

A Review on Collecting Fault and Disturbance Data in the Modern Substation

Author: Zach Makki

Affiliation: Softstuf Inc.

Author Email: zach@softstuf.com

Country: United States of America

1 – Abstract

Today a number of standard protocols are commonly used to collect fault and disturbance data. These include the file transfer protocol (FTP), the secure FTP (FTPS), the SSH FTP (SFTP), and the IEC61850 manufacturer message specification protocol (MMS). The intent of the author is to inform readers about the challenges encountered when attempting to use these protocols to collect fault and disturbance data records securely and effectively from various types of digital relays and digital fault recorders, and to make recommendations on how some of these challenges can be mitigated.

The paper is composed of five sections, with the first being the abstract. The second section gives a brief introduction, providing definitions for fault and disturbance data, and the sources which produce it. The third section discusses the challenges that come with supporting certain protocols, highlighting common issues that can be encountered when collecting data records in the field. The fourth section provides examples of security vulnerabilities that come with the use of certain protocols and how these vulnerabilities can be addressed with complimentary solutions such as serial data diodes. The fifth and final section will be a brief conclusion.

2 – Introduction

Fault and disturbance data which includes but is not limited to fault records (FRs), sequence of events records (SERs), dynamic disturbance records (DDR) [1], settings files and health checks, is used to evaluate the reliability of the power system and the performance of the protection system. Historically fault and disturbance data records were mainly produced by digital fault recorders (DFRs). However, in modern substations there are now a variety of sources which produce fault and disturbance data records, including but not limited to DFRs, digital relays, remote terminal units (RTUs), and digital meters [2]. This increase in sources has created a large increase in the sheer volume of available fault and disturbance data. Much has been written about this increase in volume and the uses that can be made from the large data sets produced by it. This paper will focus on the critical first step in attempting to use this data, data collection. The fault and disturbance data collection process is done via standard data transfer protocols most commonly run over the transmission control protocol (TCP). Historically a wide range of both standard and proprietary protocols have been used, but modern equipment most commonly uses one of four standard protocols, FTP, FTPS, SFTP, and MMS.

3 – Frequently Encountered Collection Issues using Standard Protocols

3.1 – File Transfer Protocol (FTP)

The first version of FTP can be traced back to 1971 with the release of RFC 114 [3], making this by far the oldest of the four standard protocols covered in this paper. Today FTP is arguably the most used standard protocol for the collection of fault and disturbance data records in North America. This prevalence can be attributed to the protocols age and relative ease of use. Although FTP is mostly reliable there can be some challenges in attempting to automatically collect fault and disturbance data records from multiple devices. One of the most frequently encountered challenges in attempting this is the lack of consistency in file directory listing formats found in implementations of FTP servers. To request a file directory listing an FTP client issues the FTP file directory listing command, the FTP server upon receiving this command will check to ensure the directory exists, then send a formatted list of files and

folders stored in that directory to the client. The FTP command issued to receive a directory listing can be either LIST, MLSD, or NLIST depending on the implementation of the FTP server. The NLIST command will only return the name of files and folders without any additional information such as file creation time or file size, therefore this command is not useful for automatic collection of fault and disturbance data records. The LIST command was first defined in RFC 765 [4]. In both RFC 765 and RFC 959 the LIST command does not have a defined format, in fact in RFC 959 under the LIST command definition it states, "Since the information on a file may vary widely from system to system, this information may be hard to use automatically in a program but may be quite useful to a human user [5]." This lack of a defined format led to the use of proprietary directory listings on a large scale.

