ICCP Exposed: Assessing the Attack Surface of the "Utility Stack"

Matthew Franz
Digital Bond, Inc.
1580 Sawgrass Corporate Parkway, Suite 130
Sunrise, FL 33323
mdfranz@gmail.com

Abstract: ICCP server applications are much more than the implementation of the ICCP protocol. In fact, ICCP servers implement a set of protocols we refer to as the "Utility Stack". These protocols are not exclusive to ICCP, but they are not commonly tested by the security community. In this paper, we review the Utility Stack, discuss the details of two published vulnerabilities in implementations of the Utility Stack, and identify areas where additional vulnerabilities are likely to be found. The paper finishes with a discussion of a Utility Stack vulnerability testing tool that mimics the approach attackers have taken to identify exploits in IT protocols.

Keywords: COTP, Denial of Service, ICCP, SCADA, TPKT, Vulnerability Testing

1 Introduction

Unlike simple application layer SCADA protocols such as Modbus, the Inter Control Center Protocol (ICCP, also known as TASE.2) is not a single protocol, but a complex, multi-layered suite that provides a large attack surface for adversaries to exploit and ample opportunities for developers to make coding errors. Just as many application administrators may be unfamiliar with nuances of securing TCP/IP, many SCADA/EMS application administrators are unfamiliar with the obscure OSI protocol layers required for successful communication between ICCP servers.

Although ICCP is not typically deployed on public facing networks such as the Internet, ICCP is used to exchange data across organizational boundaries -- typically through firewalls, most which have limited to no ability to inspect traffic or make policy decisions above the transport layer. Based on publicly disclosed vulnerabilities [1, 2, 3], ICCP servers have already suffered from a number of security vulnerabilities based on failure to handle invalid messages generated by Open Source scanners or crude fuzzers. Given the relatively small number of ICCP implementations and dominant market share of the largest ICCP stack vendor there is a high probably of a "software monoculture" often ascribed to Microsoft where the discovery (and exploitation) of a new, previously unknown vulnerability could have a significant impact on the management and monitoring of the electric power grid.

Given these concerns, this paper seeks to provide a detailed technical treatment of ICCP threats and vulnerabilities that will provide a concise review of the "Utility Stack" used by ICCP Servers; to define, implement, and demonstrate likely attacks against real ICCP implementations; and identify the network traffic patterns necessary to detect and prevent attacks described in the paper. We believe by bringing to light real world threats and vulnerabilities and security relevant details of the protocol stack, end users, security researchers, and vendors can make risk decisions to deploy appropriate security technology or process changes.

2 Approach

A common approach to analyzing protocol security is to review the specification and analyze the security measures. We did not take this approach to IEC-60870-6-503 [4] because there are minimal security controls in the standard. Without protection in place for confidentiality, integrity and availability it is clear that the protocol can be abused by an attacker with network access to an ICCP server.

Standards bodies developing SCADA protocols or Internet protocols have typically focused exclusively on security functionality[1] and if threats are considered, it is typically only at the highest levels of abstraction. There is however a larger issue. What if the ICCP server application and underlying stack can be compromised to not only compromise EMS data on the server, but also to gain complete control of the server? ICCP servers typically have widespread communication to ICCP servers in other utilities and to SCADA servers in the organization.

Imagine the nightmare scenario of a Utility Stack implementation vulnerability in an ICCP server that allows an attacker with network access to gain privileged access on an ICCP server. Once the attacker comprised the ICCP server he could launch attacks on unpatched SCADA servers, an all too common occurrence, as well as try to compromise other utilities' ICCP servers where required ICCP communication was allowed through the firewall. Given the reliance on a small number of ICCP stacks, an implementation vulnerability could have widespread implications. For example, the LiveData ICCP stack vulnerability [2] impacted at least twelve different vendors' ICCP servers.

Exploits of an ICCP implementation vulnerability could potentially be an automated worm where the "ICCP Payload" simply replaces the payload of another Microsoft platform vulnerability. The worm could first use the ICCP vulnerability to exploit an ICCP server; exploit other communicating ICCP servers with security associations; and then try to exploit any other reachable servers inside the security perimeter with a set of common worms such as Slammer and Zotob. Given the required ICCP communication in the electric grid and widespread patching issues in SCADA servers the results could indeed by a nightmare.

[1] After raising the issue of OSI protocol implementation flaws to UCA board members, we were invited to attend a UCA testing meeting. The consensus among committee members was the lower layer protocols were "secure" and any attempt to define test procedures for subordinate layers of the utility stack was out of scope for the committee. There was also a great deal of confusion between Windows OS vulnerabilities and vulnerabilities in Utility applications (such as ICCP servers) that were deployed on Windows.

We believe a more useful approach to the Utility Stack is to apply well-known attack patterns and examine the behavior of common implementations. To that end, this paper does not seek to be formal (or comprehensive) security analysis of relevant protocol specifications, nor a comprehensive assessment of all likely attack vectors against ICCP. In fact, much of our discussion will focus on lower protocol layers below TASE.2 – where there are the significant risks and where vulnerabilities have been disclosed by SCADA vendors and government coordination centers.

As we describe selected protocol layers and likely threats and countermeasures to each protocol layer, our approach will be to:

1. Focus "on the wire" starting at the bottom of the stack and working our way up, as opposed to the beginning with protocol specification or abstract data models.
2. Develop and demonstrate proof of concept tools for a set of likely attacks against ICCP servers, instead of defining high level threats to confidentiality, integrity, availability, etc.
3. Avoid generic statements whenever possible. The community requires detail rather than more FUD.
4. Use terminology for protocol layers based on Ethereal/Wireshark and publicly available protocol specifications such as RFC's.

