新型 ICS 攻击框架“TRITON”导致工业系统运营中断技术分析与防护方案

发布时间：2017 年 12 月 29 日

综述

最近一起针对关键基础设施的攻击事件中，攻击者通过部署恶意软件来操作工业安全系统。目标系统为工业流程提供了紧急关闭功能。相关证据表明，攻击者在研发破坏物理装置功能的过程中，无意间关闭了操作流程。这款被称为“TRITON”的恶意软件是一种攻击框架，用来与 Triconex 安全仪表系统(Safety Instrumented System)(SIS)控制器交互。目前还无法确定该攻击是由什么人发起的，但是这种行动与国家准备的攻击一致。Dragos Inc.的研究人员表示这起攻击的目标至少有一个是在中东地区。

事件背景

2017 年 11 月中旬，Dragos Inc.团队发现了针对 ICS 量身定做的恶意软件，目标至少一个在中东地区。 该团队将此恶意软件命名为 TRISIS（本文中的 TRITON），因为它将目标锁定施耐德电气的 Triconex 安全仪表系统（SIS），从而能够更换最终控制元素中的逻辑。

TRITON 具有高度针对性，可能不会对其他施耐德电气客户构成直接威胁，其他 SIS 产品也不会受到威胁。 重要的是，恶意软件不会在施耐德电气产品中利用固有的漏洞。 然而，这个特定事件中的这种能力，方法和流通现在可能被其他攻击者所复制，给工业资产所有者和运营商带来一类新的威胁。

攻击者获得了对 SIS 工程师站的远程访问权限，并部署了 TRITON 攻击框架来重新下装 SIS 控制器。事件中，一些 SIS 控制器进入失效的安全状态，自动关闭工业控制流程，并促使资产拥有者开始调查。调查发现，当冗余控制器之间的应用程序代码未通过验证检查时，SIS 控制器启动了安全关闭 - 导致 MP 诊断失败并在调查的过程中发现了 TRITON 的存在。

攻击者的长期目标是引起物理破坏后果的能力。基于这样一个事实，即攻击者最初在 DCS 上获得了可靠的立足点，并且已经具备了操纵流程或关闭工厂的能力，进一步危害了 SIS 系统。入侵后的 DCS 和 SIS 系统将使攻击者最大限度的对物理装置造成破坏。

一旦接触到 SIS 网络，攻击者使用预先构建的 TRITON 攻击框架

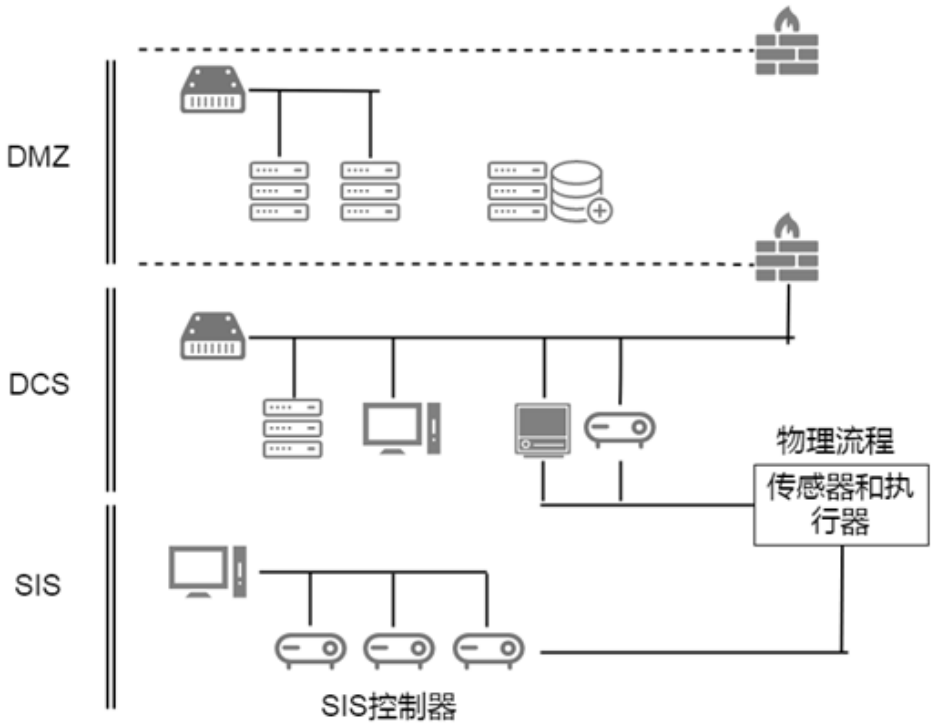
与使用 TriStation 协议的 SIS 控制器进行交互。攻击者可能通过发出暂停命令或向 SIS 控制器上传有缺陷的代码而导致进程关闭。相反，攻击者在一段时间内进行了多次尝试，为此目标环境中的 SIS 控制器提供功能控制逻辑。虽然由于 SIS 系统对攻击脚本的限制而导致攻击尝试失败，但攻击者仍然在不停的测试。这表明攻击者意图在造成关闭过程之外的特定结果。

