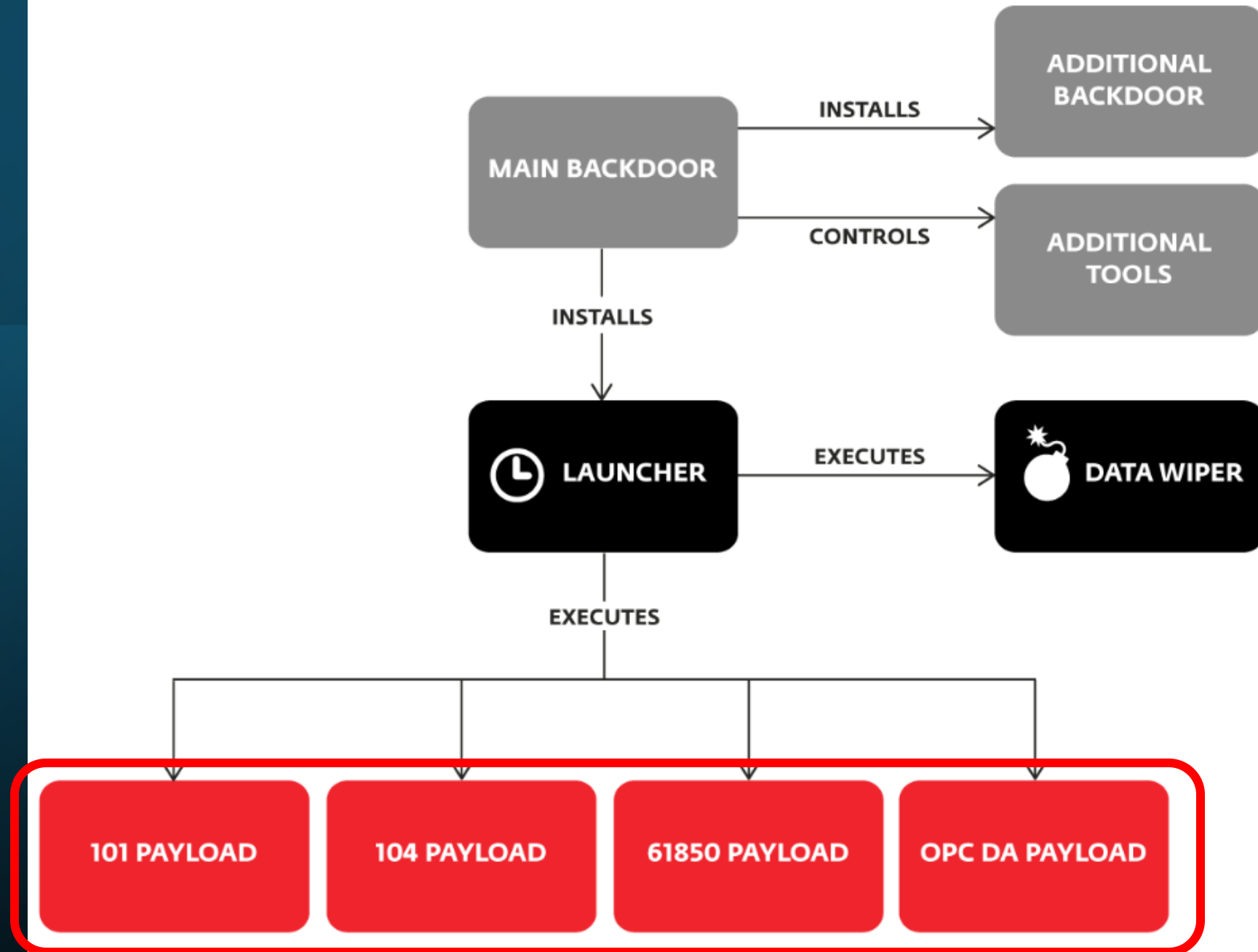
```

;
; Export directory for Crash101.dll
;
                dd 0                ; Characteristics
                dd 5855F8EDh         ; TimeDateStamp: Sun Dec 18 02:48:13 2016
                dw 0                ; MajorVersion
                dw 0                ; MinorVersion
                dd rva aCrash101_dll ; Name
                dd 1                ; Base
                dd 1                ; NumberOfFunctions
                dd 1                ; NumberOfNames
                dd rva off_100355F8  ; AddressOfFunctions
                dd rva off_100355FC  ; AddressOfNames
                dd rva word_10035600 ; AddressOfNameOrdinals
;
; Export Address Table for Crash101.dll
;
off_100355F8    dd rva Crash        ; DATA XREF: .rdata:100355EC↑o
;
; Export Names Table for Crash101.dll
;
off_100355FC    dd rva aCrash       ; DATA XREF: .rdata:100355F0↑o
;                                     ; "Crash"
;
; Export Ordinals Table for Crash101.dll
;
word_10035600   dw 0                ; DATA XREF: .rdata:100355F4↑o
aCrash101_dll   db 'Crash101.dll',0 ; DATA XREF: .rdata:100355DC↑o
aCrash          db 'Crash',0        ; DATA XREF: .rdata:off_100355FC↑o

