PPPL-5166

SECURING MDSPLUS FOR THE NSTX-U DIGITAL COIL PROTECTION SYSTEM

Gregory J. Tchilinguirian, Keith G. Erickson

July 2015

Prepared for the U.S. Department of Energy under Contract DE-AC02-09CH11466.

SECURING MDSPLUS FOR THE NSTX-U DIGITAL COIL PROTECTION SYSTEM

Gregory J. Tchilinguirian, Keith G. Erickson
Princeton Plasma Physics Laboratory, Princeton NJ, USA
gtchilin@pppl.gov

NSTX used MDSplus extensively to record data, relay information and control data acquisition hardware. For NSTX-U the same functionality is expected as well as an expansion into the realm of securely maintaining parameters for machine protection. Specifically, we designed the Digital Coil Protection System (DCPS) to use MDSplus to manage our physical and electrical limit values and relay information about the state of our acquisition system to DCPS. Additionally, test and development systems need to use many of the same resources concurrently without causing interference with other critical systems. Further complications include providing access to critical, protected data without risking changes being made to it by unauthorized users or through unsupported or uncontrolled methods either maliciously or unintentionally. To achieve a level of confidence with an existing software system designed with minimal security controls, a number of changes to how MDSplus is used were designed and implemented. Trees would need to be verified and checked for changes before use. Concurrent creation of trees from vastly different use-cases and varying requirements would need to be supported. This paper will further discuss the impetus for developing such designs and the methods used to implement them.

Keywords— NSTX; Control; Protection; Real-time; Data

I. INTRODUCTION

MDSplus is a versatile software package used for managing and visualizing data, initializing and recording from hardware and relaying information between clients and servers. NSTX has made good use of MDSplus during its years of operation. Initially it was used to control already aged Tokamak Fusion Test Reactor (TFTR) era Computer Automated Measurement and Control (CAMAC) hardware and act as a database for recording raw and analysis data sets. Simple UNIX user/group based access control lists (ACLs) were incorporated in its design, allowing users to be mapped to local or domain accounts or even to the privilege-less UNIX nobody account as a catch all for unmapped users. Through logs and ACLs it was possible to prevent unauthorized users from changing data or the physical structure of shot trees. Remote access to hardware, such as CAMAC based equipment could also be provided and controlled in a similar manner. Through logging and monitoring it was possible to have a reasonable degree of attribution if undesired changes were made. These levels of security were adequate for systems that could do no harm to the machine itself, short of losing or corrupting post shot data or configuration nodes.

Over the years a MDSplus has proven to be a reliable platform for NSTX with familiarity and confidence developing in both the physicists and engineers who use it. For these reasons it was determined that NSTX-U would continue to use MDSplus in a similar manner.

Despite these positives, MDSplus does not provide a rich platform of tools to ensure that data is not changed or corrupted by users either maliciously or accidentally. Under the hood the main protection method is to set nodes containing data that should be protected (such as raw digitizer data) to only be writable once via the "write_once" flag. This can easily be defeated with a few minutes of perusing the (open) source code or reading the MDSplus web site's documentation. While UNIX file system permissions also provide a modicum of protection, users with super user privileges on the server or other machines where they can switch accounts to one that has correct privileges to change the protection flag. This is not to imply that users or administrators are attempting to behave maliciously, sometimes these sorts of activities are seen as a method to accomplish a needed task.

MDSplus also does not provide a good mechanism for controlling who can create tree and when. It is possible for a user in the correct group to recreate a "pulse" which effectively erases the previous data contained within, with no simple way to tell exactly when and how it happened. A few potential solutions exist such as making copies of raw data trees immediately after they are populated or making nightly backups to tape. Relying on nightly backups is fine for somewhat minor risks such as losing experimental data or configuration settings for acquisition or timing hardware, but for critical machine protection systems such as the Digital Coil Protection System (DCPS) this is an unacceptable risk. The ramifications include physical damage due to missing or corrupted DCPS parameter data which could prove fatal to NSTX-U.