TRISIS(TRITON)时间线

时间	影响
2017-11-17	Dragos 发现 TRITON 并开始跟进分析
2017-11 月底	Dragos 确认 TRITON 的恶意行为并确认至少有一个目标受影响 Dragos 联合 DOS 和 DHS 确认没有敏感信息泄漏并且决定暂时不公开 FireEye 知悉 Dragos 有该恶意样本，并和相关组织进行分析，再次确保敏感信息未被泄漏
2017-12-6	初始的安全通告发给 Dragos ICS WorldView 客户
2017-12-8	深度技术报告完成并且发给 Dragos ICS WorldView 客户
2017-12-10	Dragos 准备对外通告
2017-12-12	FireEye 发布相关技术报告，Dragos 也发布了相关报告
2017-12-12	绿盟科技安全团队关注并跟踪此事件，随时获取公布的样本
2017-12-13	绿盟科技安全团队完成主样本 trilog.exe 的分析
2017-12-25	绿盟科技安全团队获取到完整样本，开始技术分析
2017-12-29	绿盟科技发布技术分析报告与防护方案

系统架构

下图为 SIS 的结构简图：



现代工业过程控制和自动化系统依靠各种先进的控制系统和安全功能。这些系统和功能通常被称为工业控制系统（ICS）或操作技术（OT）。

分布式控制系统（DCS）为操作人员提供远程监视和控制工业过程的能力。它是由计算机，软件应用程序和控制器组成的计算机控制系统。工程工作站是用于控制系统应用和其他控制系统设备的配置，维护和诊断的计算机。

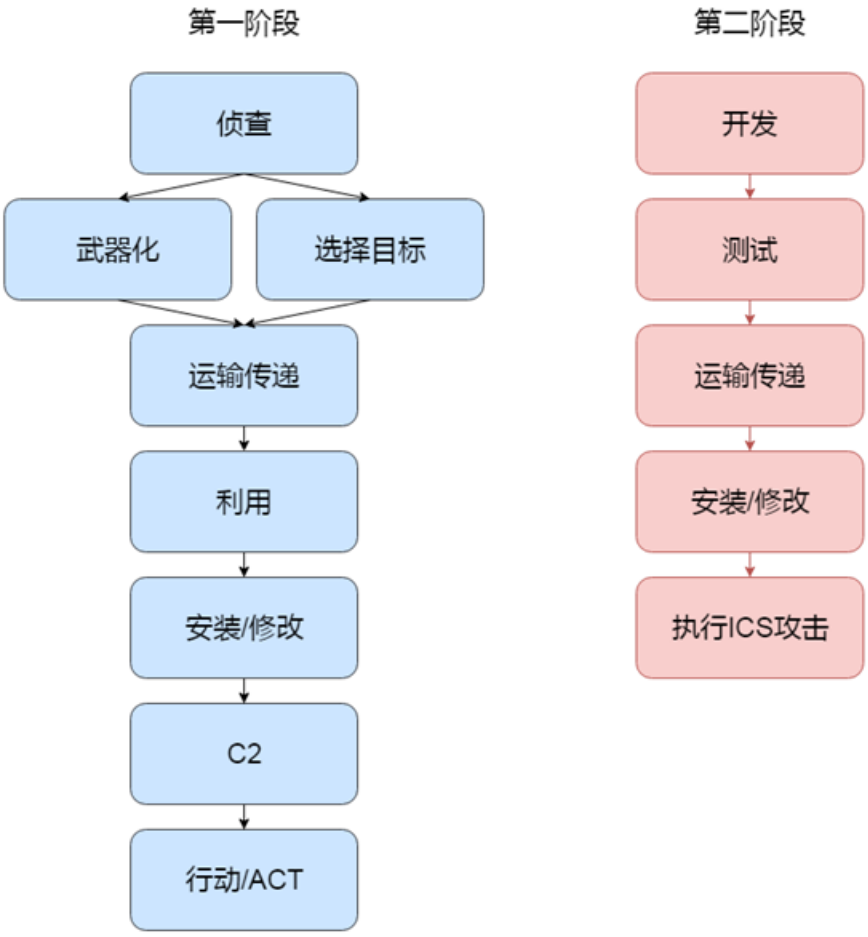
SIS 是一个独立的控制系统，可独立监控受控过程的状态。如果过程中有参数超过了定义的危险状态，则 SIS 会尝试将过程恢复到安全状态或自动执行流程的安全关闭。如果 SIS 和 DCS 控制失败，最后的防线就是工业设施的设计，其中包括设备（如爆破片）的机械保护，物理报警，应急程序和其他缓解危险情况的机制。

资产所有者采用不同的方法将工厂的 DCS 与 SIS 连接起来。传统

的方法依赖于通信基础设施和控制策略的隔离原则。至少在过去的十年中，由于各种原因（包括较低的成本，易用性以及通过 DCS 与 SIS 之间的信息交换所获得的益处)而出现了集成 DCS 和 SIS 设计的趋势。我们相信 TRITON 能够敏锐地展示与集成设计相关的风险，从而实现 DCS 与 SIS 网络主机之间的双向通信。

