INL/EXT-21-00000
Revision 001

Cyber Attack and Defense Use Cases for Autonomous and Remote Operations for Advanced Reactors

August 2021

CT-21IN110414

Idaho National Laboratory
Chris Spirito

Georgia Institute of Technology
Dr. Fan Zhang

University of Massachusetts Lowell
Dr. Sukesh Aghara
Collin Duffley
Joel Strandburg

University of Tennessee Knoxville
Dr. Jamie Coble

INL is a U.S. Department of Energy National Laboratory
operated by Batelle Energy Alliance, LLC

DISCLAIMER

This information was prepared as an account of work sponsored by an agency of the U.S. Government. Neither the U.S. Government nor any agency thereof, nor any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness, of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the U.S. Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the U.S. Government or any agency thereof.

INL/
Revision 001

Cyber Attack and Defense Use Cases for Autonomous and Remote Operations for Advanced Reactors

Error! Reference source not found.**T-21IN1104014**

Idaho National Laboratory
Chris Spirito
Georgia Institute of Technology
Dr. Fan Zhang
University **of Massachusetts Lowell**
Dr. Aghara, Collin Duffley, Joel Strandburg

University of Tennessee Knoxville
Dr. Jamie Coble, Dr. Fan Zhang

August 2021

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Office of Nuclear Engineering
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

Page intentionally left blank

SUMMARY

The next generation of *Advanced Reactors* include planned capabilities for both Autonomous (operating without human interaction for a set period-of-time) and Remote (operating with human interaction from a separate physical location) Operations. Existing Nuclear Reactor architectures include a set of safety and security constraints tightly coupled with personnel policies and procedures. As *Advanced Reactors* are fielded with these new *Autonomous* and *Remote* operational capabilities, the architectures and associated infrastructure services and components will perceivably expand the overall attack surface and risk calculations with regards to safe and secure operations. This paper is part of an FY21 work program focused on ensuring *Advanced Reactor* designs are informed with threat-based guidance on design and operation of Secure Architectures with a specific focus on the deployment of *Autonomous Systems in support of Advanced Reactor Operations.* This Use Case catalog complements the methodology for assessment of the cyber threat against these architectures released in March of 2021.

Page intentionally left blank

CONTENTS

FIGURES

TABLES

Page intentionally left blank

ACRONYMS

ARAMS	Advanced Reactor Autonomous Management System
AS	Autonomous System
CISO	Chief Information Security Officer
CSIRT	Cyber Security Incident Response Team
DT	Digital Twin
ESFAS	Engineering Safety Features and Auxiliary Systems
EXFIL	Exfiltration
FPGA	Field Programmable Gate Array
ICD	Interface Control Document
ICSs	Industrial Control System
ICT	Information and Communication Technology
ML	Machine Learning
NRC	Nuclear Regulatory Commission
NPP	Nuclear Power Plants
O&M	Operations and Maintenance
OPSEC	Operational Security
OSINT	Open-Source Intelligence
PHM	Plant Health Monitoring
POAMS	Plant Operations Autonomous Management System
RPS	Reactor Protection System
SDM	Shutdown Margin
SME	Subject Matter Expert
SSD	Solid State Drive
TTP	Tactics, Techniques, and Procedures
TWh	Terawatt Hour

Page intentionally left blank

Cyber Attack and Defense Use Cases for Autonomous and Remote Operations for Advanced Reactors

1. Background

In March of 2021 we published a Cyber-Threat Assessment Methodology for Autonomous and Remote Operations for Advanced Reactors. This paper provided readers with a 5-step approach for conducting this type of assessment. This paper includes Use Cases derived from the application of this methodology to Autonomous and Remote Operations and their underlying systems and functions.

1.1 Cyber-Threat Assessment Methodology Overview

This full assessment methodology is provided in the publication referenced in the *Background* section above. An overview of the assessment methodology is included for the reader who has not read through the full methodology but would like to understand the Use Cases and how to use them. As we stated in the methodology paper, we recommend cycling through the methodology steps and cyber threat assessment phases as many times as required to gain a sufficient understanding of the autonomous system attack surface and the commensurate adversarial capabilities that are considered plausible in subverting the defined system functions, processes, and components. With each finding is an opportunity to perform a consequence-based analysis and suggest countermeasures for mitigating the identified risks.

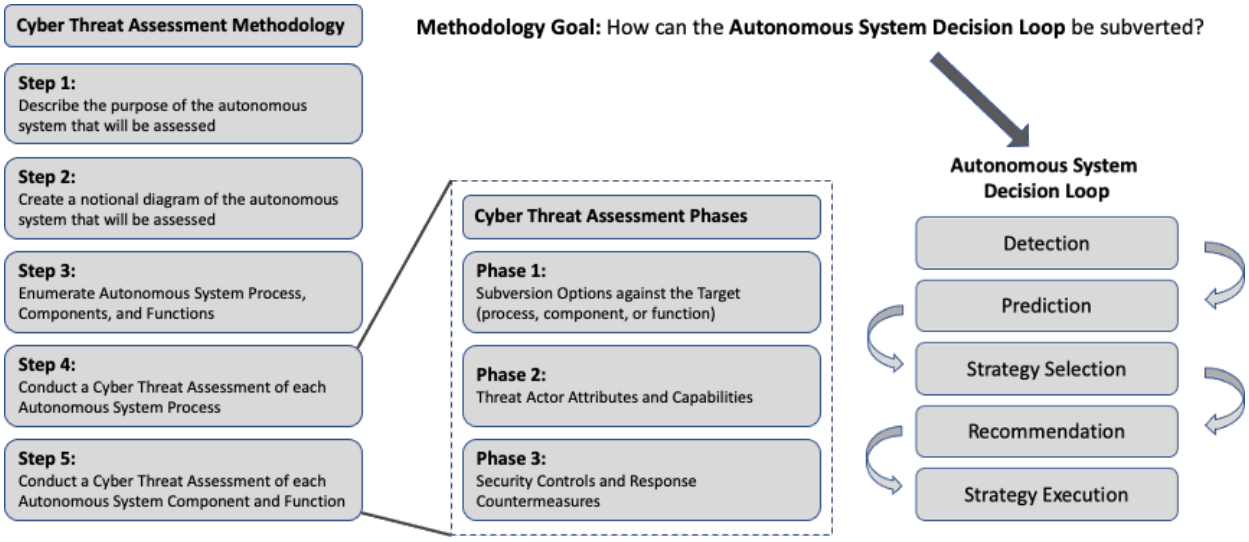


Figure 1. Cyber Threat Assessment Methodology Flow

1.2 Use Case Scope and Usage

The usefulness of any methodology is derived by its adoption and subsequent lessons learned as it is applied to operational environments. For this collection we include four use cases that represent: system level analysis of a set of Digital Twins (the example included within the Methodology); component level analysis of a distributed plant sensor system (DPSS) and distributed plant sensor monitoring system (DPSMS); subversion of a Digital Twin by attacking the dependent machine learning algorithms and implementation; and a broad-scale attack against the machine learning infrastructure of an Advanced Reactor vendor.

2. Plant Operations Autonomous Management System (POAMS) Threat Assessment

This Use Case is sourced from the Methodology and is focused on a fictionally derived system responsible for managing Reactor Objectives and managing the suite of generation and transmission assets at a remote location.

2.1 Description of the Assessment Target

This assessment will be performed on a Digital Twin (DT) responsible for diagnosis and strategy assessment of a Gen IV Reactor. The DT receives data from the reactor and plant subsystems and uses a machine learning classifier to implement a detection function. The DT uses algorithms (magically implemented for now) to process the classified protection data to implement a prediction function. The DT uses another set of algorithms to combine the prediction data with reactor and plant goals to implement a strategy selection function. The DT uses a fourth set of algorithms analyzing strategy options and implements a recommendation function. The recommendation function interfaces with plant management systems to execute the recommended strategy. If a cyber threat actor compromised data prior to its arrival at the detection function, the DT may recommend a strategy for execution that at best undermines the plant goals and at worse trigger a safety event. If a cyber threat actor compromised the algorithms used for strategy selection of recommendation optimization, the DT may induce a plant-wide lack-of-confidence event as the time to deconflict compromised algorithms in operational systems is significant.

2.2 Notional Diagram

This is a notional architecture for an **Advanced Reactor Autonomous Management System (ARAMS)** responsible for managing Reactor Objectives within the Remote Plant and managing Power Generation and Transmission Objectives for the geographic region. The architecture includes an Advanced Reactor with power generating components, an autonomous system (upper right) responsible for reactor and plant subsystem autonomous functions; and an autonomous system (lower right) responsible for power generation and transmission. The reactor and plant subsystem autonomous functions manage the

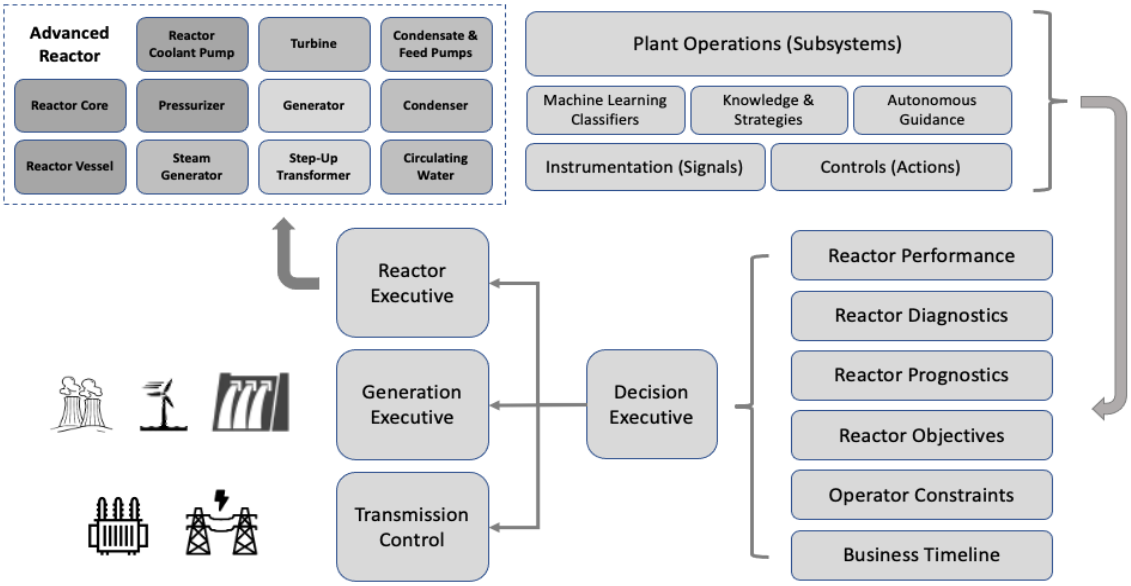
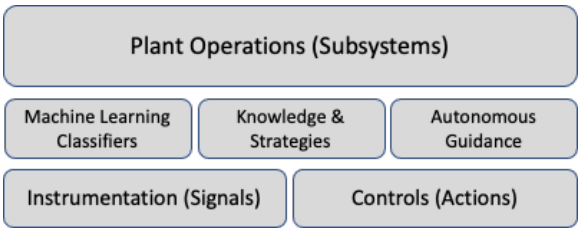


Figure 2: Notional Diagram for Advanced Reactor Autonomous Management System

instrumentation (signals) and controls (actions) of the reactor and reactor subsystems through the use of machine learning classifiers that when combined with knowledge and strategies produce guidance and recommendations for the safe operation of the power reactor.

