A Generic Missile Defense Model for Use in Cybersecurity Assessments

Shelton R. Wright, William M. Meehan, Thomas H. Morris
The University of Alabama in Huntsville

Huntsville, AL

srw0010@uah.edu, wmm0006@uah.edu, thm0009@uah.edu

John A. Bland, C. Daniel Colvett, Christian T. Schenck
U.S. Army Combat Capabilities Development Command Aviation & Missile Center
Redstone Arsenal, AL

ABSTRACT

This paper discusses a generic missile defense system model used for examining impacts of cyber-attacks against the system. As cyber-warfare has become nearly ubiquitous in military theaters throughout the world, there exists a growing need to improve the cyber resiliency of defense systems. However, analysis of the potential vulnerabilities of these systems presents a multitude of barriers when such analysis necessitates the use of the physical system itself. For missile defense systems, barriers range from accessibility of the systems to the risk of physical harm. Virtual models, such as the one outlined in this paper, provide invaluable, low cost, repeatable analysis. The model is designed to be modular and scalable, allowing a wide variety of missile defense systems and scenarios to be emulated. Modeling each component's physical, state, and networking characteristics to a high level of fidelity allows for the identification and exploitation of vulnerabilities and observation of the consequences within the system. The model was developed using open-source information and insight from missile defense experts and incorporates three types of components. Two of these, the Radar and Launcher, allow a variable number of each to be included in a given simulation. Additionally, a single Command and Control unit interacts with and instructs each Radar and Launcher. These components form isolated nodes on a network that employ communication protocols representative of those in real systems. A Simulink model provides physical data of incoming ballistic threats and launched interceptors, which stimulates and interacts with the system in real time. The model has been used to analyze the propagation of malware and its effects throughout missile defense systems. The data generated by logging component states and capturing network traffic during this analysis has informed the development of machine learning algorithms for the detection of cyber-attacks.

ABOUT THE AUTHORS

Dr. Shelton R. Wright is an experienced leader of cybersecurity engineering teams with expertise in digital twinning, SCADA/ICS, red teaming, software development, and reverse engineering. He has published multiple papers on modeling and simulation of cyber physical systems for use in cybersecurity assessments. Dr. Wright is currently an Assistant Director of the Center for Cybersecurity Research and Education at the University of Alabama in Huntsville.

William M. Meehan is a mathematician supporting cybersecurity research. He has experience in modeling and simulation of cyber physical systems for use in cybersecurity assessments, hardware security, machine learning, and elliptic curve cryptography. Mr. Meehan is currently a Research Engineer at the Center for Cybersecurity Research and Education at the University of Alabama in Huntsville.

Dr. Thomas H. Morris has over 29 years experience as an electrical and computer engineer and is a noted expert in cybersecurity for cyber physical system and industrial control systems. He published over 80 peer-reviewed articles, conference papers, and book chapters on cybersecurity modeling and simulation, vulnerability assessment, intrusion detection, and machine learning. Dr. Morris is the founding Director of the Center for Cybersecurity Research and Education at the University of Alabama in Huntsville.

Dr. John A. Bland is currently employed as a computer engineer and cybersecurity researcher at a Department of the Army research center. He received a B.S.E. degree in computer engineering, an M.S. in software engineering, and a

Ph.D. in Computer Engineering with a focus on modeling and simulation and cybersecurity from the University of Alabama in Huntsville. Prior to working in the field of cybersecurity, Dr. Bland has worked in multiple hardware-in-the-loop facilities and supported multiple live test events. His experience includes developing high fidelity simulations, embedded system specification, test planning, and system analysis. His research interest includes applied cybersecurity, machine learning, and application of simulations.

Christian T. Schenck is a computer engineer and cybersecurity research engineer at a Department of the Army research center. He holds a M.S. degree in computer engineering with a concentration in cybersecurity from the University of Alabama in Huntsville. His research areas include machine learning, system model development, supply chain risk management, and secure RF communications.

Dr. C. Daniel Colvett is a computer engineer at a Department of the Army research center. He has worked in network security, secure software development, and cyberattack simulations during his 10-year tenure with the Department of the Army. He completed his Ph.D in Systems Engineering from the University of Alabama in Huntsville in 2023.