工业控制系统攻击链模型

ICS 攻击链模型分 2 个阶段，如下图所示：



攻击场景

TRITON 属于第二阶段的攻击。以下的 SIS 威胁模型着重描述了

攻击者在成功攻击 SIS 后，可以选择的后续操作。

1. 使用 SIS 关闭进程：

攻击者重新下装 SIS 逻辑使其跳闸并关闭一个处于安全状态的进程，换句话说就是触发一个误报。

结果：

由于停机时间造成财物损失并且停工后再启动的复杂流程。

2. 重新下装 SIS 来允许不安全的状态：

攻击者可以重新下装 SIS 逻辑来允许不安全的条件持续下去。

结果：

由于 SIS 逻辑被修改，会增加造成物理损坏（例如对设备，产品，环境和人身安全的影响）的风险。

3. 重新下装 SIS 以允许不安全的状态，同时使用 DCS 来创建不安全的状态或者危害：

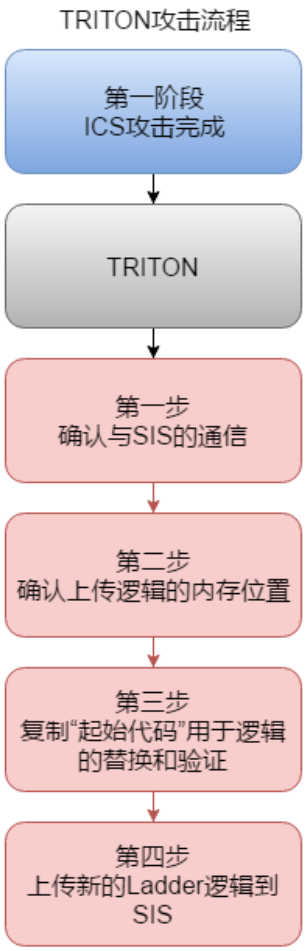
攻击者可以利用 DCS 把进程切换到不安全的状态，影响 SIS 的正常功能。

结果：

影响人身安全，环境或设备损坏，其程度取决于过程的物理限制和工艺流程。

攻击流程

TRITON 攻击流程简图如下：



- ICS Cyber Kill Chain 第一阶段的完成：

识别并访问能够与目标 SIS 通信的系统。

- 第二阶段发展：

识别目标 SIS 类型并用替换逻辑和加载程序开发 TRITON。

- 第二阶段测试：

确保 TRITON 按照预期工作。

- 第二阶段交付：

将 TRITON 转移到 SIS 中，该 SIS 包含用于提供新逻辑的新逻辑和支持二进制文件的“加载器”模块。

- 第 2 阶段安装/修改：

在运行 TRITON 可执行文件时，伪装成 Triconex 软件来分析 SIS 日志，恶意软件利用嵌入的二进制代码更改控制器的运行逻辑，并上传“初始化代码”。

- 第二阶段执行 ICS 攻击：

TRITON 验证上一步成功后，将新的梯形图逻辑上传到 SIS。

机器分析-绿盟威胁分析中心

上传时间	文件名称	文件 MD5	威胁
2017-12-25 15:11:39	trilog.exe	6c39c3f4a08d3d78f2eb973a94bd7718	中
2017-12-25 15:10:12	TsLow.pyc	f6b3a73c8c87506acda430671360ce15	低
2017-12-25 15:10:05	TS_cnames.pyc	e98f4f3505f05bf90e17554fbc97bba9	低
2017-12-25 15:09:57	TsBase.pyc	288166952f934146be172f6353e9a1f5	低
2017-12-25 15:09:47	imain.bin	437f135ba179959a580412e564d3107f	高
2017-12-25 15:09:39	TsHi.pyc	27c69aa39024d21ea109cc9c9d944a04	低
2017-12-25 15:09:30	sh.pyc	8b675db417cc8b23f4c43f3de5c83438	低
2017-12-25 15:09:05	library.zip	0face841f7b2953e7c29c064d6886523	中
2017-12-25 15:08:58	inject.bin	0544d425c7555dc4e9d76b571f31f500	高

高阶分析

TRITON 包括两部分：基于 Windows 平台的 Python 脚本程序 Trilog.exe，以及两个恶意代码 inject.bin、imain.bin。Trilog.exe 是 Triconex 应用软件中用于记录日志的程序。TRITON 病毒将该程序感染，用来与 Triconex 控制器通信。检测控制器状态，并将两个恶意代码 inject.bin 以及 imain.bin 注入到 Triconex 控制器中。

Trilog.exe

此程序为 py 打包成的 exe，经过如下命令可以反编译出最原始的执行脚本 script_test.py。

```
root@kali:~/triton# unpy2exe trilog.exe
root@kali:~/triton# ls
imain.bin  script_test.py.pyc  trilog.exe
root@kali:~/triton# uncompyle6 script_test.py.pyc > script_test.py
root@kali:~/triton# ls
imain.bin  script_test.py  script_test.py.pyc  trilog.exe
root@kali:~/triton#
```

Trilog.exe 通过 TriStation 网络协议（“TS”，UDP，端口 1502）与 Triconex 控制器进行通信，基于 5 个 python 模块（TsHi、TsBase、TsLow、TS_cnames、sh）完成，该协议没有任何认证或者加密算法。