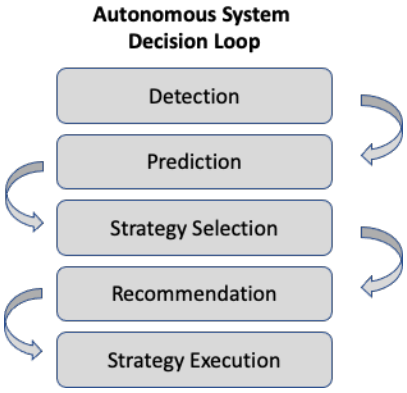
The power generation and transmission autonomous system implements a traditional autonomous system management cycle to include ingesting reactor performance and diagnostic information into a machine learning classifier that computes reactor prognostics. The reactor prognostics combined with generation and transmission objectives and bounded by operator constraints (if any) inform the Decision Executive that is able to execute functions via the Reactor and Generation Executives. If a cyber threat actor conducted an attack against the Decision Executive function, generation resources could be constrained when demand thresholds have requested additional electricity load.

2.3 Enumeration of Processes, Components and Functions



The POAMS is deployed as three components that replace human operators, responsible for autonomously managing the nuclear reactor and plant subsystems. The three components take inputs from the Instrumentation sensors and execute actions via the control actuators. POAMS is delivered with a manifest that includes technical details on how the autonomous system is architected with full visibility into the design and development processes enabling the cyber threat assessment team to work through any subset of components they are interested in. The next section includes an edge case where the autonomous system is produced by a vendor and delivered as a black box.

POAMS Process Implementation



POAMS utilizes a traditional autonomous system decision loop implemented across three components. This decision loop includes algorithms that perform detection of events (including anomalies) using machine learning classifiers; prediction of future operational states; strategy selection based upon analysis of future operational states; generation of recommendations on how to implement each strategy; and execution of the strategy chosen by the autonomous system. Annex II contains a detailed description of this process that is the main focal point of cyber threat assessments of autonomous systems.

The assessment team may choose to merge this assessment methodology with their reactor and plant subsystems cyber threat assessment. The assessment team may also choose to merge this methodology with existing safety assessments incorporating the cyber threat elements. The boundary for this specific methodology is on the autonomous system processes, components, and functions and cyber threats related to designing, building, and deploying these capabilities in Advanced Reactors.

POAMS Component and Function Enumeration

Within each POAMS component and function description, targets are denoted with the **(target)** attribute. This approach allows the assessment team to create a list of targets for cyber threat analysis.

The following descriptions contain terminology specific to the components and functions being described. The reader is encouraged to search on YouTube for tutorial on these topics.

Instrumentation (Signals): This component is responsible for ingesting data from the monitoring systems and sensors. The instrumentation data (target) is written to a database (target) that is queried by a process (target) controlled by the Machine Learning (ML) classifiers (target).

Machine Learning Classifiers: The ML Classifiers implement algorithms (target) that were developed by a team of Nuclear Engineering and ML experts (targets) at the Acme Corporation (target). The ML algorithms implement Supervised Learning (target), Reinforcement Learning (target), and Deep Learning (target), and were developed using the TensorFlow (target) ecosystem in Python[a] (target) and Keras[b] (target).

Knowledge & Strategies: A second suite of Python-implemented algorithms (target) provide Prediction, Strategy Selection, and Recommendations based upon a knowledgebase (target) and updatable strategy generation algorithm suite (target) provided by the Acme Corporation quarterly (distribution target).

Autonomous Guidance: A third suite of Python-implemented algorithms (target) analyzes and optimizes the recommended strategies and executes the strategy by sending commands to reactor and plant subsystem management systems via interfaces (target) defined by the autonomous and control system interface specifications (target).

Controls (Actions): This component is responsible for the command interfaces (target – protocol implementation) with control systems (target – out of scope) and their corresponding actuators (target – out of scope).

It is the interaction between *Machine Learning Classifiers, Knowledge & Strategy Implementation, and Autonomous Guidance* that are priorities for assessment. These areas contain algorithms that are often implemented without best practice guidance and with unknown vulnerability exposures.

Aggregated Target List

The assessment team can now produce an initial target list from the process, component and function enumeration tasks.

Processes (procedural)
Autonomous Systems Decision Loop
Autonomous Systems Design Cycle
Quarterly Strategy Update

Organizations
Acme Corporation

People
Acme Corporation Experts
Advanced Reactor Design Team

Algorithms
Machine Learning Classifiers
-Supervised Learning Algorithms
-Reinforcement Learning Algorithms
-Deep Learning Algorithms
Development Tools
-TensorFlow
-Keras
-Python
Strategy Generation Suite

Data Targets
Instrumentation Data
Instrumentation Database
Strategy Knowledgebase

Interfaces
*Instrumentation Data ICD**
Control Systems ICD

* ICD: Interface Control Document

[a] Python. https://www.python.org/

[b] Keras. https://keras.io/

2.4 Cyber Threat Assessment of Autonomous System Processes

Phase 1: Subversion Options against the Target (process, component, or function)

Suppose that the analyzed attacker has an identified objective to prevent reliable energy production. Distortion and/or disruption of data flows from the Rod Control and Indication System to the central autonomous system could use that erroneous data to facilitate this objective and shutdown the reactor. Threat analysts would analyze the vulnerabilities of the Rod Control and Indication System as well as all direct and indirect supporting systems.

1. The goal of the assessment is to determine:
2. Whether this type of attack scenario (data distortion and disruption) is feasible.
3. Identify digital dependencies of the Rod Control & Indication System and how these systems interact with autonomous control system.
4. Determine process level vulnerabilities in the system the attacker could leverage.
5. Demonstrated techniques and tools that could facilitate exploitation to create the desired effect.

This includes identification of published vulnerabilities in autonomous system endpoints and subcomponents, vulnerabilities in the interconnected networks, how trust relationships are implemented for data flows, and how the data is generated, transmitted, processed, and stored. Central to this analysis is understanding whether accepted network rules and processes represent exploitable vulnerabilities that could cause the distortion and/or disruption of data to the Rod Control & Indication system.

Phase 2: Threat Actor Attributes and Capabilities

The presence of vulnerabilities in the system that could potentially lead to data distortion and/or disruption from the Rod Control & Indication System and trip the reactor under false pretenses is important information. However, context provides assistance in prioritizing action for programmatic mitigation plans. Does the adversary have the necessary capabilities to leverage these vulnerabilities can provide context for prioritization? Specifically, threat assessment should determine if the attacker has the necessary Tactics, Techniques, and (demonstrated) procedures (TTPs) to:

1. Identify discovered vulnerabilities.
2. Weaponizing the vulnerability information into an executable attack plan.
3. Build/Acquire the necessary resources to execute the attack plan.
4. Establish initial access into the system and gain access to the Rod Control & Indication System.
5. Control the data generated and transmitted by the Rod Control & Indication System.

Identifying these 5 threat actor attributes and capabilities determines the exposure of the system to attack and assists in optimal placement of cyber detection and protection capabilities.

Phase 3: Security Controls and Response Countermeasures

Every threat actor TTP identified during Phase 2 generates a signature that informs protection, detection, and response procedures. Advanced reactor autonomous systems will require highly controlled and monitored networks, requiring defensible network security and endpoint protection, and a methodical threat analysis to evaluate threat actor capabilities to evade defensive measures and avoid detection while

executing the attack plan. While the autonomous network might have continuous protection and detection parameters surrounding the endpoints of the Rod Control and Indication System, the attacker must demonstrate techniques to:

1. Evade defensive security controls and countermeasures by uninstalling or disabling security software and obfuscating and/or encrypting data and scripts.
2. Exploit trust relationships to avoid detection.
3. Escalate privileges in the target environment.

Step 3 expands the threat actor capability analysis by:

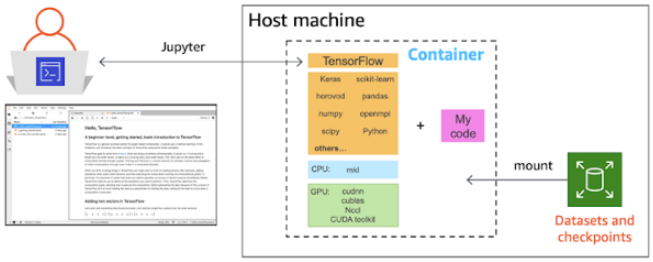
1. Identifying Control Rod and Indication System security measures.
2. Enumerating threat actor capabilities to evade these security measures through OSINT analysis.

Understanding this attack tactic is critical to complete Step 5 which transitions from understanding how an adversary can compromise a critical process to focusing on compromises of specific systems and components.

2.5 Cyber Threat Assessment of Autonomous System Components and Functions

The threat assessment team should conduct a cyber threat assessment against each autonomous system component and function using the three-phase assessment approach. Prioritize the assessment targets based upon perceived targetability and associated consequence of subversion. Recognize that there will be inter-component and inter-function dependencies that may benefit from an expanded assessment boundary that is left up to the discretion of the assessment team. Applying the Cyber Threat Assessment Process to POAMS is straight-forward.

POAMS algorithms are created within a TensorFlow environment at the Acme Corporation. The ML Laboratory Technical Manager provided this notional diagram[c] of their development environment. The POAMS ML subject matter experts (SME) work in Jupyter Notebooks[d] that to execute their TensorFlow algorithms within a docker container that is provided access to a database of training data seeded by the Advanced Reactor Nuclear Engineering SMEs. The ML SME develops and tests their algorithms and once they are complete requests a code and function review from another team member before exercising the two-person commit rule for submitting code to the production code repository.



Phase 1: Subversion Options against the Target (process, component, or function)

How can this target be subverted through the use of cyber capabilities? A cyber threat actor has the following subversion options against this target: TensorFlow (development environment and library

[c] Why use Docker containers for machine learning development?
https://aws.amazon.com/blogs/opensource/why-use-docker-containers-for-machine-learning-development/

[d] Jupyter. https://jupyter.org/

dependencies); Docker (containers that execute the code); Jupyter Notebooks (development interface with TensorFlow); Training Data (via an attack on the database or the data access functions); Dependency libraries; Hardware (CPU/GPU); ML SMEs (social engineering); Advanced Reactor Nuclear Engineering SMEs (social engineering). OSINT research identifies seven Tensorflow two Jupyter Notebook vulnerabilities published in 2018 and 2019. These include:

- CVE-2019-9635, CVE-2018-7574/6: Null Pointer Dereference, Denial of Service, Context Dependent
- CVE-2018-10055: Invalid Memory Access and Heap Buffer Overflow, Crash
- CVE-2018-8825/7575: Buffer Overflow with Arbitrary Code Execution, Exec Code
- CVE-2018-8768: For < v5.4.1, a maliciously forged notebook can bypass sanitization to execute code
- CVE-2018-7577: Memcpy parameter overlap in Google Snappy library
- CVE-2018-7206: JupyterHub OAuthenticator vulnerability with GitLab (not applicable)

Each of these vulnerabilities has been remediated but an attack vector (bolded) that was once successful may be a future opportunity for subversion of this autonomous system component. A cyber threat actor may also engage the research community, similar to what Southern Eagle did, and papers published in the past two years provide insight into additional attack vectors and pathways:

- Anthi, Eirini, et al. "Adversarial attacks on machine learning cybersecurity defences in Industrial Control Systems." Journal of Information Security and Applications 58 (2021): 102717.
- Trott, David. "Deceiving Machines: Sabotaging Machine Learning." CHANCE 33.2 (2020): 20-24.
- Rosenberg, Ihai, et al. "Adversarial Learning in the Cyber Security Domain." arXiv preprint arXiv:2007.02407 (2020).
- de Mello, Flávio Luis. "A survey on machine learning adversarial attacks." Journal of Information Security and Cryptography (Enigma) 7.1 (2020): 1-7.
- Ayub, Md Ahsan, et al. "Model Evasion Attack on Intrusion Detection Systems using Adversarial Machine Learning." 2020 54th Annual Conference on Information Sciences and Systems (CISS). IEEE, 2020.