Although this approach is not unusual for Internet protocols, this type of work has not yet been seen in any public documents on ICCP security, or for that matter, any analyses of SCADA protocol security to our knowledge. We believe this work will not only define a starting point for additional ICCP threat and vulnerability research, but it will allow solutions to be based on real threats and vulnerabilities in actual ICCP server implementation. Previous papers have tended to be more generic or focus on vendor solutions vs. actual vulnerabilities [5].

3 The ICCP Protocol Stack

In our experience, the greatest exposures to ICCP servers are not inherent in the TASE.2 application layer that handles the exchange of process control data between control centers, but in the six layers below ICCP/TASE.2, as defined in the Figure 1 below[2]:

[2] Our analysis of Utility Stack is based on ICCP over TCP/IP, based on traffic captured from LiveData's and Sisco's ICCP servers and the free version of Tamarack UCA client & server available from Netted Automation.

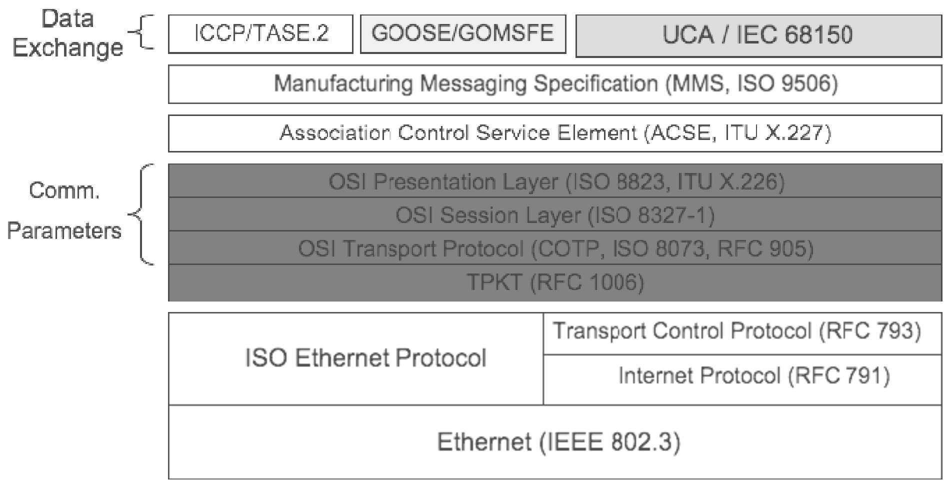

Data Exchange
ICCP/TASE.2
GOOSE/GOMSFE
UCA / IEC 68150
Manufacturing Messaging Specification (MMS, ISO 9506)
Association Control Service Element (ACSE, ITU X.227)
Comm. Parameters
OSI Presentation Layer (ISO 8823, ITU X.226)
OSI Session Layer (ISO 8327-1)
OSI Transport Protocol (COTP, ISO 8073, RFC 905)
TPKT (RFC 1006)
ISO Ethernet Protocol
Transport Control Protocol (RFC 793)
Internet Protocol (RFC 791)
Ethernet (IEEE 802.3)

Figure 1 – The "Utility Stack"

Indeed, all publicly disclosed vulnerabilities to date have been in lower layer protocol implementations--not in ICCP itself. However, given the problems with other protocols such as SNMP, SSL, and H.323, it is likely there are more latent implementation flaws at all layers of the stack. The use of ASN.1 encoding in ICCP/MMS is an area of particular concern.

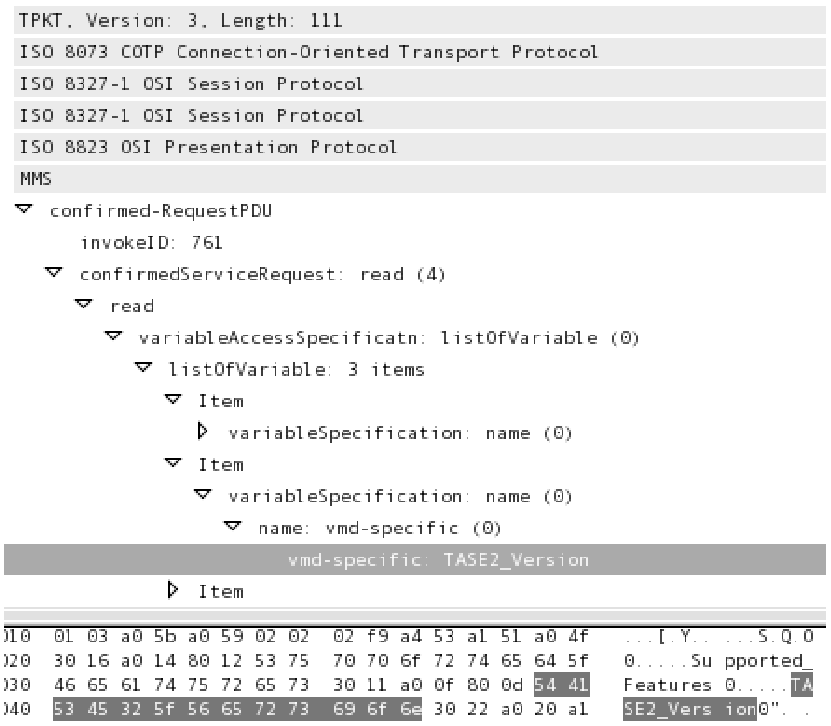
Before attackers can target upper layers they must successfully generate these lower-layer protocol data units (PDU) and negotiate sessions for TPKT, COTP, OSI Session and Presentation layer. Given the relative obscurity of these protocols and the lack of the free or Open Source implementations, this is a significant burden not only for the majority of attackers, but also for vulnerability researchers, network security vendors, and even end users. Trying to learn and understand the protocols from the available documentation proved to be difficult, and trial and error with actual implementations found this was due to many gaps in the documentation.