- TsHi 用来实现对控制器攻击的整个过程。
- TsBase 包含了 TsHi 中所调用的程序，将攻击者所要实现的意图转换为 TS 通信协议函数代码。

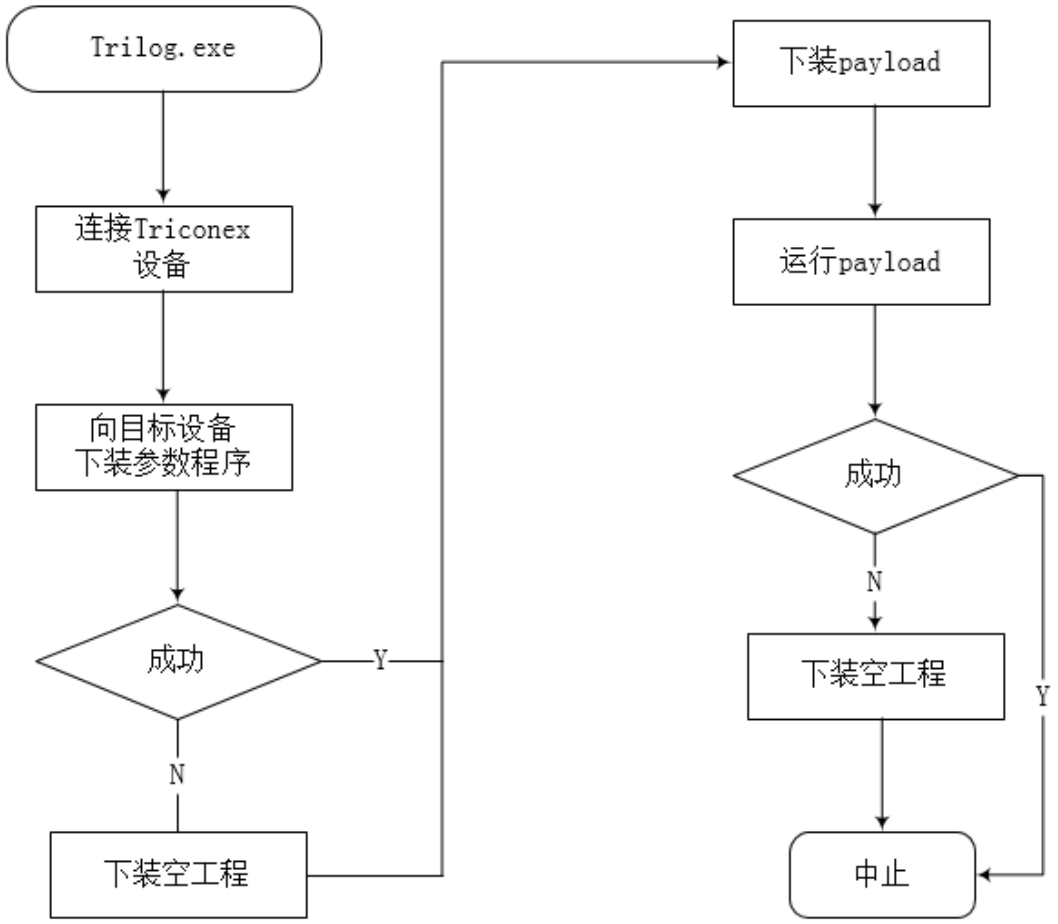
该协议的功能码如下所示：

功能码	含义
1	开始增量下装
10	所有下装结束
11	增量下装结束
12	取消下装
19	获取控制器状态
20	运行程序
21	停止程序
29	执行漏洞利用
54	获取模块版本
55	分配程序
56	分配函数
59	开始所有下装
65	上载程序
66	上载函数