These are novel approaches to subverting machine learning published in research journals that the threat assessment team should analyze for applicability to the POAMS ML Classification implementation.

Phase 2: Threat Actor Attributes and Capabilities

As the cyber threat assessment team works their way through each target, a taxonomy of attributes and capabilities will form that will serve as a source for subsequent assessments. When iterating through the process take some time to evaluate the state of threat actor attributes and capabilities as these will shift over time based upon their experiences and capability development achievements and failures.

The subversion options in Phase 1 are focused on the TensorFlow environment, the suite of training and classification algorithms, and the dataset interfaces. The threat actors capable of conducting attacks against these targets will require at least two months of resource ($$) to conduct the reconnaissance, weaponization, and delivery of the attack payloads. They will need to be familiar with private corporations that employ data scientists and have a support network for specialty areas they are not proficient in such as design and implementation of Advanced Reactor autonomous systems and vulnerability analysis of ML algorithms.

If the Acme Corporation Staff are part of the subversion target set defined by the cyber threat assessment team, the threat actor will require a social engineering capability along with either insider access to assess Operational Security (OPSEC) posture or an EXFIL capability to extract information on their OPSEC policies and procedures. Threat actors who are proficient with social engineering and influence

operations require thoughtful security controls and response countermeasures to include awareness training for all staff with privileged access to the target environment.

Phase 3: Security Controls and Response Countermeasures

Similar to the taxonomy of threat actor attributes and capabilities, the threat assessment team should maintain a taxonomy of security controls and response countermeasures. This will both ease the analysis process and allow for defensive capability development roadmaps to be influenced by the autonomous system cyber threat analysis.

Security controls should be identified for restricting access to the physical and virtual algorithm development environments as well as the training data. Security controls should be implemented that provide a non-repudiated audit of algorithm code commit events and all distribution stream events as the algorithms are sent to target systems, such as the Advanced Reactor autonomous system. Development and production environment module integrity checks should be operational. The incident response process should be reviewed such that it includes digital (cyber) events, along with Physical intrusion events, and any reported OPSEC events related to social engineering or perceived external manipulation of algorithms.

While the process for conducting cyber threat assessments is straight-forward, the assessment team must be cautious at first in bounding their target space and carefully documenting their process and findings to be incorporated into future assessments.

3. Military Base SMR Distributed Sensor System Threat Assessment

This Use Case provides an assessment of a *Distributed Sensor System* that is deployed at a Military Base Small Modular Reactor (SMR). This is an example of a traditional Use Case where each methodology step is completed to inform the risk evaluation of this component level system that supports SMR operations.

3.1 Description of the Assessment Target

The target site is a highly automated small modular reactor facility powering a military base. The reactor is assumed to have automated control and maintenance routines such that it can be normally operated with just three or more operators residing in a central control room, with short periodic maintenance requiring up to five personnel occurring on pre-planned schedules. The target is a distributed plant monitoring system, composed of a distributed plant sensor system (DPSS) and distributed plant sensor monitoring system (DPSMS). This distributed monitoring system works effectively as a theoretical "skin" around and inside the reactor and is composed of a large variety of pressure sensors, temperature sensors, vibration sensors, visual sensors, acoustic sensors, and other sensors distributed at all levels of the facility. As there is a large quantity of data produced by each sensor, there are multiple data aggregation nodes composed of digital microcontrollers which adaptively down-sample and combine this data before feeding it to the DPSMS located within the control room. The end goal of this system is to serve as a persistent, mesh-based monitoring network which allows real-time analysis of events within the facility independent of other control or monitoring systems already in-place. This allows the control room operators (who are normally the only personnel within the facility) to have "eyes and ears" at all levels of the facility to ensure installed hardware is operating as expected and that there are no unauthorized personnel or events on-site. In addition to providing collated and polished versions of the raw data coming in from across the facility, the computer hosting the DPSMS in the control room also has a variety of short- and long-term trend analysis programs, along with a database of manufacturer-provided ideal behavior and actual facility behavior, to identify long-term anomalous trends and notify the operators of such.

3.2 Notional Diagram

The following notional diagram outlines the DPSS and DPSMS and relevant connections between these two systems. The DPSS is composed of a variety of sensor types mentioned above, with hardwired connections between these sensors and one of the multiple data aggregation nodes distributed across the facility. Correspondingly, these data aggregation nodes are hardwired to the DPSMS located within the control room. The DPSMS is composed of a combined computer and operator display system which receives nodes and displays input from the data aggregation nodes to the operators while simultaneously running the inputs through a series of pre-programmed and learned trend analysis programs which independently decide on providing alarm notifications to the operators. This input is then stored internal to the computer for future reference.

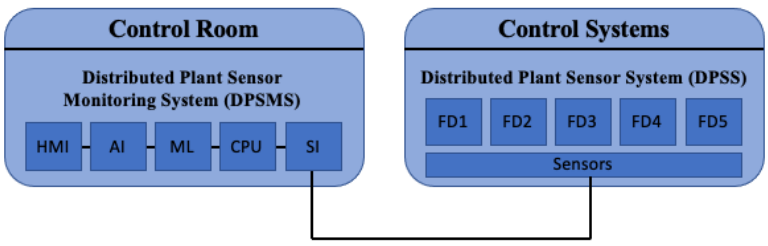


Figure 3: Notional Diagram DPSMS and DPSS

3.3 Enumeration of Processes, Components and Functions

The DPSS is composed of multiple digital and analog sensors, each of which is assumed to be unable of independent physical computation. The raw data from these sensors is fed directly to the data aggregation nodes, which are composed of single-board computers running simple data down-sampling and aggregation algorithms on top of a hardened Linux operating system. These nodes receive the data via data acquisition terminals (both analog and digital) incorporated into the node packaging. Other than the incoming data feeds and the outgoing connection to the DPSMS, the external ports and access points (including wireless communication capabilities) of the nodes are assumed disabled or removed.

The DPSMS is composed of a single engineering workstation and attached monitor. The software is composed of a scheduler to receive input and coordinate the launching of visualization/analysis programs, the analysis programs themselves, and drivers to render information to the monitor. The hardware in the workstation is composed of a multi-core processor, computer memory and data storage disk, and physical connections to the monitor. Similar to the data aggregation nodes, other than the incoming data feeds and outgoing connection to the monitor, the external ports and access points (including wireless communication capabilities) of the engineering workstation are assumed disabled or removed.

3.4 Cyber Threat Assessment

Phase 1: Subversion Options against the Target (process, component, or function)

Opportunities for direct subversion of facility operations through attacks on the DPSS and DPSMS are minimal, as the system is strictly a remote diagnostic tool for use by the operators. However, it may still be desirable to attack this system due to rules and regulations in place due to the military nature of the facility. In particular, the operators are required to document in a logbook all operator- or system-flagged anomalies/occurrences detected via the DPSS/DPSMS, and if more than five per hour are detected (or if the operators deem an anomaly sufficiently problematic) they are required to place the facility into safe mode and call in an external security team to examine the detected anomalies. Upon examination and report by the inspection team, the operators must then consult external authorities to decide if the facility must shut down for more extensive inspection/maintenance or if the facility may resume operations. During this "safe mode" period the reactor is brought to a low-power state from which it is expected to be able to shut down rapidly, and the operators/facility are forbidden from responding to external power or load following requests. An attacker may wish to use these protocols to temporarily shut down or inhibit the base the reactor facility is powering momentarily, as backup generators may take time to start or the attack may be part of a larger-scale operation. Another source of disruption may be in the form of increased maintenance overhead, as the recorded logbooks and plant-wide sensor data may be used to make periodic assessments of facility health and maintenance needs. In this case the attacker's goal may be to establish a pattern of high facility maintenance needs and untrustworthiness, resulting in distrust of diagnostic subsystems, increased maintenance personnel presence (and therefore increased chances for the exploitation of social engineering vulnerabilities) or escalating economic and political costs such that maintaining the presence of a base in the region is unappealing.

As the DPSS and DPSMS are isolated systems which are effectively "air-gapped" from the outside world, it is highly unlikely that security vulnerabilities will be introduced via regularly scheduled updates or external firewall vulnerabilities. However, as the facility is military in nature and thus may be a higher-value target to groups of interest, attacks using internal facility communications channels (similar to Stuxnet) or novel attacks using supply chain vulnerabilities should be studied. Additionally, the use of

social engineering to either introduce attack vectors into the system, or promote hidden attack vectors, should be examined.

Attacks via tainted libraries and software may be readily used for targeting the DPSS/DPSMS. If an attacker is knowledgeable about either the firmware running on the data aggregation nodes or the high-level libraries running on the engineering workstation or is merely privy to a vulnerability in a widely used library, they may be able to package a payload composed of a variety of such vulnerabilities along with directives on next steps once a vulnerability is successfully used. The air-gapped nature of the DPSS/DPSMS does not disadvantage this class of attack, as either immediate or more long-term effects can be readily introduced, depending on the goal of the attacker, through the use of simple timed-release or periodic payload directives. For example, if an attack is focused on the DPSMS it may be feasible to implement a cryptographic subroutine as part of a ransomware attack, which would have immediate short-term effects; the cryptography only needs to be complex enough to cause an alarm. Another attack might be frequent stopping/restarting of the data aggregation nodes during detected reactor transition periods, corresponding to a theoretically faulty node failure.

Attacks via trusted communications channels may be possible if the DPSMS is compromised and if communication lines are sufficiently insecure between the DPSMS and DPSS. In this context, it is assumed that the data aggregation nodes will respond to certain incoming communication codes, either as part of standard communication protocols or as part of hidden/undocumented protocols introduced during node design. Using these vulnerabilities, it may be possible for attackers to enable the loading of malicious libraries into the node memory, enabling either the disabling of the node or more sophisticated attacks such as replay attacks. Vulnerabilities also exist from the other "direction", in that if the data aggregation nodes are compromised then they may be able to send malicious input or commands to the DPSMS. Either of these attack vectors may be used maliciously on their own, or they may be combined with the tainted software attack discussed above to provide a vehicle for introducing complex payloads from device to device.

Another direct and indirect attack vector is a supply chain attack, where if components/software are sourced from sources outside of trusted countries then this hardware/software may be compromised. For example, compromised hardware may be introduced which does not have a specific port/connection fully closed/disabled, allowing for later payload injection from attackers. Additionally, the software supply chain includes the potential for algorithm contractors to use insecure libraries which may allow for a variety of vulnerabilities. If the attacker is knowledgeable about these libraries or about the library design process, they may attempt to conceal a payload at some point in either the firmware delivery process of the data aggregation nodes or the delivery process of the higher-level libraries packaged on the engineering workstation. In either case, these supply-chain vulnerabilities may also be used maliciously on their own or may be used in combination with the communication channel vulnerabilities and the tainted software vulnerabilities discussed above to attack the facility.