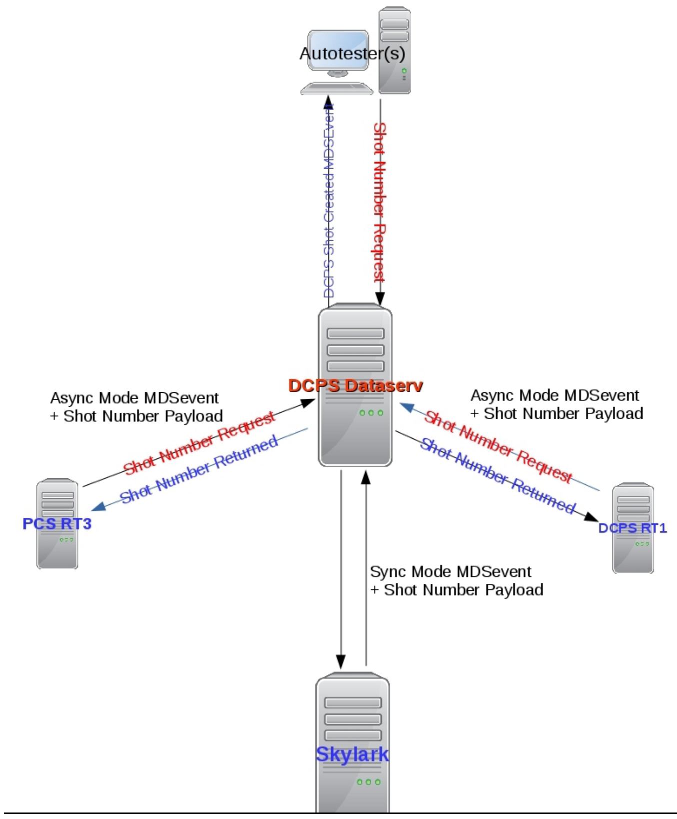
Potentially less catastrophic risks such as incorrect signal formats or parameters for initializing acquisition hardware are accepted and can eventually be found. These risks have a low impact on the performance or survival of the machine though they may impact the careers of the physicists and engineers responsible for them. A machine protection system such as DCPS requires parameter data perfection. A single incorrect decimal place could mean the difference between catastrophic failure or an amazing discovery. A method needed to be developed to prevent even authorized users from changing these important parameters once they had been tested and verified by the engineers responsible for generating and approving them for use.

II. Concurrent Resrouce MAnagement

Concurrent resource management in MDSplus is handled by an application level file locking mechanism. For tasks such as allowing multiple users to read and write to the same tree at the same time it is an adequate method. Immediately after completing an NSTX-U pulse, data is written to the shot trees by processes originating both inside and outside of MDSplus. An example of an internal MDSplus method would be a long running mdsipd server that receives incoming data from CAMAC serial highways which are driven by deeply integrated software drivers. An external example would be a computer or standalone digitizer connecting through Ethernet on a dynamically allocated port. External sources typically attach to a port on the MDSplus data server which initiates an xinetd script to open a transient mdsip process. Through these methods many different data generating sources can write to MDSplus trees concurrently though not necessarily atomically.

A. *Atomic Tree Creation*

An atomic method to provide new pulses to clients using a passkey protected handshake was developed where the client sets and MDSplus event that, with the correct data payload, initiates a tree creation process. This process or daemon uses both its own internal record keeping to keep track of shot (pulse) numbers. Once created another MDSplus event is set with the shot number as the payload. The event name contains a unique identifier that the client must be aware of so that two clients waiting for a new shot don't get the same number. This prevents clients from overwriting each other's data in a situation where they both request a new shot and wait for the same event.



This method was first put to use for the NSTX engineering test tree (eng_test) and was ultimately found to be an ideal method to produce new parameter data trees for DCPS. As a result all of the asynchronous shot trees are created through this process. Additionally, synchronous tree creation functionality must be provided and take place during NSTX-U plasma and test shots so a secondary numbering system was developed as part of the atomic daemon design. Normally the MDSplus shot number is tracked by a file included in the tree path called "shotid.sys". This binary file is updated every time a shot is created and subsequently set to be the "current shot" in MDSplus parlance. Since shots are typically created and numbered serially this shot or pulse number usually represents the most recent one. Afterwards methods like "getCurrent" open the tree and parse the contents of that file and return the value. When two different pieces of software, in this case the NSTX-U shot cycle software and the atomic daemon, create pulses of the same tree in two different number ranges the potential for a conflict exists. A example of an undesirable outcome would be NSTX-U shot data being written to an asynchronously created shot tree created for testing or development purposes.

Data access can also be affected in that clients attempting to get the most recent shot data, depending on when they ask, could get the most recent shot from either range. To prevent such conflicts a secondary index was developed and maintained by the atomic shot daemon software. This index is written to a node in the top level of the model tree so that users can query the tree using a different method to determine the most current shot in the atomic series. Additionally, the shot number is returned as a data payload from the atomic daemon at creation time so that clients are immediately informed that their request was fulfilled and a new shot has been created.