- TsLow 包含了 TS 通信协议的底层函数。另外还包括了检测与控制器连接状态的函数。

Trilog 程序的攻击过程如下图所示，首先通过探测到的控制器 IP 来连接控制器设备。一旦连接成功，通过 PresetStatusField 函

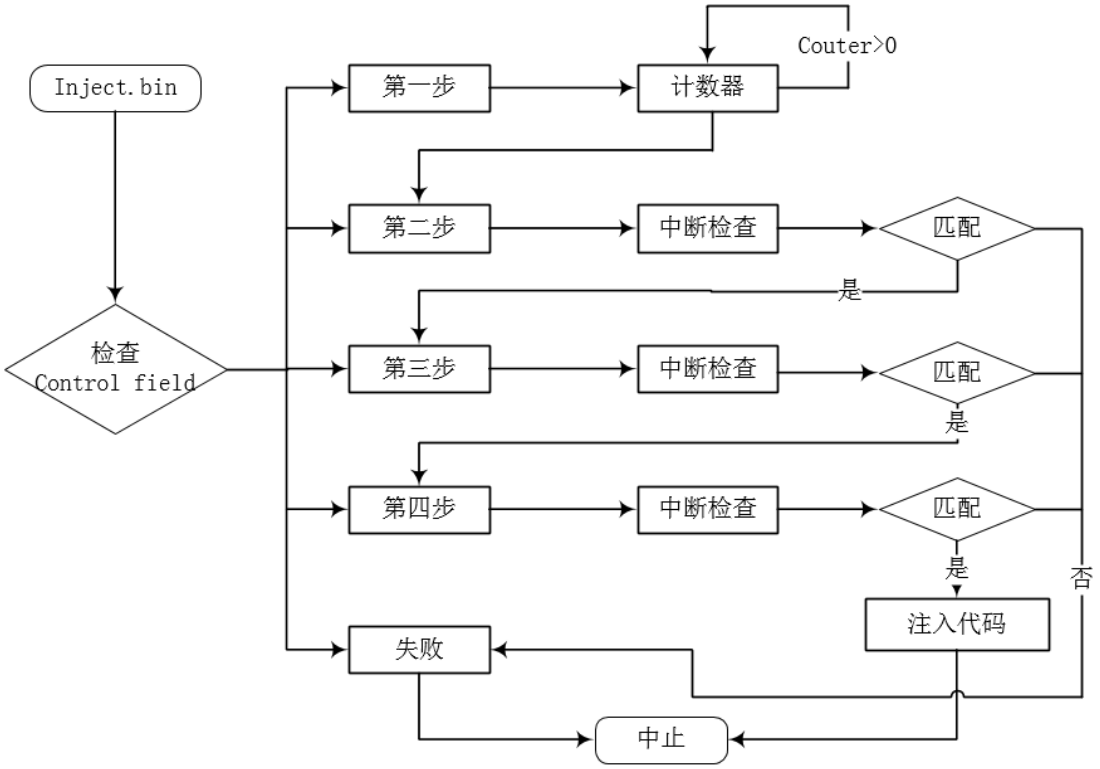
数将带有标识符号的参数程序下装到控制器中。下装之后会对标识符号进行检测，如检测到标识符号，Trilog 继续向控制器通过 SafeAppendProgramMod 函数下装 payload。如检测不到标识符，trilog 向控制器下装一个空工程以覆盖之前下装的参数程序，之后再继续向控制器下装 payload。payload 下装完成之后，执行该 payload。如果运行成功，则攻击完成，如果运行失败，Trilog 向控制器下装一个空工程用以覆盖之前下装的 payload。这一机制可以确保其攻击代码不被发现。



Inject.bin

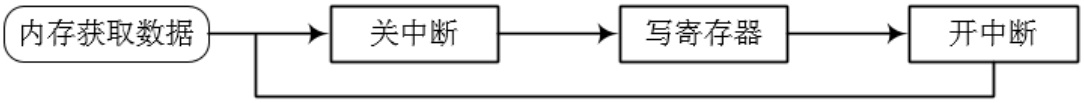
Inject.bin 为基于 PowerPC 架构的文件，其功能为向运行的固件注入 payload。该程序使用 Trilog.exe 向控制器下装的第一个参数程序中的一个特定值，这个特定值存在于控制器的一个特殊地址中。该地址可以通过使用 TS 通信协议中的“get control program staus”被获取。

Inject.bin 的实现过程如下图所示。在每一个循环周期中，Inject.bin 都会根据 Control field 中的值来进行相应步骤。第一步首先等待计数器的到来；第二步以及第三步会进行中断检查；第四步完成相应的系统中断检查后，会向内存中注入 payload。



Imain.bin

Imain.bin 为执行恶意功能的 payload 文件，也是基于 PowerPC 架构的文件。其实现过程如下图所示，首先在内存中获取数据。获取到数据后进行关中断操作，关中断完成后写寄存器，实现恶意功能，之后开中断，并不断循环该过程。



内存获取数据：

```
ROM:0000017C                 # Memory Data register
ROM:0000017C
ROM:0000017C                 Call_Function_0x6B9CC_17C:              # CODE XREF: ROM:SUIMCR↑p
ROM:0000017C 7F E3 FB 78                     mr      r3, r31
ROM:00000180 3C 00 00 06                     lis     r0, 6
ROM:00000184 60 00 B9 CC                     ori     r0, r0, 0xB9CC # 0x6B9CC
ROM:00000188 7C 09 03 A6                     mtctr   r0              # set the couter register value 0x6B9CC
ROM:0000018C 4E 80 04 20                     bctr
ROM:0000018C                 # End of function Call_Function_0x6B9CC_17C
ROM:0000018C
ROM:00000190
ROM:00000190                 # =============== S U B R O U T I N E =======================================
ROM:00000190
ROM:00000190
ROM:00000190                 Call_Function_0x68F0C_190:              # CODE XREF: ROM:00000054↑p
ROM:00000190 7F E3 FB 78                     mr      r3, r31
ROM:00000194 3C 00 00 06                     lis     r0, 6
ROM:00000198 60 00 8F 0C                     ori     r0, r0, 0x8F0C # 0x68F0C
ROM:0000019C 7C 09 03 A6                     mtctr   r0              # set the couter register value 0x68F0C
ROM:000001A0 4E 80 04 20                     bctr
ROM:000001A0                 # End of function Call_Function_0x68F0C_190
ROM:000001A0
ROM:000001A4
ROM:000001A4                 # =============== S U B R O U T I N E =======================================
ROM:000001A4
ROM:000001A4
ROM:000001A4                 Get_value_199400_1A4:                   # CODE XREF: ROM:00000010↑p
ROM:000001A4 3C 60 00 19                     lis     r3, 0x19        # r3= 0x190000
ROM:000001A8 60 63 94 00                     ori     r3, r3, 0x9400 # 0x199400 # r3 = r3 or 0x9400 = 0x199400
ROM:000001AC 88 63 00 00                     lbz     r3, 0(r3)       # r3 = [0x199400]
ROM:000001B0 4E 80 00 20                     blr                     # return the parent function
ROM:000001B0                 # End of function Get_value_199400_1A4
```