Social engineering attack opportunities towards facility operators may be available, however as these operators are expected to be in the control room for a majority of the time, clearly visible to others, this may not be a serious concern. Two other serious sources of social engineering vulnerabilities may come from contractors and maintenance personnel working on-site during scheduled or emergency maintenance, or from contractors/employees developing the software with the data aggregation nodes/engineering workstation. Specifically, contractors and maintenance personnel would be expected to move about the facility and would thus likely not arouse suspicion when carrying technical components inside, allowing

for potential attacks to be introduced via either plugged-in media or perhaps components (for example, data aggregation nodes) being swapped out entirely with malicious counterparts. Software developers during the design and iteration phase may be prompted for information on specific libraries being used through forum posts, conferences or academic papers published, or may even be compelled to introduce any number of vulnerabilities either knowingly or unknowingly by attackers.

Phase 2: Threat Actor Attributes and Capabilities

Given the target's military nature, and the aforementioned nature of the system as an air-gapped and highly controlled set of equipment, the nature of an attacker should be assumed to be a nation-state equivalent. This entails a highly motivated team with a significant amount of financing and the ability to competently assess supply chain vulnerabilities in parts manufactured in difference locations. The development and usage of several zero-day vulnerabilities should be considered a possibility, with the additional possibility of deep supply chain attacks from firmware engineers (for example) or factory workers also considered. An attacker seeking to introduce this system should be considered highly skilled in the technical domains involved, as the electronics and sampling algorithms are mostly low-level and well-known and thus knowledge and theory on the discovery of vulnerabilities is generally well-disseminated and well-taught. Additionally, since the DPSS/DPSMS are going to be operating in real-time it's likely the programming will be accomplished in C/C++, likely prompting the attacker to consult experts on security vulnerabilities in these areas and allowing them to perform a detailed survey on vulnerabilities existing in these languages.

Phase 3: Security Controls and Response Countermeasures

The air-gapped nature of the system again removes a large class of vulnerabilities from consideration, however there are several areas of improvement possible. For example, it was noted that the connections between the data aggregation nodes in the DPSS and the engineering workstation in the DPSMS are not strictly one-way with physical restriction, and that two-way communication is possible. We recommend the use of physical barriers such as data diodes to prevent such two-way communication, eliminating the possibility of communication vulnerabilities entirely.

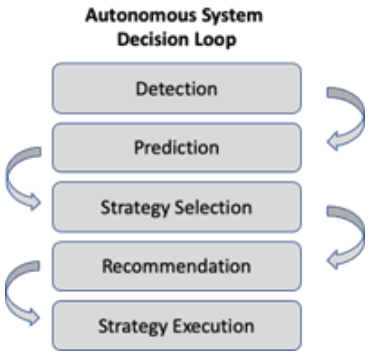
To avoid the possibility of data aggregation corruption, we recommend the deployment of redundant data aggregation nodes (suggested minimum of three) separately sourced to ensure firmware corruption in one controller cannot spread to others. We make the same recommendation for the engineering workstation in the DPSMS and suggest periodic comparisons/switching between the workstations to ensure system integrity. Barring this, we recommend a protocol of regular system wipes and reboots to disable the retention of long-term malicious software.

The inspection and maintenance protocols are satisfactory; however the possibility of social engineering being used to introduce ("swap out") functional equipment for malicious equivalents is not impossible, especially as the maintenance and inspection team may be composed of different members responsible for different jobs. Therefore, we recommend the use of a "two-man rule" where all maintenance and inspections are to be always performed by two personnel. This precludes a single person being successfully socially engineered into performing malicious activity.

4. Autonomous System Decision Loop Threat Assessment

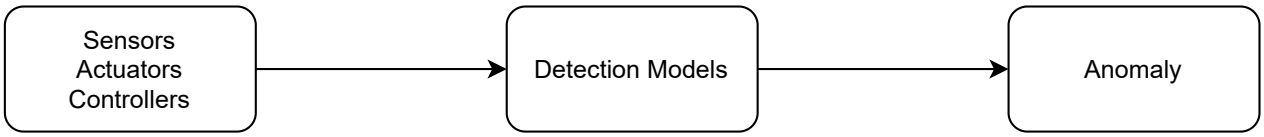
As shown in this Autonomous System Decision Loop, machine learning algorithms are implemented to perform detection, prediction, strategy selection, and recommendation. Since the system is a cyber-physical system, the implementation of all these steps includes four aspects: hardware, software, machine learning models and the data stream. Hardware includes the sensors, actuators, and controllers. Software is the platform and environment which is used to collect this data and implement the models. Models are machine learning models. The data stream consists of data points which flows to and from each of these steps based on a predetermined frequency. In this document, a data point means at a certain time t, the data array of a set of sensors contains m sensors, which can be presented as:

$$X_t = [X_{t1}, X_{t2}, X_{t3}, \dots, X_{tm}$$



4.1 Detection and diagnostics

The detection function is responsible for detecting anomalies by using supervised and/or unsupervised models to classify whether the monitored data point is an anomaly or not. Unsupervised two-class classification models can determine whether the data point belongs to either the normal or abnormal class based on historical data. Supervised multi-class classification models have more flexibility, as they can be utilized to attribute anomalies to a specific class or cause.



Supervised models require a database which contains historic data with labeled classes as shown in the Table below to perform dictionary-based detection. These classes can be normal state 1, normal state 2, abnormal state 1, abnormal state 2, …, abnormal state n. For example, a detection model for a feedwater system can have “normal operation under power level 100%” as normal state 1, “normal operation under power level 80%” as normal state 2, “feedwater pump degradation” as abnormal state 1. When a new data point is fed to the supervised model, it is classified to a certain class if the sensor array of the data point matches the certain pattern in the data base. Detection based on a supervised model has high accuracy for faults that are stored in the database, but obviously won’t be able to detect fault that are outside of the database.

Table 1: Supervised models inputs and outputs

Inputs						Output
Sensor 1	Sensor 2	Sensor 3	Sensor 4	…	Sensor m	Class
X11	X12	X13	X14	…	X1m	Normal state 1
X21	X22	X23	X24	…	X2M	Abnormal state 1
…	…	…	…	…	…	…
Xn1	Xn2	Xn3	Xn4	…	XnM	Normal state 2

Unsupervised models only utilize data under "normal states" to build the model, so they are only able to classify any data point as either being within normal operational states or deviate from these normal states as an anomaly. However, unlike supervised models, unsupervised models can detect fault that have never seen before. For unsupervised models, thresholds may be applied to residuals (the difference between the predicted values and real values) or statistics to detect anomaly (data points exceed thresholds). For unsupervised models, there are multiple methods available: a regressive model may be developed, with significant deviations from this model taken as anomalies, while other approaches which are not regressive (such as single-class support vector machines) might be used. Some fault can also be included into the normal data to be treated as normal data so that the detection results exclude certain faults. This technique is usually utilized in multiple fault analysis. For example, in the feedwater system in a PWR, an anomaly detected by an unsupervised model can result from both feedwater pump and condenser degradation. One way to identify the component which is at fault is to have two different models, one that includes normal operational data + feedwater pump degradation data and one which includes normal operational data + condenser degradation data. The root cause of the anomaly can then be discovered by comparing the outputs of these two models.

Various supervised ML models such as K nearest neighbors (KNN) and decision trees can be utilized to attribute fault into a certain class. The database utilized to build these models is usually taken to contain historic component failure data across the industry, however such a database is usually quite difficult to prove as being comprehensive. Therefore, most detection models are unsupervised in order to avoid missing a fault which simply does not appear in the database. There are many ML algorithms which have been investigated in literature, such as regression, support vector machine, and ensemble trees. These models have some level of explainability and have been applied or have promising implementation status in the near future since they have less regulatory concerns. Deep learning (DL) algorithms, such as deep neutral networks also have been investigated more deeply in recent years with the application of fault detection in NPPs. Figure * gives a summary of a set of deep learning (DL) algorithm.

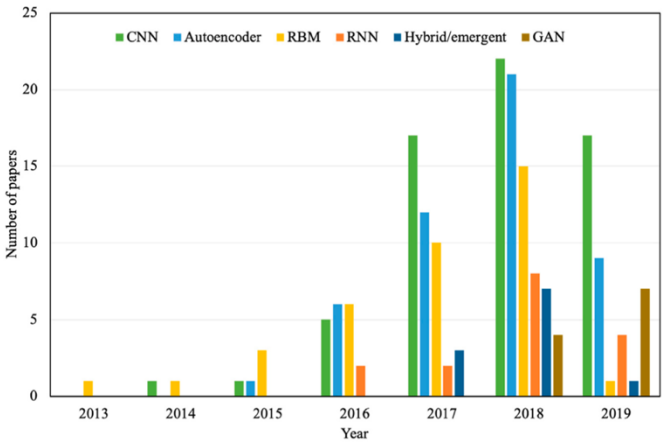


Figure 4: Prognostics and Health Management in Nuclear Power Plants: An Updated Method-Centric Review with Special Focus on Data-Driven Methods. Fronteirs in Energy Research 9 (2021): 294.

4.2 Prognostic models

Prognostic models are utilized to predict the remaining useful life (RUF), which is the time before a piece of equipment completely fails and cannot perform its intended function. Prognostic models depend on the results from fault detection and diagnostics. Given the nature of prognostics and inadequate run-to-failure data, statistics-based prognostic models are the focus of most existing literature. Regression, Markov chain, and stochastic filtering-based models are examples of methods that have been investigated to predict RUF. For example, the particle filter algorithm, a sequential Monte Carlo algorithm for nonlinear, non-Gaussian systems, is one of the common methods utilized in prognostic research. It accounts for uncertainty by updating the probabilistic state estimation with real-time measurement.

4.3 Decision making system (strategy selection & recommendation)

Once the fault is detected by detection models and the future health states are predicted by prognostics models, decision making system assesses the potential strategies and make recommendation in a timely manner to stop a fault from progressing to an emergency. An AI-based decision-making system is desired for this task, to be able to consider all possible strategies, select the optimal ones, and make a recommendation. Several decision theory methods have been investigated in the literature: Bayesian method, utility theory, and Markov Decision Process.