As one can imagine this large number of directory listing formats created major issues when attempting to automatically collect data from various types of FTP servers. To solve this problem a revision was made to the FTP standard in RFC 3695 adding the MLSD command which defined a strict but extensible format for FTP directory listings [6]. An example of the format defined by the MLSD command is listed below.

```
type=cdir;szd=4096;modify=20181030022936;UNIX.mode=0755;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g17ea9d; .
type=pdir;szd=4096;modify=20181030022936;UNIX.mode=0755;UNIX.uid=1004;UNIX.gid=99;unique=fd01g15ffae; ..
type=file;size=1011;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g180228; DR1_000001.cfg
type=file;size=8909;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g180227; DR1_000001.dat
type=file;size=30;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g180226; DR1_000001.hdr
type=file;size=1008;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g180225; DR1_000002.cfg
type=file;size=8806;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g180224; DR1_000002.dat
type=file;size=30;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g180223; DR1_000002.hdr
type=file;size=410546;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g180222; DR1_000003.dat
type=file;size=33;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g180221; DR1_000003.hdr
type=file;size=2510;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g180220; DR1_0001.CFG
type=file;size=24334;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g18021f; DR1_0001.DAT
type=file;size=2512;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g18021e; DR1_0002.CFG
type=file;size=24472;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g18021c; DR1_0002.DAT
type=file;size=23230;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g18021d; DR1_0003.DAT
type=file;size=2510;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g18021b; DR1_0004.CFG
type=file;size=23230;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g18021a; DR1_0004.DAT
type=file;size=2510;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g180219; DR1_0005.CFG
type=file;size=24564;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g180218; DR1_0005.DAT
type=file;size=2510;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g180217; DR1_0006.CFG
type=file;size=23230;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g180216; DR1_0006.DAT
type=file;size=2510;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g180215; DR1_0007.CFG
type=file;size=23368;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g180214; DR1_0007.DAT
type=file;size=2510;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g180213; DR1_0008.CFG
type=file;size=23230;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g180212; DR1_0008.DAT
type=file;size=2510;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g180211; DR1_0009.CFG
type=file;size=23230;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g180210; DR1_0009.DAT
type=file;size=2510;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g18020f; DR1_0010.CFG
type=file;size=23230;modify=20181030022936;UNIX.mode=0644;UNIX.uid=1004;UNIX.gid=1003;unique=fd01g18020e; DR1_0010.DAT
```

Figure 1 – MLSD Format

While the MLSD command has been adopted by some devices in the field, many that are active today have not adopted this command. In the area of fault and disturbance data collection, this poses a significant issue. In modern substations it is expected that fault and disturbance data records will be collected from a large number of devices in an automatic and timely manner. Devices which still support a directory listing format not designed for automatic collection make this difficult.

Until all devices support the MLSD command the issue of having to support multiple directory listing formats will exist. In an FTP server there is no way to know which directory listing format is used until issuing the LIST command and evaluating the response. Therefore, the only way to automatically parse LIST formats is by having pre-build parsers for each type of format used. For example, a collection of legacy FTP directory listings still used in substations today are displayed below.

```
00/00/80 00:00:00 <DIR> .
00/00/80 00:00:00 <DIR> ..
29/01/21 12:52:04 51200 02340001.DAT
29/01/21 12:52:04 2019 02340001.CFG
```

Figure 2 – Windows-Style Listing Format

```

drw-rw-rw- 1 ftproot ftp          0 Jan 6 2005 .
drw-rw-rw- 1 ftproot ftp          0 Jan 6 2005 ..
-rw-rw-rw- 1 ftproot ftp    3131708 Jul 16 4:15 TR1.RAW
-rw-rw-rw- 1 ftproot ftp    125386 Jul 13 3:07 SERLOG.CSV
-rw-rw-rw- 1 ftproot ftp    499990 Jul 16 4:10 SOELOG.TXT
-rw-rw-rw- 1 ftproot ftp     18718 Jul 13 3:07 VFT1.DAT
-rw-rw-rw- 1 ftproot ftp        21 Jul 13 3:07 VFT2.DAT
-rw-rw-rw- 1 ftproot ftp     6819 Jul 13 3:07 TR1.INI
-rw-rw-rw- 1 ftproot ftp        278 May 10 2018 VFT1.INI
-rw-rw-rw- 1 ftproot ftp        275 May 10 2018 VFT2.INI
-rw-rw-rw- 1 ftproot ftp     1399 Jul 27 2018 DR1_0002.ZIP
-rw-rw-rw- 1 ftproot ftp     1344 May  8 11:08 DR1_0155.ZIP
-rw-rw-rw- 1 ftproot ftp     1342 Jul 29 2018 DR1_0003.ZIP
-rw-rw-rw- 1 ftproot ftp    170501 May 11 14:41 WR1_0155.ZIP
-rw-rw-rw- 1 ftproot ftp        1406 Jul 31 2018 DR1_0004.ZIP
-rw-rw-rw- 1 ftproot ftp    170963 May 13 16:30 WR1_0156.ZIP

```

Figure 3 – Unix-Style Listing Format

size	date	time	name
8192	Dec-02-2020	10:52:24	.
8192	Dec-02-2020	10:52:22	..
79247	Apr-15-2018	11:42:02	drec_216.zip
2696	Apr-15-2018	11:42:02	drec_216h.zip
93142	Apr-15-2018	12:27:30	drec_217.zip
2882	Apr-15-2018	12:27:30	drec_217h.zip
81325	Apr-15-2018	12:31:02	drec_218.zip
2784	Apr-15-2018	12:31:02	drec_218h.zip
83321	Apr-15-2018	12:44:42	drec_219.zip
2781	Apr-15-2018	12:44:42	drec_219h.zip
86427	Apr-15-2018	12:50:30	drec_220.zip
2915	Apr-15-2018	12:50:30	drec_220h.zip