```

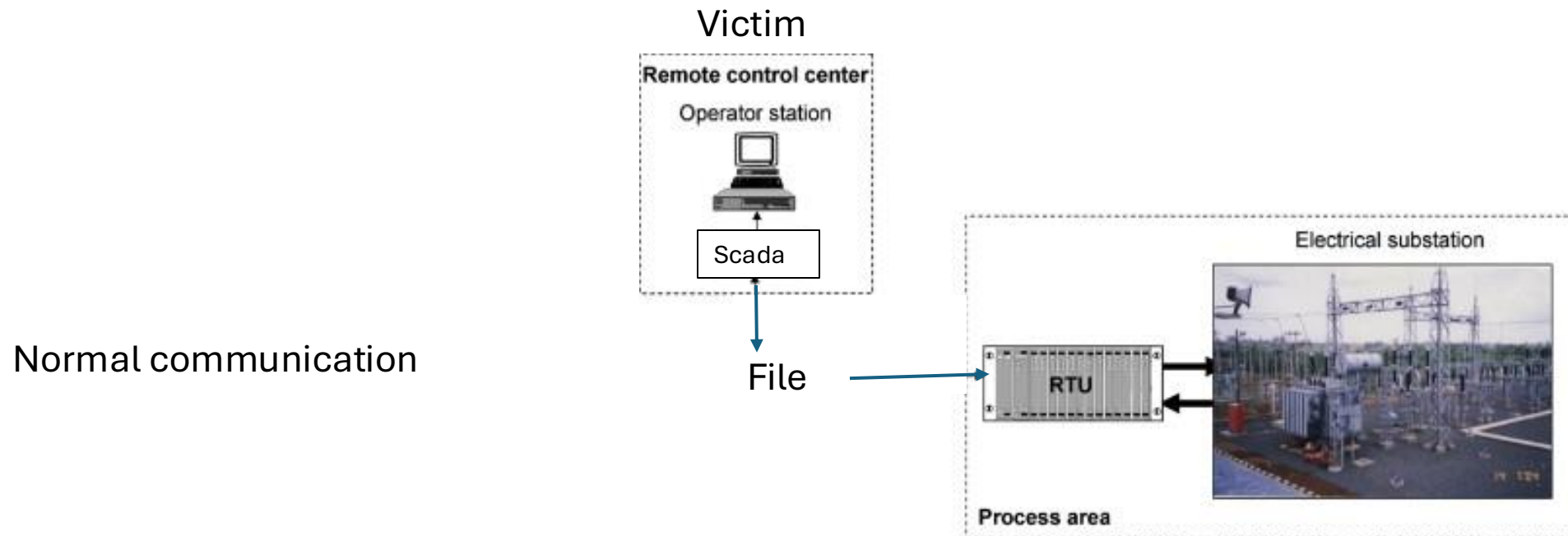
Figure 5. Example payload DLL that has internal name `Crash101.dll` and `Crash` export function.