One reasonable conclusion is the profile of a successful ICCP attacker is likely to be a highly motivated and skilled individual. In addition to time and skill, the attacker will need to gain access to one or more ICCP servers to develop and debug these complex attacks. This is not difficult if an attacker has $5,000 to $20,000 in resources. So a likely attacker profile could be a state sponsored agent, a cyber terrorist, or a disgruntled insider.

Tools for conducting passive analysis of ICCP traffic are more readily available. For several years a modified (but older) version of Ethereal for Windows, known as Ethereal-MMS[3], has been available which allows debugging and analysis of ICCP traffic.

[3] Ethereal MMS 11.3 is available from the Netted Automation page at http://www.nettedautomation.com/solutions/demo/20040720/index.html

Recent versions of Wireshark[4] have significantly improved support as compared to older versions of standard version of Ethereal. A decode of the Utility Stack is displayed in Figure 2.



```
TPKT, Version: 3, Length: 111
ISO 8073 COTP Connection-Oriented Transport Protocol
ISO 8327-1 OSI Session Protocol
ISO 8327-1 OSI Session Protocol
ISO 8823 OSI Presentation Protocol
MMS
▽ confirmed-RequestPDU
    invokeID: 761
  ▽ confirmedServiceRequest: read (4)
    ▽ read
      ▽ variableAccessSpecificatn: listOfVariable (0)
        ▽ listOfVariable: 3 items
          ▽ Item
            ▷ variableSpecification: name (0)
          ▽ Item
            ▽ variableSpecification: name (0)
              ▽ name: vmd-specific (0)
                  vmd-specific: TASE2_Version
          ▷ Item

10  01 03 a0 5b a0 59 02 02  02 f9 a4 53 a1 51 a0 4f   ...[.Y.. ...S.Q.O
20  30 16 a0 14 80 12 53 75  70 70 6f 72 74 65 64 5f   0.....Su pported_
30  46 65 61 74 75 72 65 73  30 11 a0 0f 80 0d 54 41   Features 0.....TA
40  53 45 32 5f 56 65 72 73  69 6f 6e 30 22 a0 20 a1   SE2_Vers ion0". .
```

Figure 2 – Utility Stack Decode

It was necessary to develop large portions of the Utility Stack as part of our research. To that end we have developed a toolset for generating messages at different layers of the Utility Stack. This toolset is not yet complete, and access is restricted to vetted asset owners. The toolset is a set of Python scripts that generated valid protocol headers and are used to fuzz various layers of the Utility Stack.

In the sections below we will discuss some of the protocols we have tested for implementation vulnerabilities.

[4] Wireshark URL version 0.99.2 on OSX was used for the examples in this paper.

3.1 RFC 1006 (TPKT)

RFC 1006 [6] defines a simple approach for encapsulating ISO protocols over TCP/102 using a 32-bit packet header which includes the following fields:

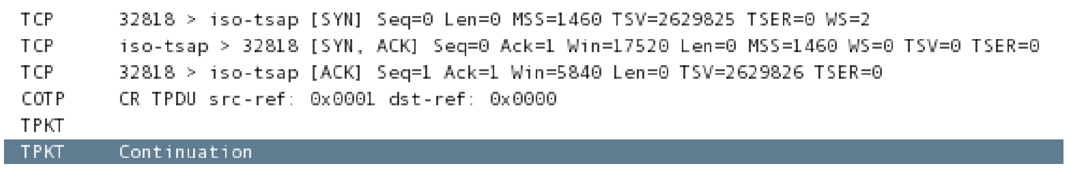
Field	Length	Comments
Version	8 bits	Set to 2 or 3
Reserved	8 bits	Set to 0
Packet Length	16 bits	size of the entire packet including the header with a maximum TPDU

Figure 3 – TPKT Fields

The data portion of the TPKT is referred to as the TPDU.

The standard briefly describes interactions with the next layer TP0/COTP which will be described in Section 3.2. It is important to note the TPKT format is not unique to ICCP and is used by other protocol suites including SS7 and H.323.[5]

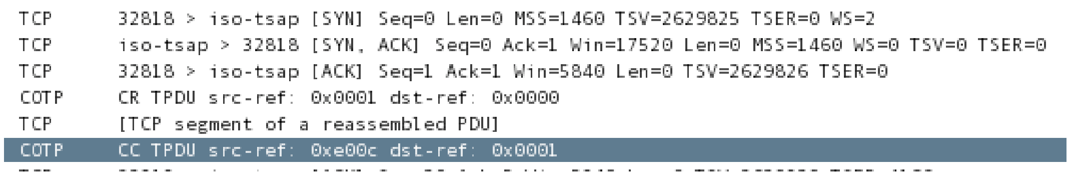
Although not defined in the standard, we observed fragmentation in the Sisco/Marzden stack where a TPKT would be split across multiple TCP segments even though it was well below the TCP maximum segment size.



```
TCP      32818 > iso-tsap [SYN] Seq=0 Len=0 MSS=1460 TSV=2629825 TSER=0 WS=2
TCP      iso-tsap > 32818 [SYN, ACK] Seq=0 Ack=1 Win=17520 Len=0 MSS=1460 WS=0 TSV=0 TSER=0
TCP      32818 > iso-tsap [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=2629826 TSER=0
COTP     CR TPDU src-ref: 0x0001 dst-ref: 0x0000
TPKT
TPKT     Continuation
```