Figure 4 – A Legacy Proprietary Directory Listing Format

```

50AABCD6B604,11/19/2012 23:12:00.000,221675,
50AABCD3D44,11/19/2012 23:12:00.000,215809,
50AABD0AC284,11/19/2012 23:13:00.000,216933,
50AABD1AD544,11/19/2012 23:13:00.000,215101,
50AABD69DB84,11/19/2012 23:14:00.000,221147,
50AABD7B4584,11/19/2012 23:15:00.000,220748,
50AABD9C32C4,11/19/2012 23:15:00.000,221385,
50AABDA38204,11/19/2012 23:15:00.000,220631,
50AABDB2F8C4,11/19/2012 23:16:00.000,221096,
50AABDBE98C4,11/19/2012 23:16:00.000,221234,
50AABDC5B1C4,11/19/2012 23:16:00.000,216070,
50AABDE67784,11/19/2012 23:16:00.000,215616,

```

Figure 5 – Another Legacy Proprietary Directory Listing Format

08/28/2018 12:00AM	146,160	180828,131951515,-St,Radnor,USI M9000,USI,R03F3.dat
08/28/2018 12:00AM	200	180828,131951515,-St,Radnor,USI M9000,USI,R03F3.inf
08/28/2018 12:00AM	5,556	180828,132417553,-St,Radnor,USI M9000,USI,R03F4.cfq
08/28/2018 12:00AM	143,080	180828,132417553,-St,Radnor,USI M9000,USI,R03F4.dat
08/28/2018 12:00AM	200	180828,132417553,-St,Radnor,USI M9000,USI,R03F4.inf
08/30/2018 12:00AM	5,555	180830,130533659,-St,Radnor,USI M9000,USI,R03F5.cfq
08/30/2018 12:00AM	139,300	180830,130533659,-St,Radnor,USI M9000,USI,R03F5.dat
08/30/2018 12:00AM	200	180830,130533659,-St,Radnor,USI M9000,USI,R03F5.inf
08/30/2018 12:00AM	5,644	180830,233319450,-St,Radnor,USI M9000,USI,R03F6.cfq
08/30/2018 12:00AM	261,138	180830,233319450,-St,Radnor,USI M9000,USI,R03F6.dat
08/30/2018 12:00AM	206	180830,233319450,-St,Radnor,USI M9000,USI,R03F6.inf
09/03/2018 12:00AM	5,556	180903,210842190,-St,Radnor,USI M9000,USI,R03F7.cfq
09/03/2018 12:00AM	140,980	180903,210842190,-St,Radnor,USI M9000,USI,R03F7.dat
09/03/2018 12:00AM	200	180903,210842190,-St,Radnor,USI M9000,USI,R03F7.inf
09/10/2018 12:00AM	5,556	180910,121926148,-St,Radnor,USI M9000,USI,R03F8.cfq
09/10/2018 12:00AM	146,300	180910,121926148,-St,Radnor,USI M9000,USI,R03F8.dat
09/10/2018 12:00AM	200	180910,121926148,-St,Radnor,USI M9000,USI,R03F8.inf
09/11/2018 12:00AM	5,556	180911,132857922,-St,Radnor,USI M9000,USI,R03F9.cfq
09/11/2018 12:00AM	141,400	180911,132857922,-St,Radnor,USI M9000,USI,R03F9.dat

Figure 6 – And Another Legacy Proprietary Directory Listing Format

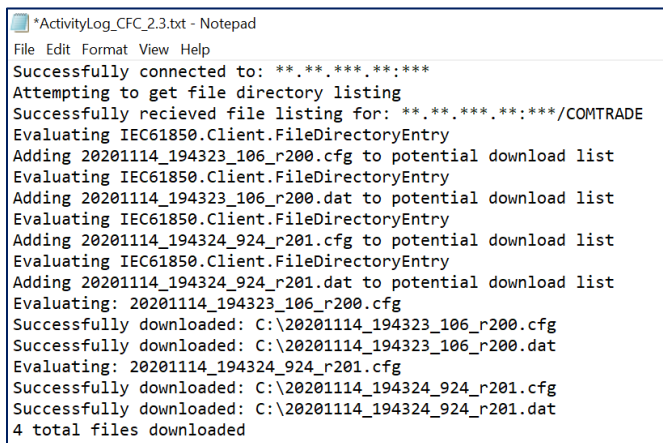
3.2 – Manufacturing Message Specification (MMS)