关中断：

```
ROM:000000E0 40 80 00 0C                     bge     loc_EC          # PCMCIA Interface General Control Register A
ROM:000000E4 3B 40 00 01                     li      r26, 1          # PCMCIA Interface General Control Register B
ROM:000000E8 48 00 00 2D                     bl      OR2             # 114
ROM:000000E8                                                         # Disable External Interrupt
ROM:000000EC
ROM:000000EC                 loc_EC:                                 # CODE XREF: ROM:000000E0↑j
ROM:000000EC 38 9D 00 14                     addi    r4, r29, 0x14 # src addr
```

```
ROM:00000114                loc_114:                                # CODE XREF: ROM:000000E8↑p
ROM:00000114 7C 71 13 A6                    mtspr     eid, r3 # External interrupt disable
ROM:00000118 38 80 FF C0                    li        r4, -0x40
ROM:00000118                # ---------------------------------------------------------------------------
ROM:0000011C 7C                             .byte 0x7C              # Option Register 3
ROM:0000011D 63                             .byte 0x63 # c
ROM:0000011E 00                             .byte    0
ROM:0000011F A6                             .byte 0xA6
ROM:00000120                # ---------------------------------------------------------------------------
ROM:00000120 7C 83 18 38                    and       r3, r4, r3
ROM:00000124 7C 63 01 24                    mtmsr     r3, 3         # context_trans_120
ROM:00000128 4E 80 00 20                    blr
ROM:0000012C                # ---------------------------------------------------------------------------
```

开中断：

```
ROM:0000013C                # ---------------------------------------------------------------------------
ROM:0000013C 60 63 00 30                    ori       r3, r3, 0x30  # Option Register 7
ROM:00000140 7C 63 01 24                    mtmsr     r3, 3
ROM:00000144 7C 00 04 AC                    sync
ROM:00000148 7C 70 13 A6                    mtspr     eie, r3 # External interrupt enable
ROM:0000014C 4E 80 00 20                    blr
```

写指令缓存状态控制寄存器：

```
ROM:0000012C                loc_12C:                                # CODE XREF: ROM:00000104↑p
ROM:0000012C 3C 60 0C 00                    lis       r3, 0xC00
ROM:00000130 7C 70 8B A6                    mtspr     ic_csr, r3 # I-cache Control and Status Register
ROM:00000134 4C 00 01 2C                    isync
ROM:00000134                # ---------------------------------------------------------------------------
```

指令缓存状态寄存器如下表所示，imain.bin 中会向该寄存器写值 0xC00，即第 10、第 11 位为 1，这两位分别表示总线错误故障以及锁定错误故障，如下图所示，造成控制器停止运行。

位	含义	值
0	IEN 指令缓存状态	0
1-3	保留	0
4-6	CMD 指令缓存命令	0
7-9	保留	0
10	CCER1 总线错误	1
11	CCER2 锁定错误	1
12-31	保留	0

10	CCER1	Instruction cache error type 1—bus error during an IC_CST load and load cache block command 0 No error detected 1 Error detected **Note:** This is a read-only, sticky bit, set only by the MPC860 when an error is detected. Reading this bit clears it.
11	CCER2	Instruction cache error type 2—no unlocked way available for an IC_CST load & lock cache block command 0 No error detected 1 Error detected **Note:** This is a read-only, sticky bit, set only by the MPC860 when an error is detected. Reading this bit clears it.

防护建议

如果资产拥有者想防护上述的攻击，建议考虑以下控制措施：

- 在技术上可行的情况下，将安全系统网络与过程控制和信息系统网络分开。能够对 SIS 控制器进行编程的工程师站应该与其他 DCS 或信息系统进行网络隔离。
- 利用硬件进行访问控制。一般采用由物理密钥控制的交换机的形式。在 Triconex 控制器上，密钥只应该保留在预定的编程事件中。
- 定期更改密码。
- 对于依赖于 SIS 提供的数据的任何应用程序，使用单向隔离设备。
- 在通过 TCP/IP 访问 SIS 系统的服务器或工作站端点上实施严格的访问控制和应用程序白名单。
- 监控 ICS 网络流量，以预测异常通信流量和其他活动。
- 在非必要的情况下，禁止外部设备连入本地 1502 端口。

绿盟科技检测与防护方案

绿盟科技检测服务

● 绿盟科技工程师前往客户现场检测。

绿盟科技木马专杀解决方案

● 短期服务：绿盟科技工程师现场木马后门清理服务（人工服务+IPS+TAC）。确保第一时间消除网络内相关风险点，控制事件影响范围，提供事件分析报告。

● 中期服务：提供 3-6 个月的风险监控与巡检服务（IPS+TAC+人工服务）。根除风险，确保事件不复发。

● 长期服务：基金行业业务风险解决方案（威胁情报+攻击溯源+专业安全服务）。

总结

TRITON 以多种方式表明了对于 ICS 网络防御的“决定性”影响。虽然以前在理论中的攻击情况下，针对 SIS 设备只是代表了 ICS 计算机网络攻击内的一个演变，然而潜在影响包括设备损坏，系统停机时间和潜在的生命损失。鉴于这些影响，重要的是要确保行业如何响应和沟通这种攻击。应该合理适当的利用 TRITON 来作为推进安全的一种契机。