4.4 Implementation

To implement the autonomous control, all the above-mentioned ML models need to be implemented into a platform and interact with physical hardware. This Figure outlines the data flow: data points are acquired from sensors, actuators, and controllers such as distributed control system and fed into detection models. Detection models pulls data from database and determines the classification and analysis results

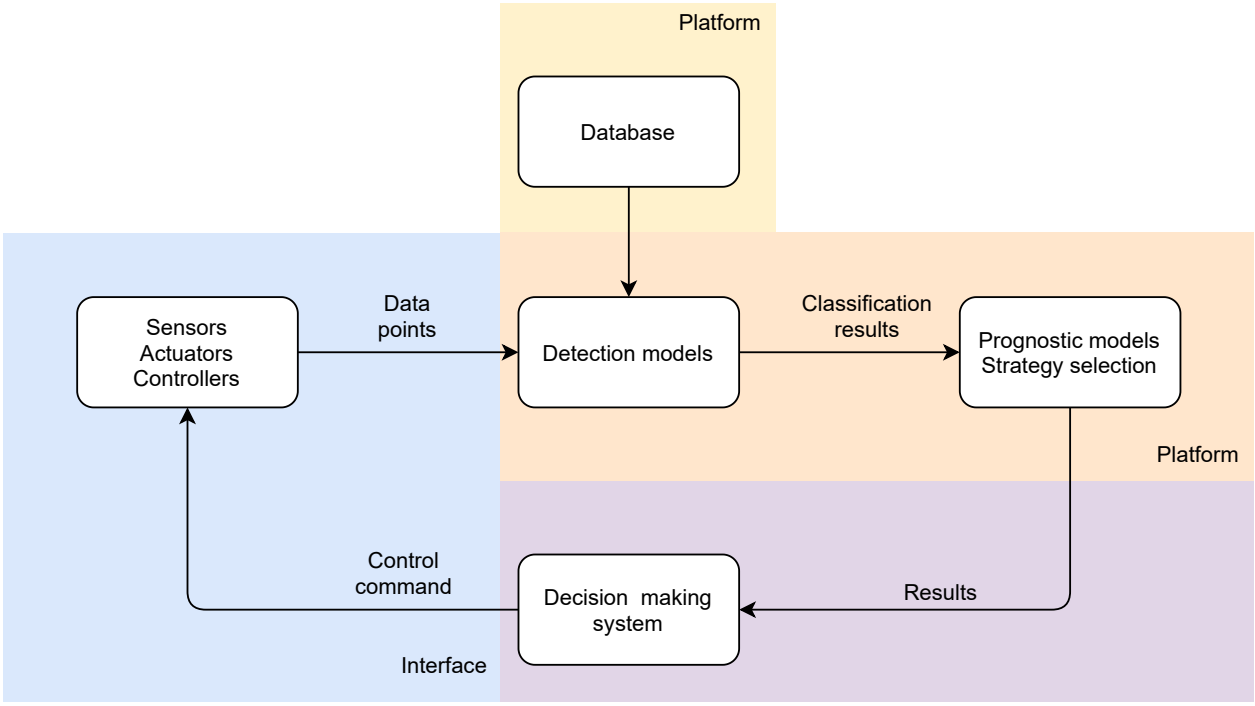


Figure 5: Data Flow Implementation

and then sends this data to prognostic models. Here, it is assumed that both the detection models and prognostics models are implemented on the same computational platform in the same core environment, such as Python. Then the results from the prognostic models are sent to the decision-making system to assess potential strategies and make recommendations. These recommendations usually involve control commands for controllers and actuators to act. It is assumed that decision making system utilizes a different platform from the detection and prognostics model.

4.5 Attack points and skill required

In the above-described system, there are four general types of attacks points that threat actors can attack: the hardware, the software environment, the ML models, and their interfaces.

Attack to hardware and the communication channels of the hardware

Attacking the hardware directly (modifying low-level PCB components and firmware, for example) is difficult, and almost always requires physical access. However, threat actors can alter the data moving into and out of the hardware using a false data injection attack. For example, attackers can tamper with the sensor lines coming from the steam generator water level meter that goes into the detection system, which in turn misleads the detection and prognostics, which can further lead to incorrect control command recommended by the decision-making system. Threat actors can also tamper with the control command recommendation coming from the decision-making system back into the actuators they're controlling, directly causing unexpected changes not in line with what the system expects. In the former situation, threat actors need to have high-level system knowledge in order to manipulate the control commands indirectly via the detection system. If the detection models have high robustness, the models may identify the single compromised sensor as a degraded or failed sensor, which may not cause the decision-making system to take any action that impacts the actuator. If the threat actors tamper with several sensors in an uncorrelated manner, then it is more likely that the detection model will determine that there is a component or system level anomaly instead of single sensor failure. In short, the attackers need to compromise multiple sensors at the same time in a coordinated and intelligent manner, having extensive knowledge of the system, in order to pull off an effective attack. Therefore, the required skill level for this kind of attack is high. In the latter situation, the attackers may change a control command arbitrarily to cause physical impact without deep knowledge about the system. However, the threat actors still need to have some knowledge such as control command thresholds to avoid triggering an alarm and exposing themselves. The skill level for this attack is medium since there are available NPP simulators that attackers could use to get a sense of system and control command behavior. Another, more advanced attack which is unlikely but possible is if the threat actors have high knowledge about the system and have access to the system's data stream. The attackers could send data from a simulator to the detection models and obtain the outputs from decision making system, and then make a surrogate model for the entire detection, prognostics, and decision-making system and then future manipulate the control command recommended by the decision system by manipulating the inputs of the detection system.

Attacks to software environment

Two types of attacks on the software environment are possible: an attack on the platform for the system, and an attack the database environment for the database. Both require threat actors having knowledge of the software platform which is used to implement the system. In the former attack, an attacker could (for example) exploit known vulnerabilities in the Python runtime in order to gain control of, or disable, the system or the entire computer running the system. This kind of attack would obviously be debilitating if

carried out, since this would be exploiting inherent vulnerabilities to break out and achieve the attacker's objectives in a straightforward manner – unlike in the hardware-based attacks, the attacker does not have to have intimate knowledge of the entire NPP, just information such as the computer and OS version, exploits for which are commonly worked on in open-source research. In the latter attack, attacker might target the database instead, since directly disabling the system, itself may result in backup systems coming online and the attack being quickly identified. The attacker may instead be able to leverage vulnerabilities in the database software to inject or remove specific fault states corresponding to planned attacks, thereby causing the system not to alarm or identify properly the anomaly when it is introduced. Either of these attacks can disable usage of the detection, prognostics, and decision-making system, which could be significant if the autonomous system is the only system that can maintain the safe operation of the NPP. It's also worth mentioning that the attacker is not limited to strictly software-based vulnerabilities when attacking the software environment – methods which exploit hardware characteristics to enable software vulnerabilities are commonly called "side-channel" attacks. Many side-channel attacks are quite well-known and easy for attackers to implement, as these attacks have been the focus of much research by the information security community in recent years. These attacks are also particularly dangerous, since unlike other attacks which can be mitigated by further research and development or software hardening, these attacks target the underlying compute fabric to enable vulnerabilities in the software layer and thus are not easily answered by using more advanced or robust techniques. Of course, these side-channel attacks are mitigated by frequent security patching policies, however patching may be accounted for by the attacker (and may even be included as part of their infiltration strategy) and patching these vulnerabilities may entail slowing down the system to the point where reliability becomes an issue. While much of these concerns might be addressed by defense-in-depth security approaches, the estimated attacker skill for both of these attacks is Medium, since exploit information and training for common software frameworks is commonly available and there is a very active research community dedicated to discovering vulnerabilities in commercial software and in discovering enabled vulnerabilities via side-channel attacks.

Attackers may perform a simultaneous attack during denial of service of the autonomous control system so that no control command will be given by autonomous control system to take the NPP to a safe state.

Attacks on models

Since the system is using machine learning models, attackers may instead develop methods specifically targeting the models themselves. This again takes the form of two classes of attacks, one focused on the detection, diagnostics, prognostics and decision-making systems themselves and another focused on the model's "supply chain". For the first class of attack, an attacker may draw upon existing research in adversarial machine learning to determine unique inputs which might cause the system to give an "incorrect" answer based on the current model state and training data. These adversarial attacks are commonly demonstrated as a minimal amount of mathematical noise added into computer vision inputs, however there is a broad field of research devoted to finding vulnerabilities in all kinds of machine learning models. The second class of attack is based on the attackers having enough resources and information to poison either the database, the models training process or the developed models, for example as part of periodic vendor-provided updates. Such attacks are also an active research topic and may be easier to accomplish than the first class, given that introducing specific information into the training process and deriving system behavior under those introduced poison states may be easier than attempting to derive the system state and determine precise vulnerabilities from that. Both attacks require detailed access by the attacker to the implemented machine learning models and/or access to the vendors supplying the models,

although once access is gained the attacker may be able to easily apply open-source reverse engineering tools and knowledge to accomplish their goals. Additionally, many of these vulnerabilities may be addressed by applying sufficient hardening techniques to the ML models, however doing so may require online updates to the underlying model and methods as all vulnerabilities will likely not be known upon first developing the model. Therefore, the estimated skill of an attacker to accomplish this attack is Medium.

Attack to the interface

Finally, an attacker may conduct an attack on interfaces between either the system and the NPP, or on the internal system interconnects. General examples of attacks on system interconnects might be direct, such as denial of service attacks, or might be indirect such as side channel attacks. For denial-of-service attacks, the most effective target for these attacks would likely be the data transmission network within the NPP itself, as targeting the industrial network would disable both knowledge of the system of the current state of the plant and would also disable system response. For internal system interconnect attacks, side-channel attacks such as CPU-based attacks which focus on effectively exploiting memory and predictive scheduling vulnerabilities might also be effectively exploited to contaminate the data being fed to the system. The attack to the interface can cause denial of service, and data tamper. This requires attackers have knowledge the specific interface that is implemented.

5. Advanced Reactor Machine Learning Threat Assessment

At the start of the FY21 research cycle for this project our team invested a month of thought into defining the boundaries of this problem space. Our dialogue mostly centered on why anyone would build into an advanced reactor architecture autonomous and remote functions. In the methodology paper we described some of the business drivers such as new types of reactor deployments such as what is being posed by the United States and Russia with their forward deployed SMRs to support military operations. We also captured the requirements to include fully autonomous generation and distribution infrastructures that would load balance across an energy system that included not only nuclear, but hydro, wind, solar, and other traditional energy generation assets. To accomplish these objectives an autonomous systems decision loop must be implemented. This decision loop is dependent upon detecting event states and using algorithms to predict future event states. The decision loop is dependent upon algorithms to select strategies for achieving operational states that fit to these future event states. The decision loop requires algorithmic support to recommend strategies and ultimately execute these strategies through interaction with plant systems. The architectures we have reviewed to date have all included Digital Twin concepts for supporting the autonomous system decision loop. This Use Case is focused on subverting the Machine Learning algorithms and processing infrastructure necessary to implement the autonomous system decision loop.

5.1 Description of the Assessment Target

This assessment will be performed against the Machine Learning infrastructure of an Advanced Reactor vendor that is implementing remote and autonomous operations capabilities into their design. The ML infrastructure includes a process for model creation that is dependent upon the type of data available for processing and the types of classifications that are required for successful model implementation. For this assessment we limit the attack space to subversion of algorithms using *evasion, poisoning, trojaning, backdooring, reprogramming, and inference attacks*.

Table 2: Categories of attacks on ML Models[e]

Stage/Goal	Espionage	Sabotage	Fraud
Training	Inference by poisoning	Poisoning Trojaning Backdooring	Poisoning
Operations	Inference attacks	Adversarial reprogramming Evasion *false negative evasion*	Evasion *false positive evasion*

For each of the six attack types represented in Table 2 we will provide a threat assessment using the methodology that includes defining a subset of representative processes, components, and functions to assess relevant to that attack type, enumerating subversion options against the target, describing the type of threat actor capabilities required to engage in this type of subversion attack, and offer up security controls and response countermeasures.

[e] Toward Data Science. How to attack Machine Learning (Evasion, Poisoning, Inference, Trojans, Backdoors). https://towardsdatascience.com/how-to-attack-machine-learning-evasion-poisoning-inference-trojans-backdoors-a7cb5832595c

The six attack scenarios based on attack types include:

- **Poisoning Attack against ML function implemented using an FPGA**

  A **process** is implemented within the autonomous systems decision loop that supports sensor data fusion (detection). Sensor data from plant field devices is aggregated by zones and levels and then passed securely to a system responsible for event classification that is then provided to a prediction process. The data arrives at the sensor fusion platform (**component**) via a TCP/IP listening daemon that writes the sensor data into a queue for processing by the ML classification engine. The ML classification engine is implemented using an FPGA (**component**) to meet the performance requirements. The FGPA gate logic outputs the ML classifications (**functions**) that are written to an output queue for ingestion by the next pipeline process element.