III. Data Security

Access to the DCPS trees varies by each individual tree and its purpose. For example, general read access is allowed for the parameter data tree, though, direct access to the server is restricted to DCPS administrators. Access to the DCPS parameter tree is provided through skylark, our main MDSplus data server. A user can connect to skylark and read data from the tree that exists on the DCPS dataserver. All external connections are mapped to the UNIX nobody account for connections originating from non-patch (crossover) cabled Ethernet adapters. The "nobody" account belongs to no UNIX groups, cannot log in to a terminal and does not possess the privileges to change tree files. The tree files themselves must be maintained as world readable for the data to be made available.

All DCPS model trees are created via Bash script that generates TCL or Tree Command Language code that is subsequently executed by the mdstcl command interpreter. TCL is specific to MDSplus and is not the more common language tcl or "Tickle". Each tree requires over thirty-thousand lines of generated TCL code to successfully be

*Work supported by U.S.DOE Contract No. ***DE-AC02-09CH11466***ext box

created. Many layers of abstraction exist to allow only a few variables to correctly generate the trees and allow for rapid modifications such as additional algorithm types or instances. One of the requirements of DCPS includes the ability to add or remove algorithms without the need to recompile the core software. This design upholds that ideal.

A form of reflection is used to turn configuration strings into variables that are used in generating the trees. A flag can be set to execute the TCL code without committing the changes so that errors can be detected before modifications to the tree are performed. Each time these changes are made the tree structure is regenerated and then unique algorithm identification data or ID's are loaded from a separate routine. Each ID correctly fingers a specific algorithm type and instance and allows the DCPS core software to correctly map them and instantiate them. They are also used to load the parameter data from files for evaluation and potentially make them available for protective use. This validation step will be discussed further in section IV.

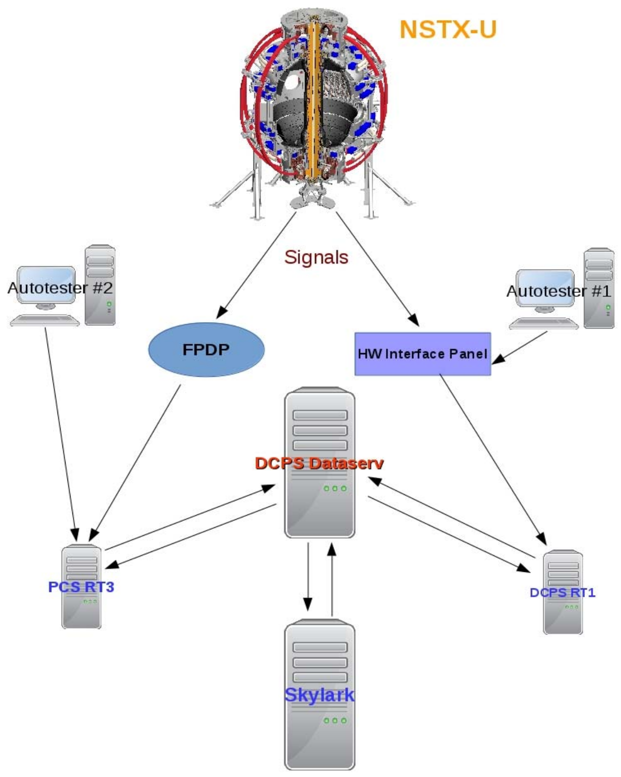
A. Access Controls

Other DCPS trees such as the autotester, development and operations trees are hosted on skylark along with the rest of the NSTX-U trees. They exist under their own DCPS subtree and are subtrees themselves. Certain trees that are used by both the FCC (Fusion Computing Center) and D-site Power Conversion Junction Area copies of DCPS have additional subtrees for each copy to archive data to. Trees can only be edited structurally from skylark with the exception of the parameter data tree which can only be edited on the DCPS dataserver. Access to the DCPS dataserver is tightly controlled.

Archival data is protected by use of the MDSplus "write once" flag. Once data is written to a particular node the node's data can no longer be changed, without the effort discussed earlier. Additionally, the date, time and user account that wrote the data is logged and stored as part of the node's meta-data. While this protection can be defeated, the restriction to editing the tree versus writing data to it prevents all but administrators from changing this attribute and thus the data once written. This is not the case for parameter data used to protect the machine, which cannot be manipulated.