High level overview



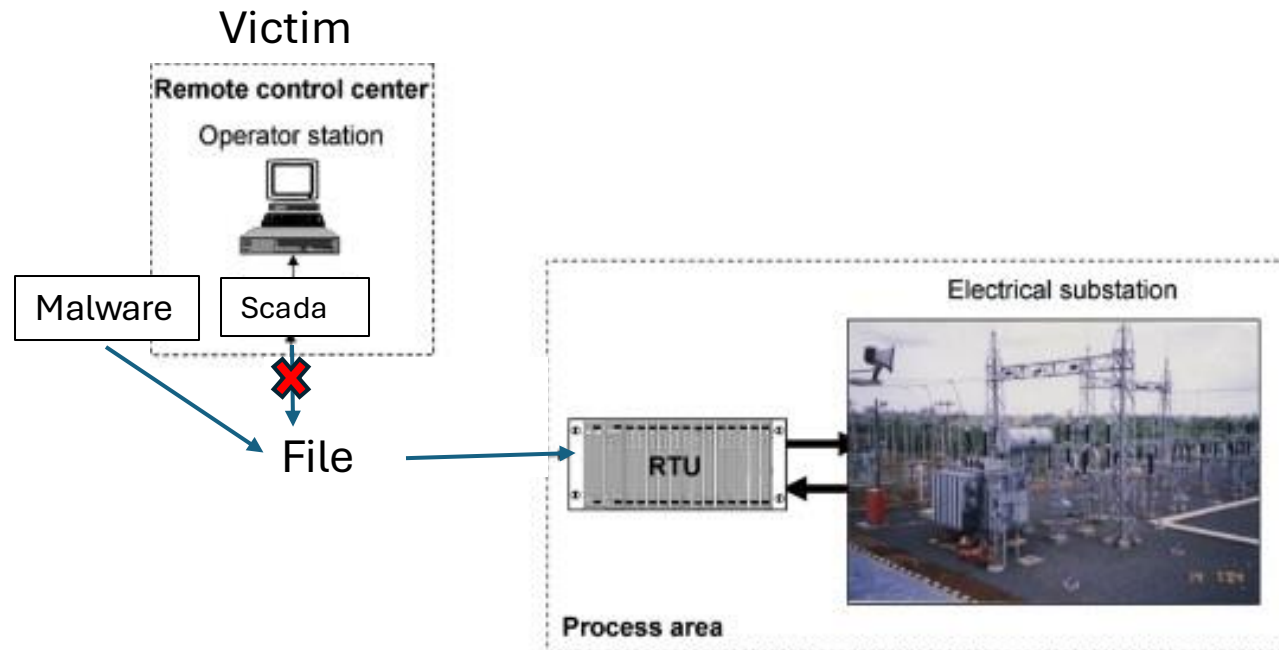
101 payload component

IEC 60870-5-101 – used for monitoring, controlling and communication with electric power systems (serial connection).



101 payload component

Attacker terminates the victim's app's connection with file that controls RTU, and instead communicates with file directly with standard WriteFile, ReadFile of Window's API. Uses only COM1 port, but opens also 2nd and 3rd, so that they're blocked.



101 payload component

Payload iterates through all IOAs (input output devices) and performs three stages:

1. Switch IOAs to Off state
2. Switch IOAs to On state
3. Switch IOAs to Off state



RTU

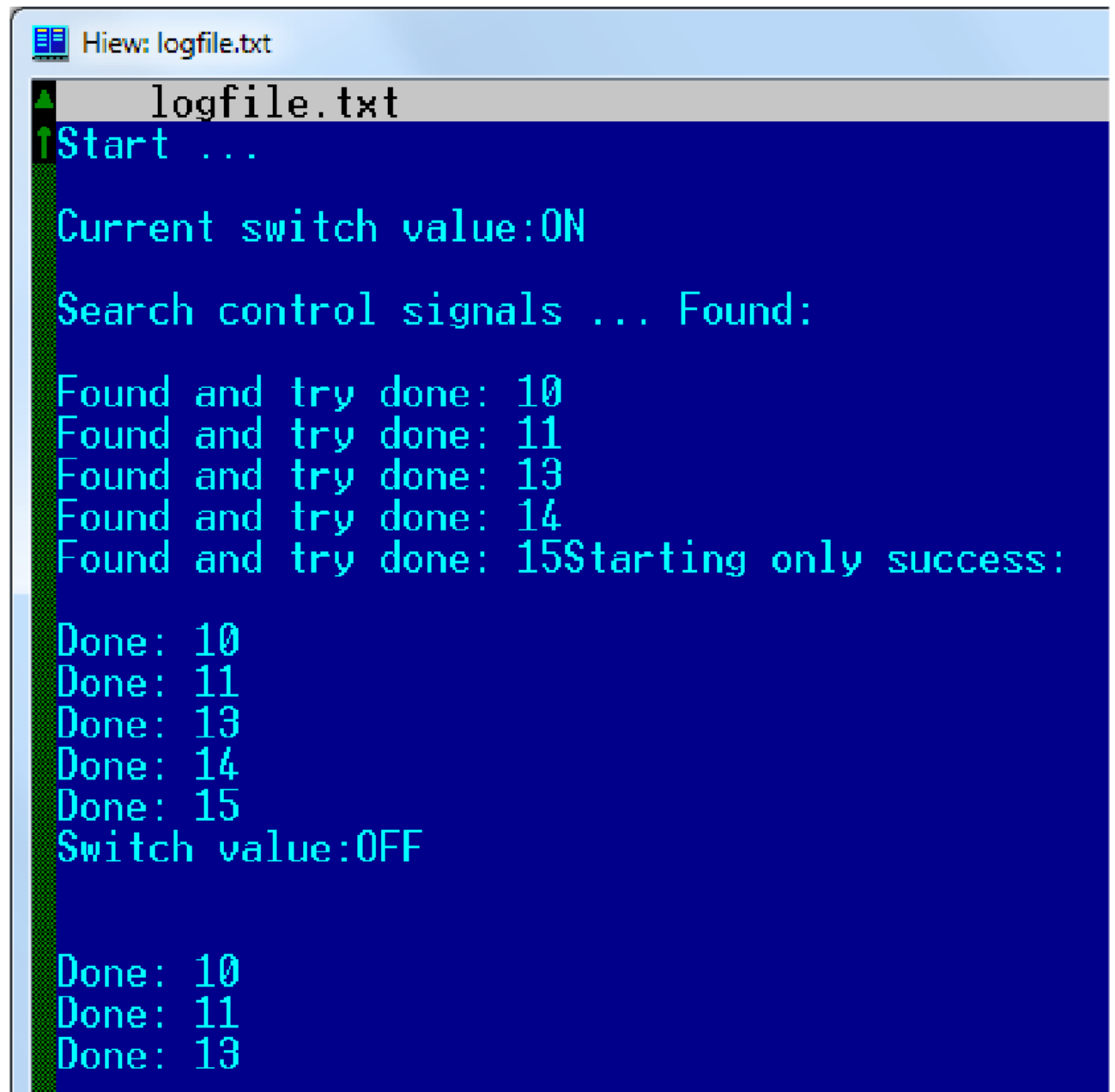
104 payload component

IEC 60870-5-104 extends 101, so that the protocol can be transmitted over TCP/IP (and it's not very safe).

Payload, like others, is configurable, so that the malware can be used on different infrastructures. Example options:

Property	Expected value	Purpose
target_ip	IP address	The IP address that will be used for the communication using IEC 104 protocol standard
target_port	Port number	Self-explanatory

104 payload at work



```
View: logfile.txt
logfile.txt
Start ...
Current switch value:ON
Search control signals ... Found:
Found and try done: 10
Found and try done: 11
Found and try done: 13
Found and try done: 14
Found and try done: 15Starting only success:
Done: 10
Done: 11
Done: 13
Done: 14
Done: 15
Switch value:OFF
Done: 10
Done: 11
Done: 13
```

Finds IOAs and then constantly turns them on and off.

61850 payload component

"This standard describes a protocol used for multivendor communication among devices that perform protection, automation, metering, monitoring, and control of electrical substation automation systems. *The protocol is very complex and robust*"

The component is a standalone malicious tool composed of .exe file and a dll.

61850 payload component

After execution:

1. Looks for i.ini file (with IPs). If it doesn't exist, scans network for all IPs and tries to connect to port 102 on each
2. Sends requests (using Connection Oriented Transport and Manufacturing Message Specification protocols) and gathers information about circuit breakers and switches

Information then can be sent to C&C server, so it can be used in other components or future attacks.

OPC DA payload component

OPC Data Access is a group of client-server standards for communicating real time data from devices like PLC to SCADA, ERP systems etc.