- **Trojaning Attack against ML classifiers exploiting the excitability of Nuclear Engineers**

  Nuclear engineering and Nuclear Power Plant operations is a world of predictability, for good reason, as stable reactor and subsystem operations is generally welcomed to ensure the safe generation of power and support the usual array of research reactor activities. For this reason, a trojaning attack, while not exactly straight-forward, would be fascinating to conduct, especially using the Nuclear Engineers as a target of the attack as it relates to their excitability. In this case a machine learning classifier **process** is targeted during training to identify trigger inputs that are reliably classified as events that Nuclear Engineers rarely see and would be excited by the observation (classification). The model implementation (**function**) is then retrained to trigger classifications based upon common observations of excitement within the neural network. To better understand this attack we suggest reading the paper, *Trojaning Attack on Neural Networks*.[f]

- **Backdooring Attack against ML Training environment to ensure persistence of attack vector**

  A threat actor has been hired to conduct a backdoor attack against a Digital Twin responsible for the strategy selection as part of the implemented autonomous system decision loop. The strategy selection algorithm utilizes a classifier using the inputs from the prediction engine and producing as an output a set of recommendations. The threat actor recognizes that the classifier **process** that runs on the Digital Twin will be trained against data that is specifically crafted for the boundaries of this operational environment. The organization that hired the threat actor realizes that with a ***Backdooring Attack*** they can increase the probability that even as the ML algorithms are exposed to updated training material that the backdooring approach will allow the attack vector they wish to target to persist and be available to them through subsequent production deployments of the model implementation **function**.

- **False Positive Evasion Attack against multi-factor Access Control system using clever inputs**

  A threat actor was hired to perform Open-Source Intelligence (OSINT) gathering against a Nuclear Power Plant. The OSINT information collected documented the use of an Access Control System that ensured that access to sensitive areas of the NPP was limited to authorized staff. The implementation of the Access Control System utilized a two-factor approach with an Id Card paired with biometric data from a facial recognition application that utilized a machine learning algorithm to classify whether the individual standing in front of the camera requesting access to the given

[f] Liu, Yingqi; Ma, Shiqing; Aafer, Yousra; Lee, Wen-Chuan; Zhai, Juan; Wang, Weihang; and Zhang, Xiangyu, "Trojaning Attack on Neural Networks" (2017). Department of Computer Science Technical Reports. Paper 1781. https://docs.lib.purdue.edu/cstech/1781

area matches against the identification and authentication data stored in the system when the person was enrolled. The thread actor offers to another group the ability to attack the machine learning implementation with a ***False Positive Evasion Attack*** that would bypass the access control system through reducing the confidence or engaging in targeted or universal misclassification.

- **Inference Attack against ML models by NPP Insider with access to Operational environment**

  A threat actor has determined that machine learning models have been implemented for classification of events as part of the autonomous system decision loop but has not been able to penetrate the ***Training*** environment. They are able to recruit an insider and provide them with the capability to attach to the production ML classifier in the operational environment such that they can use an ***Inference Attack*** to reverse engineer the ML classifier and determine which attributes are being used to perform the classification process.

- **Adversarial Reprogramming Attack against video surveillance of material access control**

  A threat actor has been hired to steal nuclear material that is located in a protected area of secure building on the site of a Nuclear Power Plant that is remotely located without staff on site. The threat actor was able to use a ***False Positive Evasion Attack*** to gain access to the building and secure area but then encountered a video surveillance system that is trained on the lock system that protects the nuclear material. The threat actor decides to use an ***Adversarial Reprogramming Attack*** such that they can move into the field of view of the protection camera without an alarm event triggering.

5.2 Notional Diagram

The following notional diagram represents the attack space that we will be performing our assessment on. The nature of Machine Learning is such that there are generally two phases: *training (including model selection and platform requirements definition) and operations or production (including model implementation on specified platforms)*.[g]

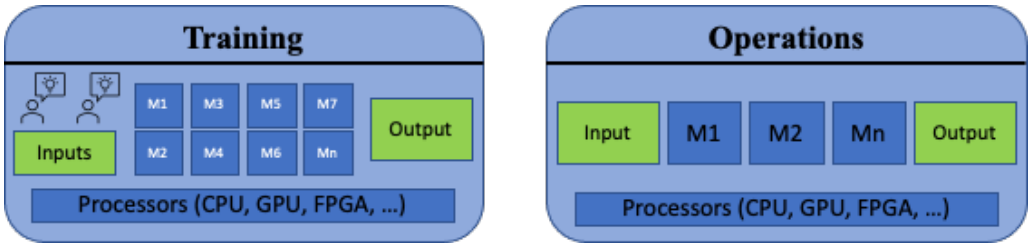


Figure 6: Notional Diagram of ML Attack Space

Our focus for this threat assessment will be on model selection based upon autonomous system decision support requirements and associated attacks against those model choices. We have included within the ***Training*** phase the humans that will influence the model selection and implementation. While each of the attack options highlighted in Table 2 has a human vulnerability associated to it, we will choose for this Use Case a subset to focus on to illustrate how this attack vector influences the other attack vectors.

[g] The authors recognize there are also an endless number of edge cases to how machine learning is implemented from training through model deployment. For the case of this paper we generalize to make the attack surface more approachable.

5.3 Enumeration of Processes, Components and Functions

Following the methodology steps, we must now enumerate the processes, components and functions of the assessment target. This *Use Case* is interesting in how the assessment targets include the Digital Twins that support the autonomous system decision loop as well as systems that support physical security of protected areas and physical security of nuclear material. We even included as an attack target the nuclear engineers responsible for training classifiers to recognize a spectrum of events from routine events observed throughout each day during plant operations to zebra events that rarely occur and when observed inspire a mix of fear and excitement, hopefully more of the later and not the former.

Aggregated Target List

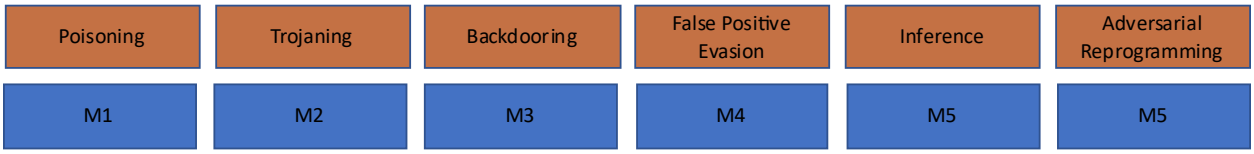
An initial target list of processes, components and functions include:

Processes (procedural)	**Algorithms** *(generically)*	**Data Targets**
Autonomous Systems Decision Loop	*Machine Learning Operational Env.*	*Instrumentation Data*
Access Control Request Adjudication	*Machine Learning Training Env.*	*Instrumentation Database*
		Strategy Knowledgebase
Organizations	**People**	*Access Control Biometrics*
NPP Machine Learning Vendors	*NPP Nuclear Engineers*	*Video Surveillance*
NPP	*Advanced Reactor Design Team*	*FPGAs*

The reader should notice that this aggregated target list, while overlapping with one provided in the first Use Case, contains a broader set of processes, components, and functions. There is nothing that prevents us from deep diving into any one of these areas, but to illustrate the ability for this methodology to be flexible relative to depth of analysis, we will, for example, look at the Machine Learning attacks as describe at the end of Section 5.1 and within the Cyber Threat Assessment identify the subversion options, associated threat actor attributes, and some recommended security controls and countermeasures based upon the published academic literature. Once again, our hope in FY22 is to instantiate these environments and assess the feasibility of each of these types of attacks and then provide guidance to the Advanced Reactor vendors on how to approach the secure implementation of Machine Learning Training and Operational environments.

5.4 Cyber Threat Assessment

The methodology calls for separate threat assessments for Autonomous System Processes and then for System Components and Functions. Our focus for this Use Case is specifically the ML Training and Operations environment both for the use of ML algorithms in support of the autonomous systems decision loop but also the use of ML in support of plant operations in the areas of physical security, material security, and access control. We will therefore take the aggregated target list along with the six attack scenarios and describe them collectively within each of the threat assessment phases.



Phase 1: Subversion Options against the Target (process, component, or function)

The assessment question we need to answer in this case is what are the conditions and prerequisites for conducting each of these attacks on ML implementations. By defining the six attack types we have already defined the subversion options against the targets, minus the nuances of applying these subversion options against the ML implementations, such as a classification algorithm implemented on an FPGA versus a GPU versus a traditional CPU.

- **Poisoning Attack against ML function implemented using an FPGA**

  In the example provided in Section 5.1, we are interested in attacking the ML implementation within the ***Operations*** environment. In order to accomplish this, we need to alter the data inputs to the model which are arriving from field devices. The ML model has been implemented within an FPGA that has been programmed to perform the model classifications. The attacker in this case has the following subversion options:

  (1) Modify the field device data prior to it arriving at the TCP/IP listening daemon such that it is passed directly into the classification model. This approach may prevent any defensive triggers on the ML processing platform but also may fail if the input munging varies too far from what is statistically acceptable and the data never arrives due to failed validation checks.

  (2) Attack the TCP/IP daemon such that select field device data is rewritten to achieve the requirements of the poisoning attack against the ML function. This is arguably a better target than the field devices assuming the field device communication pathway attack is not limited to an approachable number of pipes. The advantage of this, similar to a water-hole attack is that a single point of compromise yields access to all the data munging options necessary for this attack to at least be plausible if not successful. One concern would be handling the input queue since their design specifications required an FPGA to meet processing time requirements therefore input message volume is expected to be significant and thus could fail if attacked.

  (3) Attack the FPGA ML implementation. We would refer the reader to our publication last year on attack pathways against FPGAs. This subversion option exists although would likely be the costliest from a resource perspective.

- **Trojaning Attack against ML classifiers exploiting the excitability of Nuclear Engineers**

  This attack after reading through the paper *Trojaning Attack on Neural Networks*[h] (also referenced in Section 5.1). On Page 2 of this paper, the authors state:

  If a neuron in a hidden layer is considered representing some feature (that is difficult for humans to interpret and hence stealthy), we are essentially constructing the trigger that possesses strong presence of such features. ***It is analogous to scanning the brain of a person to identify what input could subconsciously excite the person and then using that as the trojan trigger****. Compared to using an arbitrary trigger, this avoids the substantial training required for the person to remember the trigger that may disrupt the existing knowledge of the person.*

[h] Liu, Yingqi; Ma, Shiqing; Aafer, Yousra; Lee, Wen-Chuan; Zhai, Juan; Wang, Weihang; and Zhang, Xiangyu, "Trojaning Attack on Neural Networks" (2017). Department of Computer Science Technical Reports. Paper 1781. https://docs.lib.purdue.edu/cstech/1781

The attack scenario that we derived was to utilize this concept of excitability related to intersection of Nuclear Power Plants, Nuclear Engineers, and events of interest, and work our way backwards to conduct a ***Trojaning Attack*** such that when we (the attacker) choose to deliver the type of effect that would likely trigger an alarm due to excitability, that the ML classifier will take the validated input data and when processed within the trojan-ed neural network the output event classification is benign, relative to the defined set of critical events. The attack in this case has the following subversion options:

(1) Attack the model training process by creating a taxonomy of critical events to target for misclassification and then identify with a small group of unwitting Nuclear Engineers which inputs would excite their neurons such that an ML programmed would include those attributes in the model implementation.