MMS is an international standard published by the International Organization for Standardization (ISO) in December of 1988. The goal of the MMS standard is to create an internationally standardized messaging system for exchanging real-time data and supervisory control information between networked devices

and/or computer applications in a manner that is independent of the application being performed or the developer of the device or application. The specifications of the MMS standard are explained in detail in [7]. IEC 61850 section 61850-8-1 maps the abstract model defined by 61850 to MMS, thereby making MMS the standard method of point-to-point communication in devices which support 61850 [8]. MMS was chosen in 61850 because it supports the standard's complex naming system and services. This has helped lead to MMS being, along with FTP, one of the most used standard protocols for fault and disturbance data collection. Unlike FTP, MMS servers have a standard directory listing format across all implementations which does not present issues when automatically collecting records. The few problems that have been encountered when attempting automatic collection via MMS are detailed below.

The first and most prevalent issue that has been encountered when attempting automatic collection via MMS is the formatting of the request directory listing command. To request a directory listing from an MMS server the MMS client must issue the list file directory command with the desired directory name as a passed parameter. When attempting to issue this command inconsistencies have been found in how different implementations of MMS respond. These inconsistencies stem from the use of forward slashes in the desired directory name. MMS servers will respond with the file directory listing if the correct number of forward slashes are used, and this varies in different implementations. For example, some MMS servers will only successfully respond to this command if the desired directory name is formatted as: "/desired_directory". Others will only successfully respond if it is formatted as "/desired_directory/", and furthermore some do not require the use of forward slashes at all.

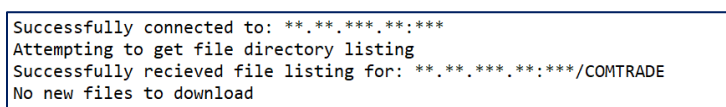
Due to the frequency with which this issue is being encountered some developers have removed automatic formatting of desired directory names, leaving it up to the user to specify the appropriate forward slashes during MMS connections. This lack of consistency has occasionally caused field engineers to attempt multiple connections to an MMS server, trying different formatting of the desired directory name before successfully being able to receive a listing. Examples of log files showing these inconsistencies are listed below. In the first example an MMS server successfully responds to the list file directory command when one forward slash at the beginning of the desired directory name was used. This log is used to keep track of all communications between the MMS client and the MMS server.



```
*ActivityLog_CFC_2.3.txt - Notepad
File Edit Format View Help
Successfully connected to: **.**.**.**:***
Attempting to get file directory listing
Successfully recieved file listing for: **.**.**:***/COMTRADE
Evaluating IEC61850.Client.FileDirectoryEntry
Adding 20201114_194323_106_r200.cfg to potential download list
Evaluating IEC61850.Client.FileDirectoryEntry
Adding 20201114_194323_106_r200.dat to potential download list
Evaluating IEC61850.Client.FileDirectoryEntry
Adding 20201114_194324_924_r201.cfg to potential download list
Evaluating IEC61850.Client.FileDirectoryEntry
Adding 20201114_194324_924_r201.dat to potential download list
Evaluating: 20201114_194323_106_r200.cfg
Successfully downloaded: C:\20201114_194323_106_r200.cfg
Successfully downloaded: C:\20201114_194323_106_r200.dat
Evaluating: 20201114_194324_924_r201.cfg
Successfully downloaded: C:\20201114_194324_924_r201.cfg
Successfully downloaded: C:\20201114_194324_924_r201.dat
4 total files downloaded
```

Figure 7 – Data Collection Log

In the next example an MMS server does not successfully respond to the list file directory command when one forward slash was used at the beginning of the desired directory name, as in the previous example.



```
Successfully connected to: **.**.**:***
Attempting to get file directory listing
Successfully recieved file listing for: **.**:***/COMTRADE
No new files to download
```

Figure 8 – Data Collection Log 2

The log here states that it has successfully received the file directory listing, but the list returned by the MMS server contains no objects. The log states this has been successful due to a lack of exception being thrown by the MMS server. This presents a problem within itself as at times it can be difficult to determine if a directory is empty, or if the wrong number of forward slashes were used in the list directory command. The next example shows the same MMS server responding to the list file directory command when two forward slashes are used, one in the front of the name, and one trailing the name.

```
Successfully connected to: **.***.***.***
Attempting to get file directory listing
Successfully recieved file listing for: **.***.***.***:/COMTRADE/
Adding 20200722_173243_882_r1949.cfg to potential download list
Evaluating IEC61850.Client.FileDirectoryEntry
Adding 20200722_173243_882_r1949.dat to potential download list
Evaluating IEC61850.Client.FileDirectoryEntry
Adding 20200722_173247_010_r1950.cfg to potential download list
Evaluating IEC61850.Client.FileDirectoryEntry
Adding 20200722_173247_010_r1950.dat to potential download list
Evaluating: 20200722_173243_882_r1949.cfg
Successfully downloaded: C:\20200722_173243_882_r1949.cfg
Successfully downloaded: C:\20200722_173243_882_r1949.dat
Evaluating: 20200722_173247_010_r1950.cfg
Successfully downloaded: C:\20200722_173247_010_r1950.cfg
Successfully downloaded: C:\20200722_173247_010_r1950.dat
4 total files downloaded
```