Figure 4 – TPKT Fragmentation

When the "Reassemble TPKT messages spanning multiple TCP segments" option is checked the exchange is correctly displayed in Wireshark.



```
TCP      32818 > iso-tsap [SYN] Seq=0 Len=0 MSS=1460 TSV=2629825 TSER=0 WS=2
TCP      iso-tsap > 32818 [SYN, ACK] Seq=0 Ack=1 Win=17520 Len=0 MSS=1460 WS=0 TSV=0 TSER=0
TCP      32818 > iso-tsap [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=2629826 TSER=0
COTP     CR TPDU src-ref: 0x0001 dst-ref: 0x0000
TCP      [TCP segment of a reassembled PDU]
COTP     CC TPDU src-ref: 0xe00c dst-ref: 0x0001
```

Figure 5 – TPKT Reassembled

[5] Some commercial IDS engines (such as Cisco's) have support for TPKT validation

3.2 COTP

ISO/IEC 8073 [7] commonly known as the Connection Oriented Transport Protocol (COTP) is best thought of as an additional transport layer on top of TCP. COTP includes Type-Length-Value fields which have been notorious in other binary protocols for their failure to handle invalid.

Below are the messages from a COTP connection to an ICCP server with the Sisco Utility Stack.

Message from Client:

```
TPKT, Version: 3, Length: 19
    Version: 3
    Reserved: 0
    Length: 19
ISO 8073 COTP Connection-Oriented Transport Protocol
    Length: 14
    PDU Type: CR Connect Request (0x0e)
    Destination reference: 0x0000
    Source reference: 0x0001
    Class: 0
    Option: 0
    Parameter code:   0xc1 (src-tsap)
    Parameter length: 2
    Source TSAP: 0002
    Parameter code:   0xc2 (dst-tsap)
    Parameter length: 2
    Destination TSAP: 0002
```

Message from Server:

```
TPKT, Version: 3, Length: 14
    Version: 3
    Reserved: 0
    Length: 14
ISO 8073 COTP Connection-Oriented Transport Protocol
    Length: 9
    PDU Type: CC Connect Confirm (0x0d)
    Destination reference: 0x0001
    Source reference: 0xe00c
    Class: 0
    Option: 0
    Parameter code:   0xc0 (tpdu-size)
    Parameter length: 1
    TPDU size: 1024
```

3.2.1 Data Exchange

After the COTP client and server have successfully established a connection, data exchange begins. All COTP data is encapsulated with a 3 byte header of "02F080".

3.2.2 COTP Error Messages

On both the LiveData and Sisco ICCP implementations, we observed that the COTP Disconnect Request PDU returned several different responses depending on the type of data that was sent. This information is useful for both the attacker and defender.

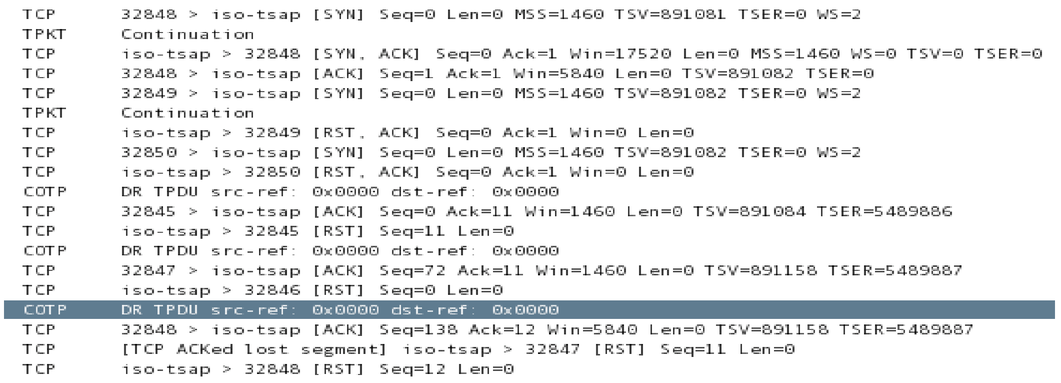
4 Exploiting ICCP

Although there are a wide variety of ways of classifying attacks against network protocols, we define three simple classes of attacks against ICCP servers:

1. *Attacks against process control data* – protocol security efforts within the standards community (IEC TC 57, Secure DNP, OPC-UA) are meant to address these threats to integrity or confidentiality of the data exchanged by control system elements by adding appropriate controls to the specification at different layers.
2. *Attacks against ICCP server (or EMS) processes* – these attacks attempt to undermine the integrity or availability of the ICCP server. These attacks may be then leveraged to run arbitrary code and gain complete control of the server.
3. *Attacks against ICCP server operating system* – the underlying operating system, and any other applications other than the Utility Stack, are part of the attack surface.