(2) Attack the classification process by identifying trojan triggers based upon sessions with the unwitting Nuclear Engineers that when injected cause the ML classification process to return the altered results.

- **Backdooring Attack against ML Training environment to ensure persistence of attack vector**

  The attacker in this case wants to conduct a ***Backdooring Attack*** to ensure persistence of the misclassifications after the model is exposed to updated training data. The researchers who have implemented this attack have identified operational environments that use transfer learning as vulnerable to this type of attack. As autonomous systems are architected with supporting Digital Twins that are dependent upon ML classifiers, it would seem logical that transfer learning would be prevalent as reactor architects and vendors hope to take advantage of previously trained models to operate their reactor implementations. This attack will take place on the ***Training*** environment and the subversion options include:

  (1) Injecting the ***Backdoor-ed Training Data*** into the training process, taking advantage of any opportunities for injection during the exchange within the context of transfer learning.

  (2) A variant of (1) would be to conduct an unwitting insider attack against the staff responsible for training the models, although functionally it is the same subversion option, just an alternative engagement pathway, of which there are surely a few more that could be enumerated.

- **False Positive Evasion Attack against multi-factor Access Control system using clever inputs**

  This attack is specifically focused on image classification related to identification and authentication of users to protected areas of the Nuclear Power Plant, or really any facility where there are areas that require individual or group-based access control. This misclassification attack has the goal to take input that should be rejected and classify it as acceptable, based upon defined thresholds. The practical implementation of this is to have an unauthorized user swipe a stolen Id card and have their face (image) classification return as an acceptable match. Subversion options for this attack include:

  (1) Depending upon the implementation, there are usually tunable parameters in the operational environment that allow for confidence intervals to be set. While this is not an attack directly against the ML component, subverting the confidence interval attributes would allow this ***False Positive Evasion*** attack to be successful.

(2) One of the unique aspects to this attack is that the ML models need to be updated based upon newly enrolled images which provides an opportunity for the attacker to engage in a poisoning attack that is not limited to a one-time or periodic model training activity.

(3) Not entirely sure this would work but attacking the enrollment process by using an image merging technique to combine a newly enrolled target with a known-good target based upon known classification attributes may lead to a successful attack. This would imply there is a subversion option that exists further left in the enroll – training lifecycle.

- **Inference Attack against ML models by NPP Insider with access to Operational environment**

  As described in Section 5.1, this attack is focused on discerning model attributes through interaction with model inputs and outputs. The attack is bounded by only having access to the ***Operations*** environment. The one subversion option in this case:

  (1) Gaining access to the ***Operations*** environment to interact with the ML model. In this case we bounded the access to a trusted insider so that we did not have to penetrate into the networks and systems externally. Our goal is to subvert the ML classifier by targeting extracted attributes and crafting an attack on the ***Training*** or ***Operations*** environments.

- **Adversarial Reprogramming Attack against video surveillance of material access control**

  In this case the attack surface includes a video surveillance system that is trained on a protection mechanism for physical protection of nuclear material (think of a camera with a fixed point of interest on a lock). The subversion options (1) and (2), in this case, assume knowledge of the model attributes and include:

  (1) Conduct a gradient-based attack such that the modifications injected into the video / image stream are maximized to impact the classifier's output.

  (2) Conduct a score-based attack such that the modifications injected into the video / image stream are optimized against the classifier's confidence score.

  (3) Conduct a decision-based attack such that the video / image streams are manipulated to produce classifications that meet attack requirements such as not triggering an alert.

Phase 2: Threat Actor Attributes and Capabilities

There are two central characteristics of threat actors that would be capable of conducting these types of attacks: expertise in Machine Learning and expertise in Nuclear Reactors and Power Plant architectures and operations. Each of the attack scenarios provided requires access to either the ML ***Training*** or ***Operations*** environments and each of these environments has some unique characteristics. In the ***Training*** environment the threat actor will have to be familiar with the model language and training data attributes that will be unique to the classifier being implemented. The threat actor will have the opportunity to observe the model compilation and deployment pipeline which should help in better understanding the ***Operations*** environment. In that environment the threat actor will need to be familiar with data pipelines and the varying types of processing platforms that implement ML classifiers to include traditional CPUs, GPUs, and in some instances, FPGAs. Add to this the highly specialized nature of data related to reactor and subsystem operations and the threat actor will experience a higher success rate in a shorter period of time should them be able to maximize their capabilities in both of these areas.

Phase 3: Security Controls and Response Countermeasures

We assume that attacks against the *Training* and *Operations* environments outside of attacks on the Machine Learning components will be addressed in a separate threat assessment. While they could be included here our primary focus is on security controls and response countermeasures related to the ML algorithm attacks that are described in Section 5.1 and Phase 1 of this section. In the paper ***Challenges and Countermeasures for Adversarial Attacks on Deep Reinforcement Learning*** the authors included a summary table of countermeasures for attacks against Machine Learning. The full listing with references to source papers is included in Appendix 4. They grouped these defenses into five (5) categories: *Adversarial Training; Robust Learning; Adversarial Detection; Defensive Distillation; and Game theoretic approach.* Each of these defensive approaches provides some coverage against the six (6) attacks described in Section 5.1.

Adversarial Training

This approach is the one that seems most obvious. Take the known adversarial models / training data and apply it to the model being protected such that the classifier is aware of the perturbations. This would apply for instance against the ***Poisoning Attack*** and ***Backdooring Attack*** where each attack includes injection during the model training phases.

Robust Learning

This approach will be familiar to those who have worked on cyber-security protocols and defensive solutions. In this case the training mechanism introduces a timing component that adds noise to parameter states. Variations on this approach focus on the use of noise such as what Kumar recommends in using noisy rewards, but the core premise is that adversarial models that are not trained with recognition of the noise will underperform compared to those that have, thus indicating that the ML classifier is probabilistically under attack. This countermeasure would be applicable to the ***Inference Attack*** since it aligns optimally with black-box attacks.

Adversarial Detection

For this approach the ML classifier includes a mechanism for segregating true samples from adversarial ones such that the adversarial ones can be discarded, and the ML classifier need not be any more aware than as designed. One approach recommended by Havens utilizes a supervisory agent and a set of policies that are applied such that when data enters the ML classifier adversarial ones will cause unexpected policy states. This countermeasure would be applicable to ***Trojaning*** and possibly ***False Positive Evasion Attacks***.

Defensive Distillation

Defensive distillation is a training method where a model is trained to predict the output probabilities of another model which is trained on the baseline standard to give more importance to accuracy.[i] It is not entirely clear to the authors of this paper which attack types this could be applied to.

Game Theoretic Approach

This approach includes an interactive loop between the adversary and the ML model they are attacking. This approach extends to multi-player interactions with adversarial injects. There may be a home for this approach but not entirely sure where it is yet.

[i] Challenges and Countermeasures for Advanced Attacks on Deep Reinforcement Learning. https://arxiv.org/pdf/2001.09684.pdf

Page intentionally left blank

Appendix 1
Threat Actor Attributes and Characteristics

This appendix is for the threat assessment team to use as a reference when performing the analysis of threat actor attributes and capabilities.

Attacker Advantages

When considering cyber threat actor capabilities, it is helpful to understand how they will be used against target systems. These operational details provide insight into how defensive security controls and countermeasures can be implemented to protect and defend against cyber-attacks. For example, cyber threat actors operate using *tight feedback loops*, maneuvering against target systems while quickly pivoting based upon effect delivery observables.

Tight Feedback Loops

Very little doubt about success

Better understanding of attack costs

Ability to choose attack timing and cadence

Complexity of attack surface

Ability to discard unhelpful tools and tactics

Clarity and lack of blurred lines

Attackers write more code

Least privilege failures are found too late

Defensive attitude

Defender Advantages

When constructing security controls and defensive countermeasures it is helpful to remember how the defensive position is advantaged against threat actors and their capabilities. For example, cyber threat actors do not (usually) have a complete view of the target environment, but the cyber defenders (should) do. Given the complexity of autonomous system implementations, defensive teams should be able to use this to their advantage.

Attackers only see half of the chess board, Defenders see the whole board

Forcing attackers to reveal themselves will cause them to burn toolsets and methodologies (TTPs)

If attackers are discovered once, they have to use or develop new capabilities

Defender Disadvantages

Similar to taking advantage of advantageous positions as a defender, defenders must also recognize and adapt to their limitations. For example, when designing security controls and countermeasures for an autonomous system, components and perceived targets must be prioritized, but it is not always clear on what the attacker is interested in and whether the defensive prioritization is mostly correct.

Not always clear on what the attacker is interested in

Never really know if their chosen security controls are helpful

Complexity: Defenders have to understand everything in their environment

Defensive specialization is not common

Lack of attacker awareness

Defender Philosophy and Approach

When planning out a defensive strategy, there are at least three tenets to live by:

Expect to be hacked

Attackers think in graphs… Defenders should as well. Stop thinking in lists!

When performing attack graph analysis, choose detection chokepoints with care

Appendix 2
Characterizing Cyber Threat Actors by Tier

Threat Actors belong to one of three classes of increasing sophistication:

Class I (Tier I and II) Investment: $0 to $10000+ USD

Rely on others for tool and exploit code development. Limited ability to develop their own tools based upon publicly known vulnerabilities.

Class II (Tier III and IV) Investment: $1,000 to $10M+ USD

Develop their own cyber-attack tools and exploits. Proficient in reconnaissance and targeting, and in the use of user and kernel mode root kits. Adept at discovering new vulnerabilities.

Class III (Tier V and VI) Investment: $1M to $1B+ USD

Well-resourced actors dedicated to injecting/creating vulnerabilities in systems. Capabilities include full spectrum operations (cyber + military + intelligence).

Tier I
Practitioners who rely on others to develop the malicious code, delivery mechanisms, and execution strategy.

- Hackers who are limited to toolkit and packaging/delivery capabilities (e.g., Kali Linux, Metasploit).

- Hackers not capable of developing custom tools or implants.

Tier II
Practitioners with a greater depth of experience, with the ability to develop their own tools.

- Hackers capable of developing their own malicious code

- Hackers with curated objectives, developing their own tools but not yet skilled enough to identify and exploit non-published vulnerabilities.

Tier III
Practitioners focused on discovery of novel vulnerabilities, adept at installing user/kernel mode root kits, capable of more narrow targeting using data mining techniques.

- Hackers experience in conducting vulnerability analysis of humans, hardware and software platforms

- Hackers capable of crafting custom implants using chained exploits and custom tools for data mining

Tier IV
Criminal or state actors who are organized, highly technical, proficient, and well-funded

- Hacking Teams with resources to perform narrow protocol analysis and reverse engineering

- Hacking Teams capable of toolset lifecycle management including weaponization of new vulnerabilities

Tier V
State actors capable of vulnerability placement through active programs against commercial products

- Hacking Teams with resources to conduct sophisticated supply chain attacks

- Hacking teams capable of hardware destruction/disruption; proficient in information and influence ops.

Tier VI
States with the ability to successfully execute full spectrum (cyber capabilities in combination with all of their military and intelligence capabilities) operations to achieve a specific outcome in the political, military, or economic, etc. domains.