Figure 9 – Data Collection Log 3

Another prevalent issue encountered when attempting to automatically collect records via MMS is a non-defined file last modified time. In MMS, a files last modified time is listed in UNIX time, which is the number of seconds since 01/01/1970. Frequently MMS servers will return a directory listing with each file's last modified time listed as 0. This makes it difficult for an MMS client to keep track of which files it has already collected (as to do so the client needs to know the time the file was created). Without access to the file creation time an MMS client may need to download every file in each directory it connects to.

A good work around for this issue is to download a COMTRADE records CFG file, open the file and retrieve the records trigger time. The trigger time can then be used to determine if a client has previously collected the record. If the client has previously collected the record the CFG file will be deleted, if not the rest of the record will be downloaded. While this work around does accomplish its goal, it only works with COMTRADE records, and also slows down the automatic collection process. This slowdown poses regulatory issues for certain countries. For example, in Chile it is a regulatory requirement for fault and disturbance data records to be collected from a device in under 60 seconds [9]. For devices which contain hundreds of records and do not return reliable last modified times, this regulatory requirement will be almost impossible to meet. An example of a directory listing with the files last modified times set to 0 is listed below. In this example the file's last modified time is written after the filename in each line which begins with the word "Evaluating".

```
Successfully connected to: **.***.***.***:102
Attempting to get file directory listing
Successfully recieved file listing for: **.***.***.***:102/COMTRADE/
Evaluating /COMTRADE/HR_10113.CFG|0|43008
Adding /COMTRADE/HR_10113.CFG to potential download list
Evaluating /COMTRADE/HR_10112.CFG|0|43008
Adding /COMTRADE/HR_10112.CFG to potential download list
Evaluating /COMTRADE/HR_10111.CFG|0|43008
Adding /COMTRADE/HR_10111.CFG to potential download list
Evaluating /COMTRADE/HR_10110.CFG|0|43008
Adding /COMTRADE/HR_10110.CFG to potential download list
Evaluating /COMTRADE/HR_10113.DAT|0|380928
Adding /COMTRADE/HR_10113.DAT to potential download list
Evaluating /COMTRADE/HR_10112.DAT|0|380928
Adding /COMTRADE/HR_10112.DAT to potential download list
Evaluating /COMTRADE/HR_10111.DAT|0|380928
Adding /COMTRADE/HR_10111.DAT to potential download list
Evaluating /COMTRADE/HR_10110.DAT|0|380928
Adding /COMTRADE/HR_10110.DAT to potential download list
```

Figure 10 – Data Collection Log 4

Another issue also occurs with the file size listed as 0 for all files in a directory listing, regardless of their size. This issue is difficult to work around as most MMS servers will throw an “Object Does Not Exist” exception when a client attempts to download a file of size 0. If the file size is listed as 0 for all files, the only way to determine if a file has content is by attempting to download it and catching the “Object Does Not Exist” exception. An example of a directory listing received from a device, active in the field, is listed below. In this example the file size is written at the end of each line which begins with the word “Evaluating”.

```
Evaluating LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc134.hdr|1620826156000|0
Adding LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc134.hdr to potential download list
Evaluating LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc134.dat|1620826156000|0
Adding LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc134.dat to potential download list
Evaluating LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc133.cfg|1620826156000|0
Adding LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc133.cfg to potential download list
Evaluating LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc133.hdr|1620826156000|0
Adding LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc133.hdr to potential download list
Evaluating LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc133.dat|1620826156000|0
Adding LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc133.dat to potential download list
Evaluating LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc132.cfg|1620826156000|0
Adding LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc132.cfg to potential download list
Evaluating LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc132.hdr|1620826156000|0
Adding LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc132.hdr to potential download list
Evaluating LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc132.dat|1620826156000|0
Adding LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc132.dat to potential download list
Evaluating LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc131.cfg|1620826156000|0
Adding LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc131.cfg to potential download list
Evaluating LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc131.hdr|1620826156000|0
Adding LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc131.hdr to potential download list
Evaluating LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc131.dat|1620826156000|0
Adding LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc131.dat to potential download list
Evaluating LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc130.cfg|1620826156000|0
Adding LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc130.cfg to potential download list
Evaluating LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc130.hdr|1620826156000|0
Adding LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc130.hdr to potential download list
Evaluating LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc130.dat|1620826156000|0
Adding LD/EP_TRANSFORMER1_GE_T35_1/COMTRADE/Osc130.dat to potential download list
```