4.1 Reconnaissance

Typically, an adversary begins with identification of targets. Unlike protocols such as FTP, Telnet, and SSH, ICCP servers will not return data that allows easy identification based on a simple TCP connect( ). However, when invalid data is sent to the ICCP server we see that sometimes the connections are reset, but other times the COTP Disconnect Request (Protocol Error) message is returned. We can see this behavior in the excerpt of an Amap scan of a LiveData ICCP server.



```
TCP      32848 > iso-tsap [SYN] Seq=0 Len=0 MSS=1460 TSV=891081 TSER=0 WS=2
TPKT     Continuation
TCP      iso-tsap > 32848 [SYN, ACK] Seq=0 Ack=1 Win=17520 Len=0 MSS=1460 WS=0 TSV=0 TSER=0
TCP      32848 > iso-tsap [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=891082 TSER=0
TCP      32849 > iso-tsap [SYN] Seq=0 Len=0 MSS=1460 TSV=891082 TSER=0 WS=2
TPKT     Continuation
TCP      iso-tsap > 32849 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
TCP      32850 > iso-tsap [SYN] Seq=0 Len=0 MSS=1460 TSV=891082 TSER=0 WS=2
TCP      iso-tsap > 32850 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
COTP     DR TPDU src-ref: 0x0000 dst-ref: 0x0000
TCP      32845 > iso-tsap [ACK] Seq=0 Ack=11 Win=1460 Len=0 TSV=891084 TSER=5489886
TCP      iso-tsap > 32845 [RST] Seq=11 Len=0
COTP     DR TPDU src-ref: 0x0000 dst-ref: 0x0000
TCP      32847 > iso-tsap [ACK] Seq=72 Ack=11 Win=1460 Len=0 TSV=891158 TSER=5489887
TCP      iso-tsap > 32846 [RST] Seq=0 Len=0
COTP     DR TPDU src-ref: 0x0000 dst-ref: 0x0000
TCP      32848 > iso-tsap [ACK] Seq=138 Ack=12 Win=5840 Len=0 TSV=891158 TSER=5489887
TCP      [TCP ACKed lost segment] iso-tsap > 32847 [RST] Seq=11 Len=0
TCP      iso-tsap > 32848 [RST] Seq=12 Len=0
```

Figure 6 – Identifying ICCP Servers by COTP Disconnect Requests

We could have easily developed an Amap string to identify certain vendors ICCP implementations. However, the most consistent and safest way to identify ICCP server is by sending valid protocol messages and establishing valid connections.

4.2 Vulnerable ICCP Detection using Nessus

Certain versions of the LiveData, Tamarack and Sisco Utility Stacks have published vulnerabilities [2, 3, 4]. We predict the SCADA security community is going to have much in common with the IT security community in the upcoming years as the largest number of attackers simply identify and exploit unpatched SCADA devices and applications. The Nessus SCADA plugins developed by Digital Bond for Tenable Network Security include checks to identify ICCP servers and stacks with known vulnerabilities. Asset owners can use these Nessus plugins to identify ICCP servers that need to be patched, just as they do with Microsoft, Oracle, Linux, and other software.

4.3 Known Implementation Flaws

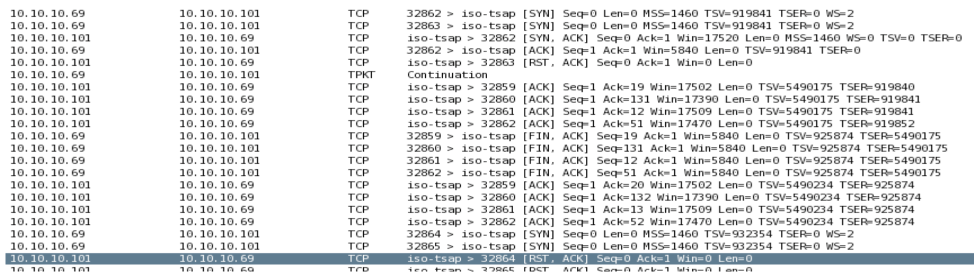
Although US-CERT has disclosed several vulnerabilities related to the Utility Stack, very few details were provided in advisories. In this subsection, we provide more detail on two very different types of vulnerabilities. At the time of this paper there was also a Tamarack malformed TPKT vulnerability (CVE-2006-1178) identified by Digital Bond and a Sisco malformed TPKT vulnerability (CVE-2005-4812) disclosed by Sisco.

4.3.1 Livedata ICCP (CVE-2006-0059)

The vulnerability reported to US-CERT was discovered initially with the Open Source scanner Amap and was then confirmed with a Digital Bond toolset we will describe in the next section.

This vulnerability would likely be discovered accidentally by any attacker or even by an IT staff member doing a basic scan. It is a cautionary note on the need to have a clear methodology for even basic scanning of control system networks.