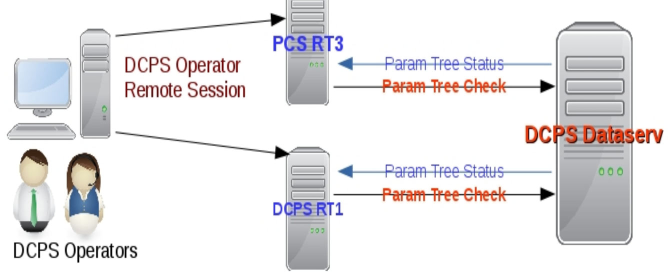
II. Tree Validation

The integrity of the DCPS parameter data tree is of the utmost importance to NSTX-U. The values contained in each parameter data set is carefully crafted by engineers and physicists familiar with the operational envelope [7] of the device. Physical forces exerted on the structure of the machine and its coils are calculated in real time and if one of these forces exceeds the machines ability to withstand them the DCPS faults and relays this information to protection hardware that will interrupt the pulse [6]. Given the importance of this function the values used to configure DCPS and prevent damage to the machine must be both written to and read from the tree without any change in precision. Additionally, the values should not change once they are validated and tested with the system. An Pre-operational Test Procedures or PTPs was created for this purpose. The PTP exercised a number of different code paths and fault limit values, all of which must be correct for the machine to operate safely insid



e the predefined envelope [2].

Initially the design of DCPS called for the verification of tree integrity to be handled by the DCPS core software itself. While MDSplus does include data validation functionality in its design, the importance of this particular dataset necessitated further protection mechanisms. During the design phase it was proposed that the entirety of the parameter tree data could be assembled into a large structure and check-summed inside of the core. It quickly became apparent that an external method would be not only easier to implement but independent of the DCPS software itself, which is desirable for a number of obvious reasons. MDSplus trees are comprised of three files, a datafile, a tree file and a characteristics file. These files can be verified independently and the value of their SHA256 stored in a read only file, in the same directory as the MDSplus troika. Methods to create these files and parse them were developed



and incorporated in the software used to start the DCPS core. Upon startup DCPS calls out to the dataserver for verification that the parameter data shot being used has been checksummed and that the file checksums of the three tree files have not changed. If a change is detected DCPS will not run.

IV. Conclusion

MDSplus continues to be an important part of NSTX-U and how it operates. In addition to its previous functions as a database, CAMAC controller and visualization suite it now is a major component of the real-time coil protection system used to protect the machine. While not all of the desired functionality existed to securely use MDSplus as a tool to accomplish this difficult task, the flexibility of the system allowed methods to be developed that ultimately provided them.

Acknowledgment

Charles Neumeyer created the top level system requirements document for the overall NSTX-U Digital Coil Protection System [8]. Ronald Hatcher created the software requirements document from which this design is derived.

References

[1] Ono, M., Kaye, S., Peng, Y., Barnes, G., Blanchard, W., Carter, M., et al. (2000). Exploration of spherical torus physics in the NSTX device. Nuclear Fusion, 40(3Y), 557. Menard, J., & Neumeyer, C. (2009). NSTX Upgrade Scientific Motivation and Project Requirements. *318*, 15-16.

[2] Gerhardt, S. P., Andre, R., & Menard, J. E. (2012). Exploration of the equilibrium operating space for NSTX-Upgrade. *Nuclear Fusion*, *52*(8), 083020.

[3] Neumeyer, C., Avasarala, S., Chrzanowski, J., Dudek, L., Fan, H., Hatcher, R., ... & Zhan, H. (2009, June). National Spherical Torus Experiment (NSTX) Center Stack Upgrade. In *Fusion Engineering, 2009. SOFE 2009. 23rd IEEE/NPSS Symposium on* (pp. 1-4). IEEE.

[4] Menard, J. E., Gerhardt, S., Bell, M., Bialek, J., Brooks, A., Canik, J., ... & Zolfaghari, A. (2012). Overview of the physics and engineering design of NSTX upgrade. *Nuclear Fusion*, *52*(8), 083015.

[5] Menard, J., Menard, J., Canik, J., Covele, B., Kaye, S., Kessel, C., et al. (2010). Physics design of the NSTX-U. *27th EPS Conf. on Plasma Physics P*.

[6] Woolley, R., Titus, P., Neumeyer, C., & Hatcher, R. (2011). Digital Coil Protection System (DCPS) algorithms for the NSTX centerstack upgrade. *Fusion Engineering (SOFE), 2011 IEEE/NPSS 24th Symposium on*. IEEE.

[7] Titus, P. H., Woolley, R., & Hatcher, R. (2011). Stress multipliers for the NSTX upgrade digital coil protection system. *Fusion Engineering (SOFE), 2011 IEEE/NPSS 24th Symposium on*. IEEE.

[8] Neumeyer, "Coil Protection System Requirements Document," NSTX_CSU-RQMT-CPS-159, unpublished.