Figure 11 – Data Collection Log 5

3.3 – FTP Secured Via TLS (FTPS)

In 2005 FTPS was introduced to create a more secure version of FTP. FTPS is FTP secured using the Transport Layer Security (TLS) protocol [10]. While FTPS is a more secure version of FTP, it is far less used in the collection of fault and disturbance data than FTP or MMS. Nevertheless, issues that have been encountered with implementations of FTPS are detailed here.

Implementations of FTPS can suffer from the same issues with directory listings as FTP implementations do. This is due to the MLSD command not being officially added to FTP until 2007, two years after FTPS was defined. Along with directory listing issues FTPS presents unique challenges to software developers attempting to support it, as multiple programming languages do not properly support this protocol. The .NET platform is a programming platform developed by Microsoft, it includes programming languages such as C#, and Visual Basic, which according to the TIOBE index for June 2021 [11] are among the top six used programming languages in the world. Currently programming languages developed and maintained on the .NET platform do not properly support FTPS. The reason for this is the underlying TLS/SSL class used by this platform does not properly support TLS session resumption, rendering any FTPS client built on top of this class unable to connect to FTPS servers which require TLS session resumption [12]. TLS session resumption was introduced in 2008 and provides a mechanism to resume or share the same negotiated secret key between multiple connections. Client and server both store this session ID along with the session keys and connection states. To resume a session, the client sends the stored session ID with the first protocol message to the server. If the server recognizes the connection and is willing to resume the session, it replies with the same session ID to re-establish the respective session [13]. This both prevents hackers from attempting to mimic a connection and reduces latency and computing costs as a new TLS handshake will not be needed to resume a session. Today TLS session resumption is considered a standard security setting in servers which support the FTPS protocol. Currently the only way to connect to an FTPS server with software built on the .NET platform is to disable the “Require TLS Session Resumption” setting. This is not a recommended practice as it makes the connection both less secure and more computationally heavy.

This issue has been open with the .NET development team and unchanged since November of 2018. One reason for this lack of correction is the development team has stated that it views FTP and FTPS as legacy protocols. In fact, Windows has depreciated their built in FTP connection class “FtpWebRequest” without offering a replacement for it (Windows now recommends using third party libraries for FTP). One piece of good news for developers waiting on this correction is the lack of TLS session resumption support in the .NET platform has created issues for other popular standard transfer protocols, most notably HTTPS. HTTPS is a file transfer protocol secured via TLS which has recently become popular for online communications. Implementations of HTTPS often make use of TLS session resumption but unlike FTPS do not require it. Due to the .NET platforms lack of TLS session resumption support, HTTPS clients built on top of this platform will experience a performance slowdown when compared to clients built on other platforms. This has caught the attention of the .NET development team and in the past year multiple threads opened on the .NET GitHub repository show potential solutions to this issue [14][15]. The hope is that one of these solutions is implemented in the near future.

3.4 – SSH File Transfer Protocol (SFTP)

SFTP was first introduced as a file transfer protocol over the secure shell protocol (SSH) in an RFC draft released in October of 2001 [16]. Unlike FTP and FTPS no standard RFC for SFTP was ever published. According to SFTP.net a website created by developers of REBEX, a popular SFTP library, most implementations of SFTP are based on the initial draft released in 2001. Multiple drafts have been released since as updates to the protocol but other then a few extensions taken from these drafts they are mostly unused. Similar to FTPS, SFTP is not used in the collection of fault and disturbance records as often as FTP or MMS. Of the few implementations that have been tested none presented any challenges to the automatic collection process.

4 –FTP and MMS Security Vulnerabilities

4.1 – FTP Vulnerabilities

Security concerns with the use of FTP are well documented [17] and were the major reason for the development of FTPS. The original FTP standard did not define any security mechanisms which can be used to encrypt data across a connection, passing all commands and data in plain text [18]. If FTP is used across an unsecured network sensitive information such as the username and password are vulnerable to a man in the middle attack, where a third-party hacker can monitor or intercept data during a connection. A man in the middle attack is not the only security concern with the use of FTP, brute force attacks, and port stealing are also common security vulnerabilities for FTP servers. A brute force attack is when a hacker attempts to gain access to a password protected server by password guessing with a long sequence of frequently used passwords. FTP servers are vulnerable to this attack due to the fact that a default FTP implementation will allow an unlimited number of attempts at entering a user's password. Port stealing is when an attacker decodes the particular order or pattern in which an operating system dynamically defines its port numbers and identifies the next port number which will be used. By illegally gaining access to a port number, the legitimate client will be denied, and access will be granted to the hacker. FTP is vulnerable to this type of attack due to the lack of encryption used in data connections. Due to these vulnerabilities, it is recommended that any device which supports FTP should be placed inside a secured network, and those not placed in a secured network should consider switching to more secured protocol such as FTPS or SFTP.