In the trace below, the “attacker” is 10.10.10.69 and the ICCP Server is 10.10.10.101



```
10.10.10.69     10.10.10.101    TCP     32862 > iso-tsap [SYN] Seq=0 Len=0 MSS=1460 TSV=919841 TSER=0 WS=2
10.10.10.69     10.10.10.101    TCP     32863 > iso-tsap [SYN] Seq=0 Len=0 MSS=1460 TSV=919841 TSER=0 WS=2
10.10.10.101    10.10.10.69     TCP     iso-tsap > 32862 [SYN, ACK] Seq=0 Ack=1 Win=17520 Len=0 MSS=1460 WS=0 TSV=0 TSER=0
10.10.10.69     10.10.10.101    TCP     32862 > iso-tsap [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=919841 TSER=0
10.10.10.101    10.10.10.69     TCP     iso-tsap > 32863 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
10.10.10.69     10.10.10.101    TPKT    Continuation
10.10.10.101    10.10.10.69     TCP     iso-tsap > 32859 [ACK] Seq=1 Ack=19 Win=17502 Len=0 TSV=5490175 TSER=919840
10.10.10.101    10.10.10.69     TCP     iso-tsap > 32860 [ACK] Seq=1 Ack=131 Win=17390 Len=0 TSV=5490175 TSER=919841
10.10.10.101    10.10.10.69     TCP     iso-tsap > 32861 [ACK] Seq=1 Ack=12 Win=17509 Len=0 TSV=5490175 TSER=919841
10.10.10.101    10.10.10.69     TCP     iso-tsap > 32862 [ACK] Seq=1 Ack=51 Win=17470 Len=0 TSV=5490175 TSER=919852
10.10.10.69     10.10.10.101    TCP     32859 > iso-tsap [FIN, ACK] Seq=19 Ack=1 Win=5840 Len=0 TSV=925874 TSER=5490175
10.10.10.69     10.10.10.101    TCP     32860 > iso-tsap [FIN, ACK] Seq=131 Ack=1 Win=5840 Len=0 TSV=925874 TSER=5490175
10.10.10.69     10.10.10.101    TCP     32861 > iso-tsap [FIN, ACK] Seq=12 Ack=1 Win=5840 Len=0 TSV=925874 TSER=5490175
10.10.10.69     10.10.10.101    TCP     32862 > iso-tsap [FIN, ACK] Seq=51 Ack=1 Win=5840 Len=0 TSV=925874 TSER=5490175
10.10.10.101    10.10.10.69     TCP     iso-tsap > 32859 [ACK] Seq=1 Ack=20 Win=17502 Len=0 TSV=5490234 TSER=925874
10.10.10.101    10.10.10.69     TCP     iso-tsap > 32860 [ACK] Seq=1 Ack=132 Win=17390 Len=0 TSV=5490234 TSER=925874
10.10.10.101    10.10.10.69     TCP     iso-tsap > 32861 [ACK] Seq=1 Ack=13 Win=17509 Len=0 TSV=5490234 TSER=925874
10.10.10.101    10.10.10.69     TCP     iso-tsap > 32862 [ACK] Seq=1 Ack=52 Win=17470 Len=0 TSV=5490234 TSER=925874
10.10.10.69     10.10.10.101    TCP     32864 > iso-tsap [SYN] Seq=0 Len=0 MSS=1460 TSV=932354 TSER=0 WS=2
10.10.10.69     10.10.10.101    TCP     32865 > iso-tsap [SYN] Seq=0 Len=0 MSS=1460 TSV=932354 TSER=0 WS=2
10.10.10.101    10.10.10.69     TCP     iso-tsap > 32864 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
10.10.10.101    10.10.10.69     TCP     iso-tsap > 32865 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
```

Figure 7 – Packet Capture of LiveData Attack

The result on the LiveData ICCP server is shown in the error log in Figure 8.

```
16:34:17.386 (0) Servxnt: Socket(1744fc4): TP0 Connect Request TPDU
parameter ignored, 54
16:34:17.386 (0) Servxnt: Socket(1744fc4): TP0 Connect Request TPDU
parameter ignored, 91
16:34:17.386 (0) Servxnt: Socket(1744fc4): TP0 Connect Request TPDU
parameter ignored, 87
16:34:17.386 (0) Servxnt: Socket(1744fc4): TP0 Connect Request TPDU
parameter ignored, 34
16:34:17.386 (0) Servxnt: Socket(1744fc4): Badly coded TP0 Connect Request
TPDU, total field sizes exceed stated header length
16:34:17.386 (3) Servxnt: Socket(1744fc4) sending data, length = 11:
16:34:17.386 (2) Servxnt: Socket(1744fc4): Termination of TP0 connection
to :192.168.169.61
16:34:17.386 (2) Servxnt: Socket(1744fc4): Local TP0 requesting close of
socket connection
16:34:17.386 (2) Servxnt: Socket(1744fc4): Incoming socket connection
received from :192.168.169.61
16:34:17.386 (3) Servxnt: Socket(1744fc4) received data, length = 105:
16:34:17.386 (0) Servxnt: Socket(1744fc4): Ignoring TP0 CR TPDU options, -
61
16:34:17.386 (0) Servxnt: Socket(1744fc4): Received TP0 Calling TSEL too
large: 124
```

Figure 8 – Compromise of LiveData Server

Based on our testing of the patched LiveData Utility Stack, we believe multiple vulnerabilities were addresses in addition to the issue that was reported. The patch to the heap overflow vulnerability also seemed to address a number of memory leaks. For example vulnerable versions of the LiveData server began to consume excessive CPU and memory resources under certain testing circumstances, and the patched versions did not.

4.3.2 Sisco OSI Protocol Vulnerability

While the previously discussed LiveData vulnerability was exploitable by a number of simple open source tools, the Sisco OSI protocol vulnerability requires the Digital Bond toolset or some similar Utility Stack protocol tool to identify and exploit. It is necessary to implement TPKT (RFC 2006) and establish a valid COTP connection prior to starting the attack. Neither of these events is trivial from a protocol availability or development viewpoint.