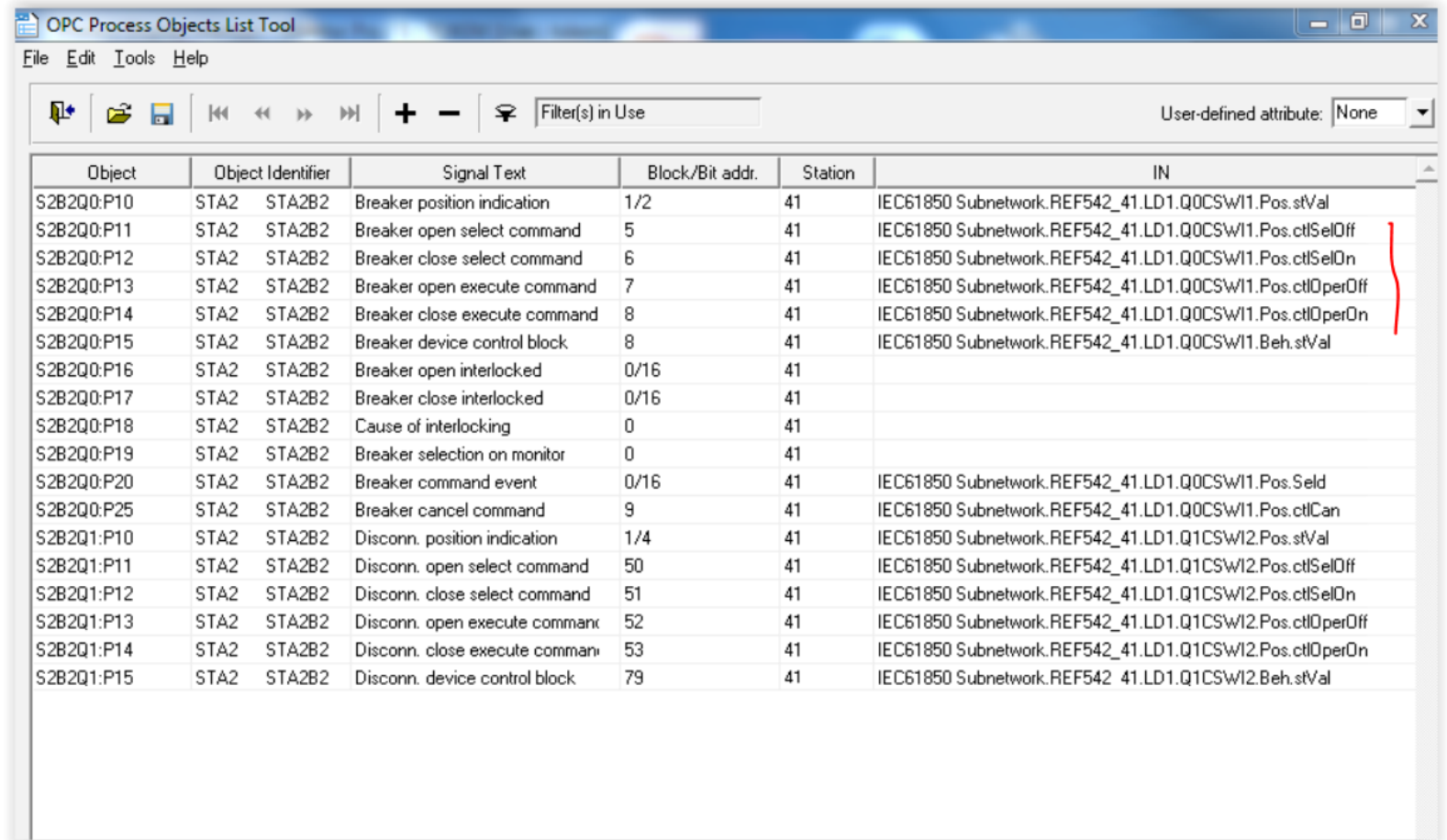
Once executed it enumerates all OPC servers and looks for items that contain: ctlSelOn, ctlOperOn, ctlSelOff, ctlOperOff, stVal, which means that the attackers are interested in OPC servers from ABB (microSCADA)



OPC DA payload component

When adding a new OPC group, attackers use name Abdul, which might be a slang term for ABB soltions.

```
; esi szName
; This
aAbdul_0:
    unicode 0, <Abdul>,0
```



The screenshot shows the 'OPC Process Objects List Tool' window. It contains a table with the following columns: Object, Object Identifier, Signal Text, Block/Bit addr., Station, and IN. The table lists various OPC items, including breaker position indications, select commands, execute commands, device control blocks, interlocked commands, cause of interlocking, selection on monitor, command events, cancel commands, disconnection position indications, disconnection select commands, disconnection close select commands, disconnection open execute commands, disconnection close execute commands, and disconnection device control blocks. The IN field contains long strings representing the OPC item names, such as 'IEC61850 Subnetwork.REF542_41.LD1.Q0CSW11.Pos.stVal'.