4.2 – MMS Vulnerabilities

Similar to FTP, security vulnerabilities in the MMS standard [19] are well documented. In [19] it states that “When implementing MMS in secure or safety critical applications, features of the OSI security architecture may need to be implemented. This International Standard provides simple facilities for authentication (passwords) and access control. Systems requiring a higher degree of security will have to consider features beyond the scope of this International Standard. This International Standard does not

provide facilities for non-repudiation.” This statement indicates that information security was not a high priority during the development of the MMS standard. In [20] the simple facilities for access control added to the standard known as the “accessControlList” are examined. This paper finds that the optional nature of the “accessControlList” does not provide adequate access control security in most implementations of MMS, making them vulnerable to replay attacks. A replay attack is where an entity with access to a network intercepts a packet then replays that packet at a later time to disrupt the transfer process. If MMS is used in an unsecured network, the possibility of replay attacks will exist. In [21] various studies that have been conducted on MMS vulnerabilities are examined. This paper finds that implementations of MMS in unsecured environments can also be vulnerable to man-in-the-middle and DoS attacks. To address these vulnerabilities IEC 62351-4 specified a method of securing the MMS standard. The method specified to secure MMS is explained in detail in [21] and [22]. In short, 612351-4 recommends using TLS to secure MMS communications, similar to how FTPS made use of TLS to create a more secure version of FTP. In [23] a study is conducted on the TLS MMS proposed in 612351-4. This study finds that this method successfully secures a connection without adding a large amount of computational overhead or latency. While [23] found TLS MMS to be a viable secure alternative to MMS, no implementations of this have been found during field tests. This can be confirmed due to the fact that the TLS version of MMS uses port 3782 instead of the default MMS port 102, and no device has been encountered during testing which makes use of this port. Therefore, it is recommended that any device currently supporting MMS should be placed within a secure network, and all devices supporting MMS currently not placed within a secure network should consider adopting the TLS MMS defined by 62351-4.

An example of FTP or MMS relays deployed in a secure network is displayed below. In the display multiple FTP or MMS relays are deployed inside a substation electronic security perimeter (ESP). Inside the ESP the FTP or MMS relays can be automatically polled via ethernet without worry of a third-party monitoring or intercepting data in transfer. Data records polled from the relays will be periodically moved from the ethernet server where the FTP or MMS client is stored to a serial data diode. The data diode concept has been around since the inception of the serial port technology, over 40 years ago. The concept of the technology is simple, we either transmit, receive or we do both. If we do not do both then we have a data diode. The diode inside the ESP is used to securely transport data records to another diode outside the ESP. The resulting data records are then transferred via Ethernet to a shared drive on the utility corporate network. This example is just one of the ways that FTP and MMS relays can be placed within a secured network.

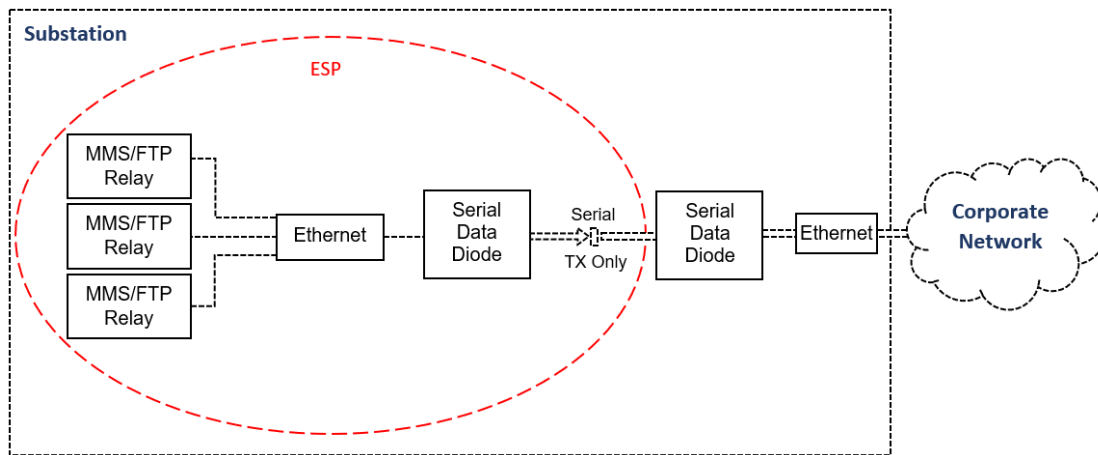


Figure 12 – Using serial data diodes to secure data transfer from a substation ESP to a utility corporate network