附录

文件列表

FileName	HASH
trilog.exe	MD5: 6c39c3f4a08d3d78f2eb973a94bd7718 SHA-256:e8542c07b2af63ee7e72ce5d97d91036c5da56e2b091aa2afe737b224305d230
imain.bin	MD5: 437f135ba179959a580412e564d3107f SHA-256:08c34c6ac9186b61d9f29a77ef5e618067e0bc9fe85cab1ad25dc6049c376949
inject.bin	MD5: 0544d425c7555dc4e9d76b571f31f500 SHA-256:5fc4b0076eac7aa7815302b0c3158076e3569086c4c6aa2f71cd258238440d14
library.zip	MD5: 0face841f7b2953e7c29c064d6886523 SHA-256:bef59b9a3e00a14956e0cd4a1f3e7524448cbe5d3cc1295d95a15b83a3579c59
TS_cnames.pyc	MD5: e98f4f3505f05bf90e17554fbc97bba9 SHA-256:2c1d3d0a9c6f76726994b88589219cb8d9c39dd9924bc8d2d02bf41d955fe326

TsBase.pyc	MD5: 288166952f934146be172f6353e9a1f5 SHA-256:1a2ab4df156ccd685f795baee7df49f8e701f271d3e5676b507112e30ce03c42
TsHi.pyc	MD5: 27c69aa39024d21ea109cc9c9d944a04 SHA-256:758598370c3b84c6fbb452e3d7119f700f970ed566171e879d3cb41102154272
TsLow.pyc	MD5: f6b3a73c8c87506acda430671360ce15 SHA-256:5c776a33568f4c16fee7140c249c0d2b1e0798a96c7a01bfd2d5684e58c9bb32
sh.pyc	MD5: 8b675db417cc8b23f4c43f3de5c83438 SHA-256:c96ed56bf7ee85a4398cc43a98b4db86d3da311c619f17c8540ae424ca6546e1

YARA 规则

```
rule TRITON_ICS_FRAMEWORK
{
      strings:
          $python_compiled = ".pyc" nocase ascii wide
          $python_module_01 = "__module__" nocase ascii wide
          $python_module_02 = "<module>" nocase ascii wide
          $python_script_01 = "import Ts" nocase ascii wide
          $python_script_02 = "def ts_" nocase ascii wide

          $py_cnames_01 = "TS_cnames.py" nocase ascii wide
          $py_cnames_02 = "TRICON" nocase ascii wide
          $py_cnames_03 = "TriStation " nocase ascii wide
          $py_cnames_04 = " chassis " nocase ascii wide

          $py_tslibs_01 = "GetCpStatus" nocase ascii wide
          $py_tslibs_02 = "ts_" ascii wide
          $py_tslibs_03 = " sequence" nocase ascii wide
          $py_tslibs_04 = /import Ts(Hi|Low|Base)[^:alpha:]/ nocase ascii wide
          $py_tslibs_05 = /module\s?version/ nocase ascii wide
          $py_tslibs_06 = "bad " nocase ascii wide
          $py_tslibs_07 = "prog_cnt" nocase ascii wide

          $py_tsbase_01 = "TsBase.py" nocase ascii wide
          $py_tsbase_02 = ".TsBase(" nocase ascii wide

          $py_tshi_01 = "TsHi.py" nocase ascii wide
          $py_tshi_02 = "keystate" nocase ascii wide
          $py_tshi_03 = "GetProjectInfo" nocase ascii wide
```

```
        $py_tshi_04 = "GetProgramTable" nocase ascii wide
        $py_tshi_05 = "SafeAppendProgramMod" nocase ascii wide
        $py_tshi_06 = ".TsHi(" ascii nocase wide

        $py_tslow_01 = "TsLow.py" nocase ascii wide
        $py_tslow_02 = "print_last_error" ascii nocase wide
        $py_tslow_03 = ".TsLow(" ascii nocase wide
        $py_tslow_04 = "tcm_" ascii wide
        $py_tslow_05 = " TCM found" nocase ascii wide

        $py_crc_01 = "crc.pyc" nocase ascii wide
        $py_crc_02 = "CRC16_MODBUS" ascii wide
        $py_crc_03 = "Kotov Alaxander" nocase ascii wide
        $py_crc_04 = "CRC_CCITT_XMODEM" ascii wide
        $py_crc_05 = "crc16ret" ascii wide
        $py_crc_06 = "CRC16_CCITT_x1D0F" ascii wide
        $py_crc_07 = /CRC16_CCITT[^_]/ ascii wide

        $py_sh_01 = "sh.pyc" nocase ascii wide

        $py_keyword_01 = " FAILURE" ascii wide
        $py_keyword_02 = "symbol table" nocase ascii wide

        $py_TRIDENT_01 = "inject.bin" ascii nocase wide
        $py_TRIDENT_02 = "imain.bin" ascii nocase wide

    condition:
        2 of ($python_*) and 7 of ($py_*) and filesize < 3MB
}
```