Object	Object Identifier	Signal Text	Block/Bit addr.	Station	IN
S2B2Q0:P10	STA2 STA2B2	Breaker position indication	1/2	41	IEC61850 Subnetwork.REF542_41.LD1.Q0CSW11.Pos.stVal
S2B2Q0:P11	STA2 STA2B2	Breaker open select command	5	41	IEC61850 Subnetwork.REF542_41.LD1.Q0CSW11.Pos.ctlSelOff
S2B2Q0:P12	STA2 STA2B2	Breaker close select command	6	41	IEC61850 Subnetwork.REF542_41.LD1.Q0CSW11.Pos.ctlSelOn
S2B2Q0:P13	STA2 STA2B2	Breaker open execute command	7	41	IEC61850 Subnetwork.REF542_41.LD1.Q0CSW11.Pos.ctlOperOff
S2B2Q0:P14	STA2 STA2B2	Breaker close execute command	8	41	IEC61850 Subnetwork.REF542_41.LD1.Q0CSW11.Pos.ctlOperOn
S2B2Q0:P15	STA2 STA2B2	Breaker device control block	8	41	IEC61850 Subnetwork.REF542_41.LD1.Q0CSW11.Beh.stVal
S2B2Q0:P16	STA2 STA2B2	Breaker open interlocked	0/16	41	
S2B2Q0:P17	STA2 STA2B2	Breaker close interlocked	0/16	41	
S2B2Q0:P18	STA2 STA2B2	Cause of interlocking	0	41	
S2B2Q0:P19	STA2 STA2B2	Breaker selection on monitor	0	41	
S2B2Q0:P20	STA2 STA2B2	Breaker command event	0/16	41	IEC61850 Subnetwork.REF542_41.LD1.Q0CSW11.Pos.SelD
S2B2Q0:P25	STA2 STA2B2	Breaker cancel command	9	41	IEC61850 Subnetwork.REF542_41.LD1.Q0CSW11.Pos.ctlCan
S2B2Q1:P10	STA2 STA2B2	Disconn. position indication	1/4	41	IEC61850 Subnetwork.REF542_41.LD1.Q1CSW12.Pos.stVal
S2B2Q1:P11	STA2 STA2B2	Disconn. open select command	50	41	IEC61850 Subnetwork.REF542_41.LD1.Q1CSW12.Pos.ctlSelOff
S2B2Q1:P12	STA2 STA2B2	Disconn. close select command	51	41	IEC61850 Subnetwork.REF542_41.LD1.Q1CSW12.Pos.ctlSelOn
S2B2Q1:P13	STA2 STA2B2	Disconn. open execute command	52	41	IEC61850 Subnetwork.REF542_41.LD1.Q1CSW12.Pos.ctlOperOff
S2B2Q1:P14	STA2 STA2B2	Disconn. close execute command	53	41	IEC61850 Subnetwork.REF542_41.LD1.Q1CSW12.Pos.ctlOperOn
S2B2Q1:P15	STA2 STA2B2	Disconn. device control block	79	41	IEC61850 Subnetwork.REF542_41.LD1.Q1CSW12.Beh.stVal

Figure 15. An example of OPC items names in IN field received using OPC Process Objects List Tool.

OPC DA payload component

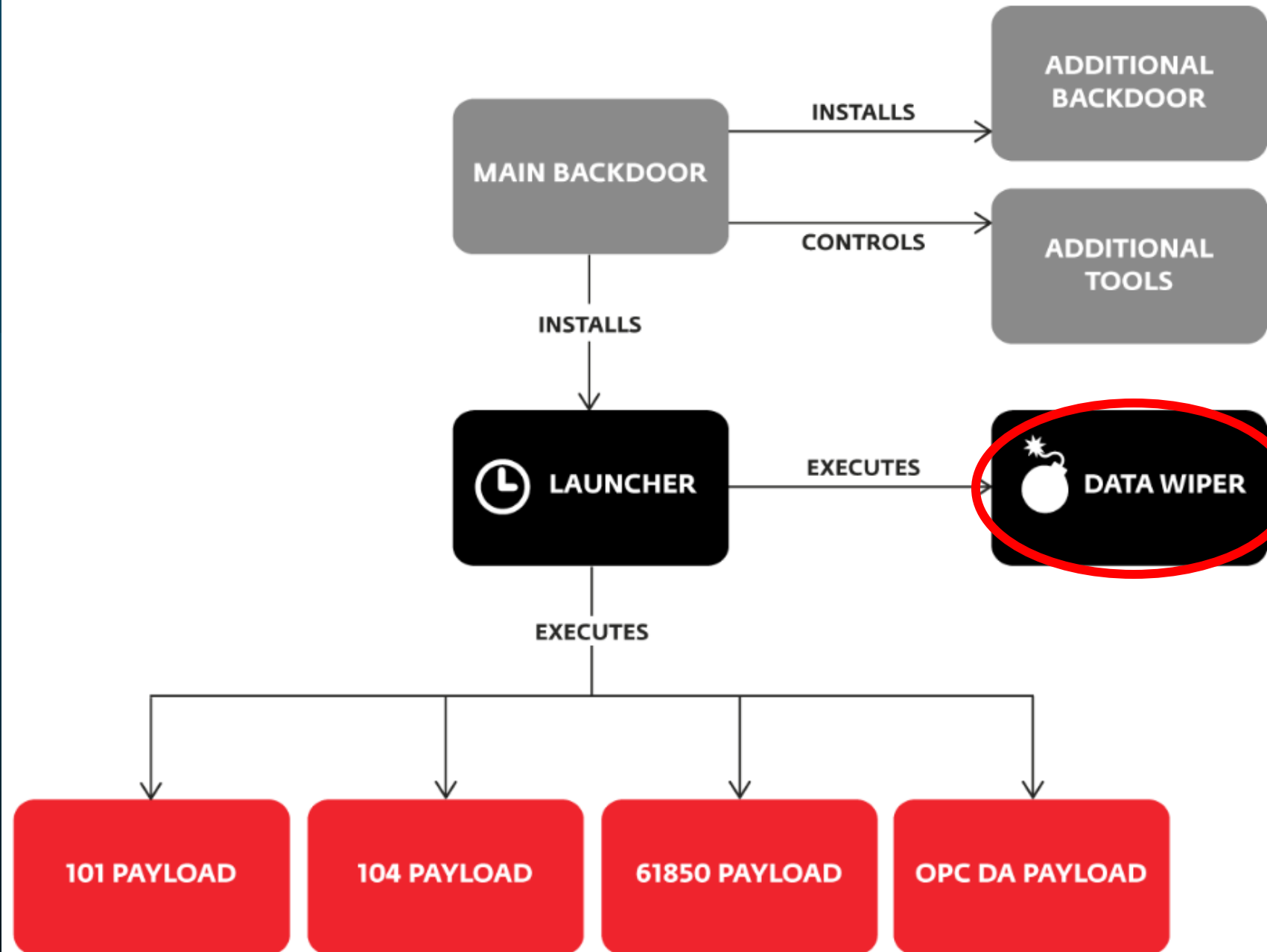
It also tries to change state of discovered items, by writing 0x01 value, using IOPCSyncIO interface.