The transport-service-access-point (TSAP) value is required to establish a COTP connection. In our experience most utilities select simple, low values for TSAP, such as 1, 2, 3, …, so guessing the TSAP is not difficult. In fact, Digital Bond developed a Nessus SCADA plugin to identify if a low and easily guessed value was used for the TSAP value. A compensating control for this vulnerability rather than applying a patch would be to select secret and complex TSAP values.

Once a COTP connection is established, random invalid data causes the ICCP servers based on certain versions of the Sisco stack to crash, typically in about 15 to 20 seconds.

4.4 Conclusions on Known ICCP Vulnerabilities

So far, the vulnerabilities discovered in ICCP servers and OSI protocol stacks distributed by LiveData, Sisco, and Tamarack only have been taken as far as a denial of server to a single application process. Digital Bond does not develop exploit code and has made no attempt to determine whether arbitrary code could be executed. The denial of service was a serious enough issue to warrant a patch.

That said, more independent research may be necessary to more conclusively determine the risks of arbitrary code execution, especially given the large number of unpatched ICCP servers in use and the lack of any application layer filter/IPS capabilities to drop or reject malformed ICCP traffic.

4.5 Other Attacks Requiring More Research

The Utility Stack is a large attack surface that garners little attention from security researchers or attackers. So it is not surprising that there are many more areas that require additional research. Some of these areas that we are looking at include:

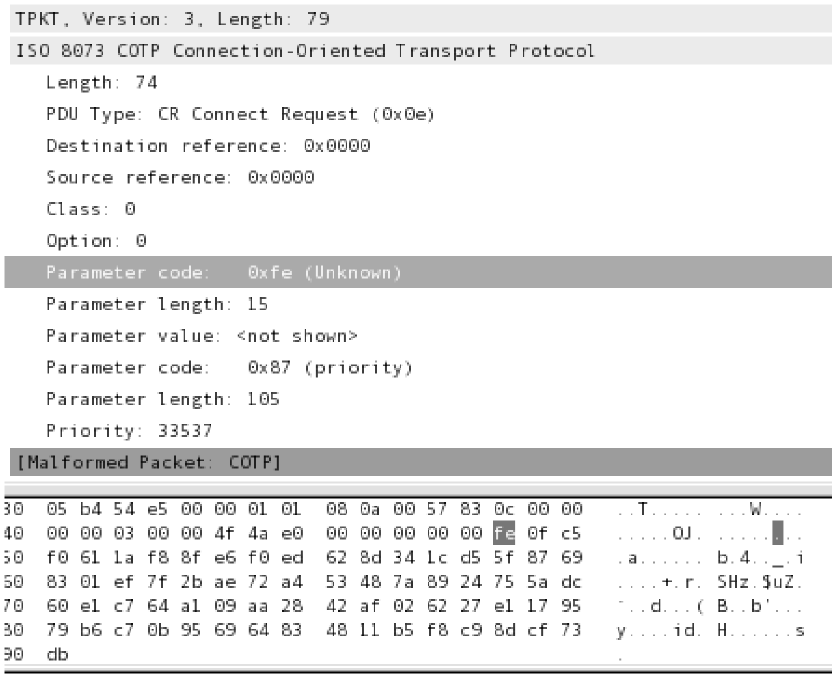
- COTP Connection Request Flooding (Think SYN Flooding for the COTP)
- COTP Disconnect Spoofing (Think RSTs)
- OSI Session Connect Flooding
- ICCP Replay Attacks (We have done replay attacks in the lab, but additional research is required to simulate what an attacker might attempt from a successful replay.)

5 A Vulnerability Discovery Toolset

While there are a number of commercial fuzzing tools that have been either specifically designed or modified to conduct robustness testing on SCADA protocols such as Modbus and DNP3, we are not aware of any other tools useful for discovering implementation vulnerabilities in ICCP servers or other OSI protocols in the Utility Stack. Digital Bond has developed such a toolset for use in a variety of consulting and research projects.

Selected modules of this toolset are made available to vetted asset owners to enable them to identify vulnerabilities in the ICCP servers they have deployed or are considering deploying. More details on obtaining the tool are available on Digital Bond's website.

We make no claims that the toolset covers the entire Utility Stack protocol space. In fact it only covers the TPKT, COTP, OSI Session Layer and OSI Presentation Layer protocols in the stack at the time of the publication of this paper. Nor do we claim any complete coverage of these protocols. Instead, the toolset was designed to test the protocols in a manner similar to the successful exploits of IT protocols since this is how an attacker is likely to approach the challenge of compromising an ICCP server. With this approach we hope to identify vulnerabilities that an attacker would be most likely to find prior to the attacker. Our main advantage over an attacker at this point is knowledge of this obscure protocol stack and access to the software and systems.



```
TPKT, Version: 3, Length: 79
ISO 8073 COTP Connection-Oriented Transport Protocol
   Length: 74
   PDU Type: CR Connect Request (0x0e)
   Destination reference: 0x0000
   Source reference: 0x0000
   Class: 0
   Option: 0
   Parameter code:   0xfe (Unknown)
   Parameter length: 15
   Parameter value: <not shown>
   Parameter code:   0x87 (priority)
   Parameter length: 105
   Priority: 33537
[Malformed Packet: COTP]

30  05 b4 54 e5 00 00 01 01  08 0a 00 57 83 0c 00 00   ..T..... ...W....
40  00 00 03 00 00 4f 4a e0  00 00 00 00 00 fe 0f c5   .....OJ. ........
50  f0 61 1a f8 8f e6 f0 ed  62 8d 34 1c d5 5f 87 69   .a...... b.4.._.i
60  83 01 ef 7f 2b ae 72 a4  53 48 7a 89 24 75 5a dc   ....+.r. SHz.$uZ.
70  60 e1 c7 64 a1 09 aa 28  42 af 02 62 27 e1 17 95   `..d...( B..b'...
80  79 b6 c7 0b 95 69 64 83  48 11 b5 f8 c9 8d cf 73   y....id. H......s
90  db                                                 .
```

Figure 9 – COTP Fuzzing Decode

6 ICCP Countermeasures

In this section we review some countermeasures for real and potential vulnerabilities discussed in this paper.