或者：

```
private global rule hatman_filesize : hatman {
    condition:
        filesize < 100KB
}

private rule hatman_setstatus : hatman {
    strings:
        $preset     = { 80 00 40 3c  00 00 62 80  40 00 80 3c  40 20 03 7c
                        ?? ?? 82 40  04 00 62 80  60 00 80 3c  40 20 03 7c
                        ?? ?? 82 40  ?? ?? 42 38                          }
    condition:
```

```
        $preset
}

private rule hatman_memcpy : hatman {
    strings:
        $memcpy_be  = { 7c a9 03 a6  38 84 ff ff  38 63 ff ff  8c a4 00 01
                        9c a3 00 01  42 00 ff f8  4e 80 00 20              }
        $memcpy_le  = { a6 03 a9 7c  ff ff 84 38  ff ff 63 38  01 00 a4 8c
                        01 00 a3 9c  f8 ff 00 42  20 00 80 4e              }
    condition:
        $memcpy_be or $memcpy_le
}

private rule hatman_dividers : hatman {
    strings:
        $div1       = { 9a 78 56 00 }
        $div2       = { 34 12 00 00 }
    condition:
        $div1 and $div2
}

private rule hatman_nullsub : hatman {
    strings:
        $nullsub     = { ff ff 60 38  02 00 00 44  20 00 80 4e }
    condition:
        $nullsub
}

private rule hatman_origaddr : hatman {
    strings:
        $oaddr_be   = { 3c 60 00 03  60 63 96 f4  4e 80 00 20 }
        $oaddr_le   = { 03 00 60 3c  f4 96 63 60  20 00 80 4e }
    condition:
        $oaddr_be or $oaddr_le
}

private rule hatman_origcode : hatman {
    strings:
        $ocode_be   = { 3c 00 00 03  60 00 a0 b0  7c 09 03 a6  4e 80 04 20 }
        $ocode_le   = { 03 00 00 3c  b0 a0 00 60  a6 03 09 7c  20 04 80 4e }
    condition:
        $ocode_be or $ocode_le
}
```

```
private rule hatman_mftmsr : hatman {
    strings:
        $mfmsr_be   = { 7c 63 00 a6 }
        $mfmsr_le   = { a6 00 63 7c }
        $mtmsr_be   = { 7c 63 01 24 }
        $mtmsr_le   = { 24 01 63 7c }
    condition:
        ($mfmsr_be and $mtmsr_be) or ($mfmsr_le and $mtmsr_le)
}

private rule hatman_loadoff : hatman {
    strings:
        $loadoff_be = { 80 60 00 04  48 00 ?? ??  70 60 ff ff  28 00 00 00
                        40 82 ?? ??  28 03 00 00  41 82 ?? ??              }
        $loadoff_le = { 04 00 60 80  ?? ?? 00 48  ff ff 60 70  00 00 00 28
                        ?? ?? 82 40  00 00 03 28  ?? ?? 82 41              }
    condition:
        $loadoff_be or $loadoff_le
}

private rule hatman_injector_int : hatman {
    condition:
        hatman_memcpy and hatman_origaddr and hatman_loadoff
}

private rule hatman_payload_int : hatman {
    condition:
        hatman_memcpy and hatman_origcode and hatman_mftmsr
}



rule hatman_compiled_python : hatman {
    condition:
        hatman_nullsub and hatman_setstatus and hatman_dividers
}

rule hatman_injector : hatman {
    condition:
        hatman_injector_int and not hatman_payload_int
}

rule hatman_payload : hatman {
    condition:
        hatman_payload_int and not hatman_injector_int
```

```
}

rule hatman_combined : hatman {
    condition:
        hatman_injector_int and hatman_payload_int and hatman_dividers
}

rule hatman : hatman {
    condition:
        hatman_compiled_python or hatman_injector or hatman_payload
            or hatman_combined
}
```

声 明

本安全公告仅用来描述可能存在的安全问题，绿盟科技不为此安全公告提供任何保证或承诺。由于传播、利用此安全公告所提供的信息而造成的任何直接或者间接的后果及损失，均由使用者本人负责，绿盟科技以及安全公告作者不为此承担任何责任。绿盟科技拥有对此安全公告的修改和解释权。如欲转载或传播此安全公告，必须保证此安全公告的完整性，包括版权声明等全部内容。未经绿盟科技允许，不得任意修改或者增减此安全公告内容，不得以任何方式将其用于商业目的。

关于绿盟科技

北京神州绿盟信息安全科技股份有限公司（简称绿盟科技）成立于 2000 年 4 月，总部位于北京。在国内外设有 30 多个分支机构，为政府、运营商、金融、能源、互联网以及教育、医疗等行业用户，提供具有核心竞争力的安全产品及解决方案，帮助客户实现业务的安全顺畅运行。

基于多年的安全攻防研究，绿盟科技在网络及终端安全、互联网基础安全、合规及安全管理等领域，为客户提供入侵检测/防护、抗拒绝服务攻击、远程安全评估以及 Web 安全防护等产品以及专业安全服务。

北京神州绿盟信息安全科技股份有限公司于 2014 年 1 月 29 日起在深圳证券交易所创业板上市交易，股票简称：绿盟科技，股票代码：300369。

绿盟科技官方微博二维码

绿盟科技官方微信二维码