Then sends gathered information to a log file.

.text:004034FE	mov	eax, VT_I2
.text:00403503	mov	word ptr [ebp+pItemValues.anonymous_0], ax
.text:0040350A	mov	eax, 1
.text:0040350F	mov	word ptr [ebp+pItemValues.anonymous_0+8], ax
.text:00403516	lea	eax, [ebp+pItemValues]
.text:0040351C	push	eax ; pItemValues
.text:0040351D	mov	eax, [ebp+OPC_items]
.text:00403523	mov	ecx, [eax+esi*4]
.text:00403526	call	IOPCSyncIO_Write
.text:0040352B	cmp	esi, edi
.text:0040352D	jb	short loc_403539
.text:0040352F	push	80070057h
.text:00403534	call	throw_exception

Figure 17. Disassembled code of OPC DA payload that uses IOPCSyncIO interface.

High level overview



Data wiper component

Used in final stage of attack to hide attacker's tracks and make recovery difficult.

Named haslo.dat or haslo.exe

Data wiper component

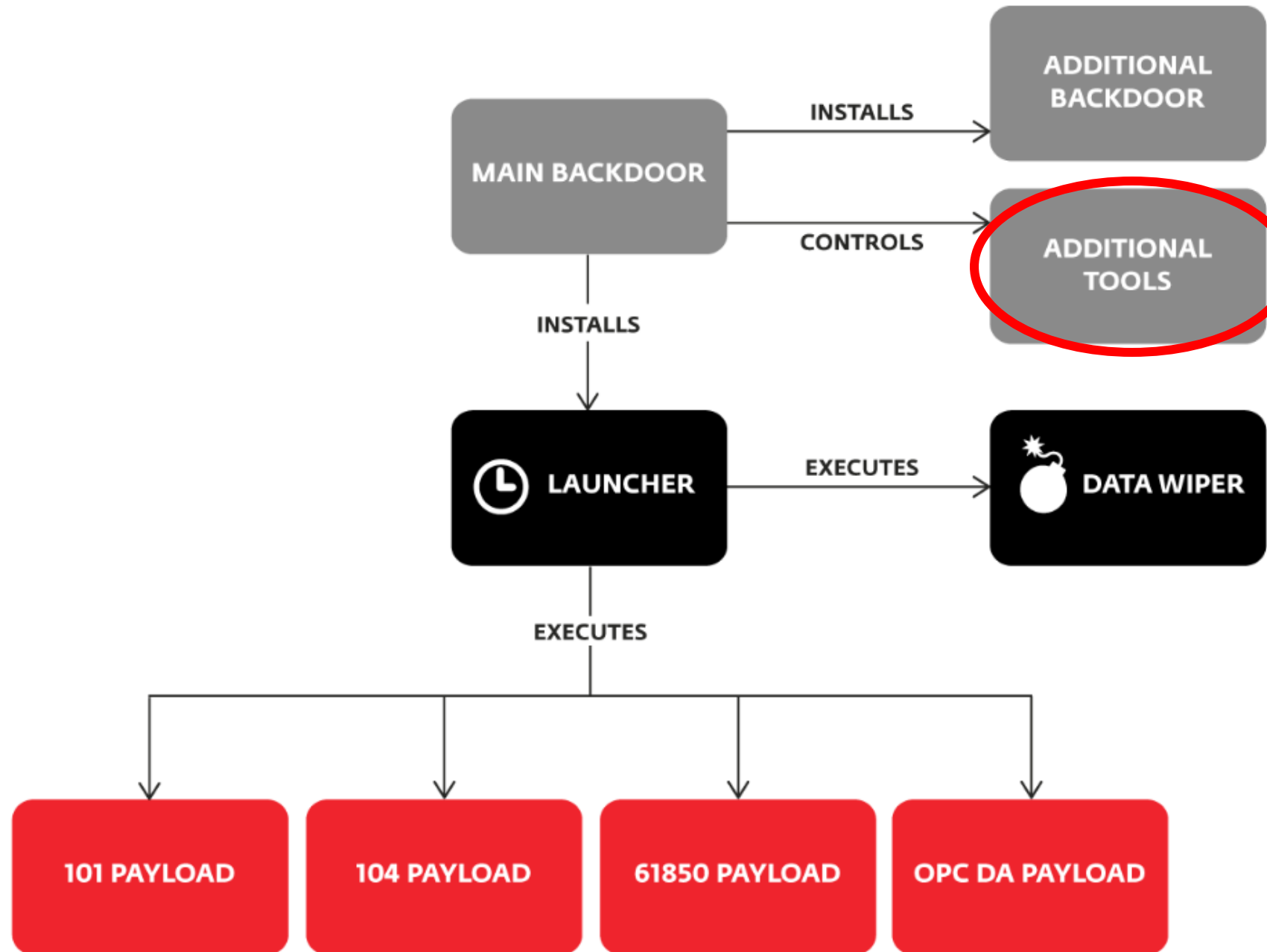
1. In "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services" it changes the ImagePath value to an empty string. This makes the system unbootable
2. Files with specific file extensions are deleted from drives C:\ to Z:\
3. Rewrites files' content with meaningless data
4. Terminates all processes except itself. System becomes unresponsive and crashes