6.1 Secure ICCP

The Secure ICCP protocol that is now available and beginning to be widely deployed is falsely considered by many to address the security issues in this paper. It is important to understand what risks Secure ICCP will address and what risks it will not address.

The primary security measure in Secure ICCP is all ICCP traffic is sent in an SSL encrypted tunnel. This encrypted tunnel will limit the source of attacks. Today, most ICCP is sent over private networks with limited access, rather than the Internet, so there is already some limitation on who can communicate with and attack an ICCP server. An attacker who has gained access to the private network will be able to send packets to a Secure ICCP server, but since the attacker does not have the SSL key or a valid certificate he will not be able to properly encrypt or decrypt packets. Therefore an attacker with only private WAN access will likely be unable to attack the Utility Stack implementation protected by Secure ICCP.

However, Secure ICCP will not prevent an attacker who has access to a legitimate Secure ICCP server from attacking any other utilities with authorized security associations. The attacks discussed in this paper would simply be sent encrypted over the wide area network. So the concept of an ICCP worm discussed in Section 2 is equally valid for Secure ICCP. If a single utility is compromised, there is the very real potential for a cascading failure of Secure ICCP servers and unpatched SCADA servers.

Secure ICCP will not be a panacea that eliminates the need for a properly implemented Utility Stack.

6.2 IDS Signatures

Digital Bond released an initial set of ICCP signatures in 2005 that can be used to identify some anomalous activity directed against ICCP servers. Given the lack of disclosed vulnerabilities and the desire to limit information to attackers that might reverse engineer IDS signatures into active exploits, our efforts focused on alerting valid protocol messages from non-ICCP clients and responses from ICCP servers to malicious messages.

Due to the lack of the useful ASN.1 decoding functions in Snort, most of the IDS signatures focus on lower protocol layers. Alerting on server responses is clearly not adequate to prevent intrusions against ICCP servers. These IDS signatures are only a small fraction of what is required.

6.3 The Need for IPS

An initial first step, which is already in use by many utilities to mitigate the threats and vulnerabilities describe above is to closely restrict connectivity to ICCP servers based on IP address and the TCP/102. This is typically enforced by a firewall, and most firewalls today offer an IPS capability. So a natural countermeasure is to develop and deploy IPS to prevent attacks discussed in this paper.

Some simple examples of potentially effective IPS rules are:

1. Reset all "ICCP client" sessions that resulted in a COTP Disconnect (Protocol Error) from an ICCP Server
2. Drop all non-TPKT traffic sent over TCP/102
3. Drop all TPKT traffic with invalid length

7 Conclusions

The Utility Stack found in ICCP servers consists of much more than the ICCP/TASE.2 protocol. In fact it is a set of obscure protocols that offer a large attack surface. Initial research on Utility Stack implementations have identified vulnerabilities that are not unlike vulnerabilities found in IT protocols.

It is likely that there will be additional vulnerabilities identified in the widely used Utility Stacks. The impact of these vulnerabilities has the potential to have catastrophic consequences to the reliability of the electric grid. For this reason it is incumbent on the SCADA security community to identify and address these vulnerabilities prior to an adversary.

About the Author – Matthew Franz performed the research in this paper while a Senior Security Researcher at Digital Bond, Inc. In addition to ICCP security research, Mr. Franz helped Digital Bond develop Nessus SCADA plugins, a SCADA Honeynet, and vulnerability disclosure practices. Prior to joining Digital Bond, Mr. Franz led Cisco's control systems security efforts in the Critical Infrastructure Assurance Group (CIAG). Mr. Franz is currently a senior network security analyst at Hewitt Associates.

References

[1] US-CERT. Vulnerability Note VU#190617 – LiveData ICCP Server heap buffer overflow vulnerability, May 2006.

[2] US-CERT. Vulnerability Note VU#372878 – Tamarack MMSd components fail to properly handle malformed packets, July 2006.

[3] US-CERT. Vulnerability Note VU#468798 – Sisco OSI stack fails to properly validate packets, September 2006.

[4] International Electrotechnical Commission. IEC 60870-6-503 Telecontrol equipment and systems – Part 6-503: Telecontrol protocols compatible with ISO standards and ITU-T recommendations – TASE.2 Services and protocol, Second edition, 2002-04.

[5] Katipamula, S. Hadley, M.D. and T.P. McKenna. Evaluation of Symantec Security Products in an Areva T&D – Implemented SCADA Environment During ICCP Communication Servers, May 2004.

[6] Cass, Dwight E. and Rose, Marshall T. IETF RFC 1006 – ISO Transport Service on top of the TCP Version: 3, May 1987.

[7] International Electrotechnical Commission. ISO/IEC 8073 – Open Systems Interconnection – Protocol for providing the connection-mode transport service, 1997.