5 – Conclusion

In summary there are multiple issues that can be encountered when attempting to automatically collect fault and disturbance records in the modern substation. In this paper various issues which have been found in attempts to automatically collect fault and disturbance data from multiple devices were examined

in detail. These issues include but are not limited to FTP legacy directory listings, inconsistencies in the formatting of MMS directory listing requests, and non-defined file last modified times in MMS directory listings. This paper also briefly covered well known security vulnerabilities in the two most commonly used protocols, FTP and MMS, for fault and disturbance collection. The good news is solutions to these issues have for the most part already been developed, such as the MLSD command for FTP directory listings, and data diodes to securely collect and transfer fault and disturbance data. It is the hope of this author that more of these solutions will be implemented in the near future.

6 – References

1. North American Electric Reliability Corporation (NERC). "PRC-002". September 2015.
2. A. Makki, M. Makki, G. Semati, T. Giuliani. "Smart Data Concentrator (SDC) for Serially Accessed Devices in the Substation". *ECNE Engineering Conference*, March 2005.
3. Internet Engineering Task Force (IETF). "RFC 114 A FILE TRANSFER PROTOCOL". April 1971.
4. Internet Engineering Task Force (IETF). "RFC 765 File Transfer Protocol Specification". June 1980.
5. Internet Engineering Task Force (IETF). "RFC 959 File Transfer Protocol". October 1985.
6. Internet Engineering Task Force (IETF). "RFC 3695 Extensions to FTP". March 2007.
7. SISCO Inc. "Overview and Introduction to the Manufacturing Message Specification (MMS) Revision 2". November 1995.
8. M. Adamiak, D. Baigent, R. Mackiewicz. "IEC 61850 Communication Networks and Systems in Substations: An Overview for Users". *The Protection & Control Journal*, 2004.
9. Chilean National Electrical Coordinator. "Technical Specifications for the Implementation of the Concentrator System and External Communication Network of the Coordinates Remote Reading Protection System". 2019.
10. Internet Engineering Task Force (IETF). "Securing FTP with TLS". October 2005.
11. TIOBE. "TIOBE Index for June 2021". <https://www.tiobe.com/tiobe-index/>.
12. Microsoft. ".NET Platform – Issue #27916" GitHub Repository. November 2018. [FtpWebRequest not reusing ssl session on linux \(FTPS\) · Issue #27916 · dotnet/runtime \(github.com\)](#)
13. Internet Engineering Task Force (IETF). "Stateless TLS Session Resumption". January 2008.
14. Microsoft. ".NET Platform – Conversation #32763" GitHub Repository". February 2020. [\[wip\] fix handling of TLS session tickets on Linux servers by wfurt · Pull Request #32763 · dotnet/runtime \(github.com\)](#)
15. Microsoft. ".NET Platform – Issue #49845" GitHub Repository". March 2021. [HTTPS handshake performance · Issue #49845 · dotnet/runtime \(github.com\)](#)
16. Internet Engineering Task Force (IETF). "SSH File Transfer Protocol draft-ietf-secsh-filexfer-02.txt". October 2001.
17. Internet Engineering Task Force (IETF). "FTP Security Considerations". May 1999.
18. D. Springall, Z. Durumeric and J. A. Halderman, "FTP: The Forgotten Cloud". *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2016.
19. International Standard Organization, "Manufacturing Message Specification. Part 1, ISO". ISO Standard ISO 9506-1:2003(E), 2003.
20. J.T. Sorensen, M.G. Jaatun. "A Description of the Manufacturing Message Specification (MMS)". *SINTEF ICT MEMO*. August 2007.
21. S. M. S. Hussain, T. S. Ustun and A. Kalam, "A Review of IEC 62351 Security Mechanisms for IEC 61850 Message Exchanges," in *IEEE Transactions on Industrial Informatics*, September 2020.
22. S. Fries, H.J. Hof, T. Dufaure, M.G. Seewald. "Security for the Smart Grid – Enhancing IEC 62351 to Improve Security in Energy Automation Control". *International Journal on Advances in Security*. 2010.

23. O. Khaled, A. Marín, F. Almenares, P. Arias, and D. Díaz. "Analysis of Secure TCP/IP Profile in 61850 Based Substation Automation System for Smart Grids". *International Journal of Distributed Sensor Networks*. 2016.