Data wiper component

File extensions meant for deletion. The uncommon ones are files used in ICS

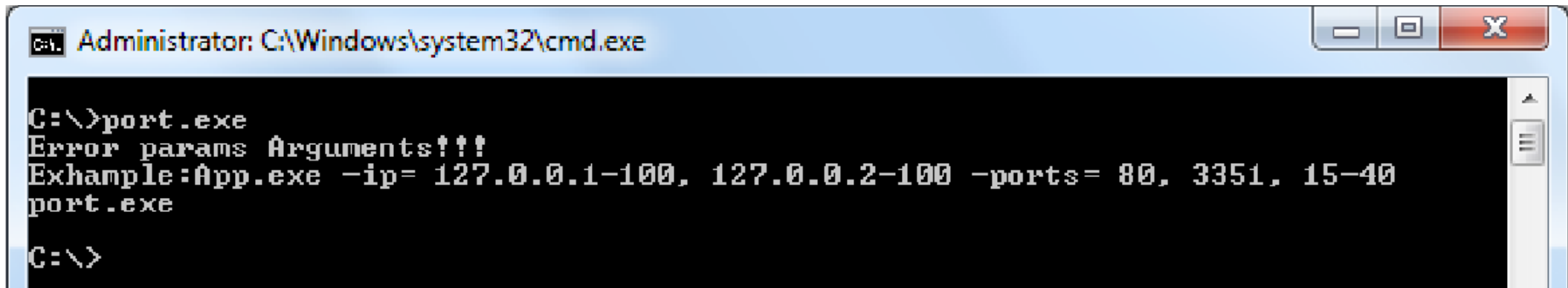
SYS_BASCON.COM	*.pcmi	*.bk
*.v	*.pcmt	*.bkp
*.PL	*.ini	*.log
*.paf	*.xml	*.zip
*.XRF	*.CIN	*.rar
*.SCL	*.cxm	*.7z
*.bak	*.elb	*.exe
*.cid	*.epi	*.dll
*.scd	*.mdf	
*.pcmp	*.ldf	

High level overview



Additional tools

- Self-written port scanner tool



The screenshot shows a Windows command prompt window titled "Administrator: C:\Windows\system32\cmd.exe". The command prompt displays the following text:

```
C:\>port.exe
Error params Arguments!!!
Example:App.exe -ip= 127.0.0.1-100, 127.0.0.2-100 -ports= 80, 3351, 15-40
port.exe
C:\>
```

Figure 19. The port scanner tool usage example.

- DoS tool targeting Siemens SIPROTEC devices, using CVE-2015-5374 exploit (this is how protective relays were supposed to be disabled). It sent 18-bytes UDP packets to port 50000 to a hardcoded IP.

Conclusion

The commonly-used industrial control protocols used in this malware were designed decades ago without taking security into consideration. Therefore, any intrusion into an industrial network with systems using these protocols should be considered as “game over”.