Appendix 3
Hardware Supply Chain Attack Options

Overview

We include this appendix based upon some artifacts collected throughout the Use Case development process. In this case we provide the reader with some ideas on how to think through hardware supply chain attacks and how they might be assessed within the context of advanced reactor architectures and the inclusion of features that support remote and autonomous operations.

Description of the Assessment Target I

A large group of hacktivist-class attackers steal IT remote access passwords through phishing attacks. These attackers eventually compromise the IT Windows Domain Controller, create new accounts for themselves, and give the new accounts universal administrative privileges, including access to ICS assets that have a trust dependency on the Domain Controller. The attackers log into the ICS equipment and observe the operation of the ICS HMI until they have learned what many of the screens and controls do. At that time, the group takes over the HMI and uses it to subvert the physical process. Simultaneously, the attackers use the administrative credentials to log into ICS equipment, wipe the hard drives, and where possible, zero out the equipment firmware.

Variations: When targeting other kinds of industries, similar attacks are possible, wiping control system equipment, and triggering unplanned shutdowns.

Sophistication: This is a summary of the attack techniques used in the 2016 attack on several Ukraine electric distribution companies. The attackers had good knowledge of cyber systems, but limited knowledge of electric distribution processes and control systems.

Consequences: In the case of the attacks on Ukraine, power was shut off to over 200,000 people, for up to 8 hours. Power was only restored when technicians travelled to each of the affected substations, disconnected control system computers, and manually turned-on power flows again. More generally, unplanned shutdowns are a consequence of this class of attack, and possibly emergency, uncontrolled shutdowns with the potential for equipment damage that accompanies such shutdowns.

Description of the Assessment Target II

A sophisticated attacker compromises the IT network of an enterprise with a heavily defended industrial site. The attacker steals information about which vendors supply the industrial site with servers and workstations, as well as which vendors routinely ship that equipment to the site. The attacker then develops a relationship with the delivery drivers in the logistics organization, routinely paying the driver modest sums of money to take 2-hour lunch breaks, instead of 1-hour breaks. When IT intelligence indicates that a new shipment of computers is on its way to the industrial site, the agency uses the 2-hour window to break into the delivery van, open the packages destined to the industrial site, insert wirelessly accessible single-board computers into the new equipment, and then re-package the new equipment so that the tampering is undetectable. Sometime after IT records show that the equipment is in production, the attackers access their embedded computers wirelessly, to manipulate the physical process. The attackers eventually impair equipment protection measures, crippling production at the plant, through what appear to be a long string of very unfortunate, random, equipment failures.

Sophistication: This is an attack be a very sophisticated adversary. This attacker has the physical “feet on the street” to carry out covert actions, such as breaking into the delivery van, and quickly disassembling, modifying, reassembling, and re-packaging compromised equipment. The attacker is cyber-sophisticated, maintaining a long-term presence on the target's IT network, and understanding the design of a variety of

computer equipment enough to understand how to subtly insert additional hardware into that equipment. The attacker has a high degree of engineering sophistication as well, to understand the structure of the physical process, the control systems, and the equipment protection systems enough to design and carry out physical sabotage and making damaged equipment look like random failures.

Consequences: Costly equipment failures and plant production far below targets.

Description of the Assessment Target III

IIOT Pivot: Hacktivists annoyed with the environmental practices of an industrial site learn from the popular press that the site is starting to use new, state-of-the art, Industrial Internet of Things edge devices from a particular vendor. The attackers search the media to find other users of the same components, at smaller and presumably less-well-defended sites. The hacktivists target these sites with phishing email and gain a foothold on the IT and ICS networks of the most poorly defended of these IIoT-using sites. The hacktivists gain access to the vendor's IIoT equipment at the sites and discover that the operating system for these devices is an older version of Linux, with many known vulnerabilities. The attackers take over one of the IIoT devices. After looking at the software installed on the device, they conclude that the device is communicating through the Internet with a database in the cloud from a well-known database vendor. The attackers download Metasploit to the IIoT device and attack the connection to the cloud database with the most recently released exploit for that database vendor. They discover that the cloud vendor has not yet applied a security update for that vulnerability, and they take over the database servers in the cloud vendor. In their study of the relational database and the software on the compromised edge devices, the hacktivists learn that the database has the means to order edge devices to execute arbitrary commands. This is a "support feature" that allows the central cloud site to update software, reconfigure the device, and otherwise manage complexity in the rapidly evolving code base in this edge device. The hacktivists use this facility to send commands and standard attack tools and other software to the edge devices in those ICS networks the hacktivists regard as environmentally irresponsible targets. Inside those networks, the attackers use these tools and remote-command facilities to look around for a time and eventually erase hard drives or cause what other damage they can, triggering unplanned shutdowns. In short, hacktivists attacked a heavily defended client of cloud services, by pivoting from a poorly defended client, through a poorly defended cloud.

Sophistication: These attackers are of moderate cyber-sophistication. They can download and use public attack tools that can exploit known vulnerabilities, they can launch social engineering and phishing attacks, and they can exploit permissions with stolen credentials. Hacktivists usually have a very limited degree of engineering sophistication.

Consequences: Unplanned shutdowns, lost production, and possible equipment damage.

Appendix 4
Defenses against Attacks on Machine Learning

Proposed Techniques	Effective Against
Adversarial Training using Random Noise & FGSM Kos et al. [54]	Random Noise & FGSM Attacks
Adversarial Training using Noise & Gradient-Based Attacks Pattanaik et al. [48]	Noise & Gradient Based Attacks
Adversarial Training using corrupted nodes in SDN Han et al. [46]	Node Corruption Falsifying Attacks
Adversarial Training using Perturbed States Behzadan et al. [67]	Attacks Perturbing a Considerable # of States
Gradient Band-Based Adversarial Training Chen et al. [63]	Gradient Band Based Adversarial Attacks
Noisy Exploration Behzadan et al. [68]	State Perturbation Attacks
Adversarially Robust Policy Learning (ARPL) Mandlekar et al. [70]	State Perturbation Attacks
Robust Adversarial Reinforcement Learning (RARL) Pinto et al. [71]	Attacks Targeting the Performance
Action-conditioned Frame Prediction Module Lin et al. [76]	Attacks Perturbing the States
Meta-learned Advantage Hierarchy (MLAH) Havens et al. [77]	Training-Time Poisoning Attacks
PCA for Adversarial Detection Xiang et al. [78]	Attacks Perturbing the States
Reward Confusion Matrix Wang et al. [79]	Attacks Perturbing the Rewards
Threatened Markov Decision Processes (TMDPs) Gallego et al. [80]	Attacks Perturbing the Reward Generation
Game-Theoretic Approach Bravo et al. [81]	Noise Based Attacks
Game-Theoretic Approach Ogunmolu et al. [82]	Attacks Targeting the Policy
Benchmarking Behzadan et al. [83]	Generic Adversarial Attacks
Water Marking Behzadan et al. [58]	Model Extraction Attacks
Adversarially Guided Exploration (AGE) Behzadan et al. [69]	Limited Attack Samples
Wasserstein Robust Reinforcement Learning (WR2L) Abdullah et al. [72]	Generic Adversarial Attacks
Distributionally Robust Policy Iteration Smirnova et al. [73]	Attacks Targeting the Policy
PR-MDPs & NR-MDPs Tessler et al. [74]	Generic Adversarial Attacks
Noise Filter Kumar et al. [75]	Attacks Perturbing the Rewards

Defenses against Attacks on Machine Learning Sources

[54] J. Kos and D. Song, "Delving into adversarial attacks on deep policies," arXiv preprint arXiv:1705.06452, 2017.

[48] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, "Robust deep reinforcement learning with adversarial attacks," in Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 2040–2042.

[46] Y. Han, B. I. Rubinstein, T. Abraham, T. Alpcan, O. De Vel, S. Erfani, D. Hubczenko, C. Leckie, and P. Montague, "Reinforcement learning for autonomous defence in software-defined networking," in International Conference on Decision and Game Theory for Security. Springer, 2018, pp. 145–165.

[67] V. Behzadan and A. Munir, "Whatever does not kill deep reinforcement learning, makes it stronger," arXiv preprint arXiv:1712.09344, 2017.

[63] T. Chen, W. Niu, Y. Xiang, X. Bai, J. Liu, Z. Han, and G. Li, "Gradient band-based adversarial training for generalized attack immunity of A3C path finding," arXiv preprint arXiv:1807.06752, 2018.

[68] B. Vahid and A. Munir, "Mitigation of policy manipulation attacks on deep Q-networks with parameter-space noise," in International Conference on Computer Safety, Reliability, and Security. Springer, 2018, pp. 406–417.

[70] A. Mandlekar, Y. Zhu, A. Garg, L. Fei-Fei, and S. Savarese, "Adversarially robust policy learning: Active construction of physically plausible perturbations," in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017, pp. 3932– 3939.

[71] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017, pp. 2817–2826.

[76] Y.-C. Lin, M.-Y. Liu, M. Sun, and J.-B. Huang, "Detecting adversarial attacks on neural network policies with visual foresight," arXiv preprint arXiv:1710.00814, 2017.

[77] A. Havens, Z. Jiang, and S. Sarkar, "Online robust policy learning in the presence of unknown adversaries," in Advances in Neural Information Processing Systems, 2018, pp. 9916–9926.

[78] Y. Xiang, W. Niu, J. Liu, T. Chen, and Z. Han, "A PCA-based model to predict adversarial examples on Q-learning of path finding," in 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC). IEEE, 2018, pp. 773–780.

[79] J. Wang, Y. Liu, and B. Li, "Reinforcement learning with perturbed rewards," arXiv preprint arXiv:1810.01032, 2018.

[80] V. Gallego, R. Naveiro, and D. R. Insua, "Reinforcement learning under threats," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 9939–9940.

[81] M. Bravo and P. Mertikopoulos, "On the robustness of learning in games with stochastically perturbed payoff observations," Games and Economic Behavior, vol. 103, pp. 41–66, 2017.

[82] O. Ogunmolu, N. Gans, and T. Summers, "Minimax iterative dynamic game: Application to nonlinear robot control tasks," in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018, pp. 6919–6925.

[83] V. Behzadan and A. Munir, "Adversarial reinforcement learning framework for benchmarking collision avoidance mechanisms in autonomous vehicles," arXiv preprint arXiv:1806.01368, 2018.

[58] V. Behzadan and W. Hsu, "Sequential triggers for watermarking of deep reinforcement learning policies," arXiv preprint arXiv:1906.01126, 2019.

[69] V. Behzadan and W. Hsu, "Analysis and improvement of adversarial training in dqn agents with adversarially-guided exploration (age)," arXiv preprint arXiv:1906.01119, 2019.

[72] M. A. Abdullah, H. Ren, H. B. Ammar, V. Milenkovic, R. Luo, M. Zhang, and J. Wang, "Wasserstein robust reinforcement learning," arXiv preprint arXiv:1907.13196, 2019.

[73] E. Smirnova, E. Dohmatob, and J. Mary, "Distributionally robust reinforcement learning," arXiv preprint arXiv:1902.08708, 2019.

[74] C. Tessler, Y. Efroni, and S. Mannor, "Action robust reinforcement learning and applications in continuous control," arXiv preprint arXiv:1901.09184, 2019.

[75] A. Kumar et al., "Enhancing performance of reinforcement learning models in the presence of noisy rewards," Ph.D. dissertation, 2019.

Page intentionally left blank