A Generic Missile Defense Model for Use in Cybersecurity Assessments

Shelton R. Wright, William M. Meehan, Thomas H. Morris
The University of Alabama in Huntsville

Huntsville, AL

wmm0006@uah.edu, srw0010@uah.edu, thm0009@uah.edu

John A. Bland, C. Daniel Colvett, Christian T. Schenck
U.S. Army Combat Capabilities Development Command Aviation & Missile Center
Redstone Arsenal, AL

INTRODUCTION

The model discussed in this paper is constructed using open-source information. Its purpose is to serve as a tool for cybersecurity analysis of missile defense systems for a wider group of researchers than is normally allowed by a potentially similar model constructed from classified or proprietary information. This model, by nature of being created from open-source information, is inherently of lower fidelity than a model created using detailed documentation of specific real-world systems. The features and behaviors common to many missile defense systems have been incorporated into this model to achieve a higher degree of resolution. As such, the model has been kept generic. The intention is that the model be versatile and modular to allow it to act as a wide range of real-world systems. In addition to the modularity of the model itself, it is also possible to manipulate the physical objects the model interacts with during simulation to allow for modularity in the scenario being simulated.

When running a simulation of a given scenario, the model makes use of three types of components. These are the radar, launcher, and command and control unit (C2). The internal operations of each component are isolated from other components such that the components form individual, distinct nodes on a network. The number of possible radars and launchers is configurable as are the geographic positions and orientations of each radar and launcher. Some number of radars and launchers, along with a C2 unit, comprise a battery. In one battery there will only ever be one C2 unit, and as it does represent a distinct physical component it also has a configurable location.

In addition to the modeled components there are several other entities that comprise this model, each of which is configurable to allow for further modularity in simulation. To stimulate the system with data during the simulation, the model utilizes a six degrees of freedom model of a ballistic missile. The ballistic missile model is based on the model described in (Lambrechts and Bennani, 1997) and refined using the information in (Nichols, 1977). The number of ballistic threats in a given simulation is configurable as are the initial distance, height, and entry angle for each threat. In addition to the incoming threats, the model incorporates a six degrees of freedom interceptor missile model which, when deemed necessary, serves to destroy a given incoming ballistic threat.

Each radar in a simulation has an area within which it can scan for objects called its field of regard. This area is bounded by a maximum radius in meters which defines a sphere, the center of which is taken to be the cartesian coordinates of the radar. It is then further bounded by maximum and minimum azimuth and elevation angles, all of which define a segment of a spherical wedge. If an object is determined to be within this field of regard, the object is said to be visible by the radar. Another entity included in the model is a set of defended assets. These are higher value targets that, if targeted by an incoming ballistic threat, warrant the launch of an interceptor missile. These defended assets are given a location and a radius which defines a circle around the asset. If the projected impact point for an incoming threat falls within this radius, it is assumed that the ballistic threat is targeting this defended asset.

The resulting model and simulations have two contributions. First, the generic missile defense model provides a platform to study and analyze potential vulnerabilities and impacts of exploitations that may be found in similar systems. These findings can trigger a need to evaluate a real system in a different environment. Second, the platform provides a method to create large data lakes of information of a system both under normal operation and while under attack. Because the model is generic, the data lake can be made available to others for use in the development of artificial intelligence and machine learning algorithms tailored to weapon systems.

MODELING APPROACH

Simulation Environment

The environment used to run simulations with this model is housed in a VirtualBox virtual machine (VM). The VM uses an Ubuntu 22.04 Linux distribution as its operating system. The scenario that is run for a given simulation is determined by a configuration file stored on the VM. This configuration file determines the number of components, incoming ballistic threats, and interceptors that will be included in the simulation. This configuration file also determines the positions, orientations, and initial conditions and where applicable the IP addresses of each object being simulated.

At the beginning of a simulation, a Docker container is created for each component. In each of these docker containers, dependent on which component they represent, is included a set of python scripts that executes the component's operations. Additionally, both a MATLAB session to perform the calculations for incoming threats and outgoing interceptors and a coordinator is started on the host VM. When each docker container is created, it receives the configuration file and reads the information pertinent to the component it represents. To provide visualization of a simulation, the requisite data is port forwarded to a SIMDIS (see figure 1) plugin located on the host computer. During data collection the visualization using SIMDIS can be disabled so that it is not included in the data that is collected.

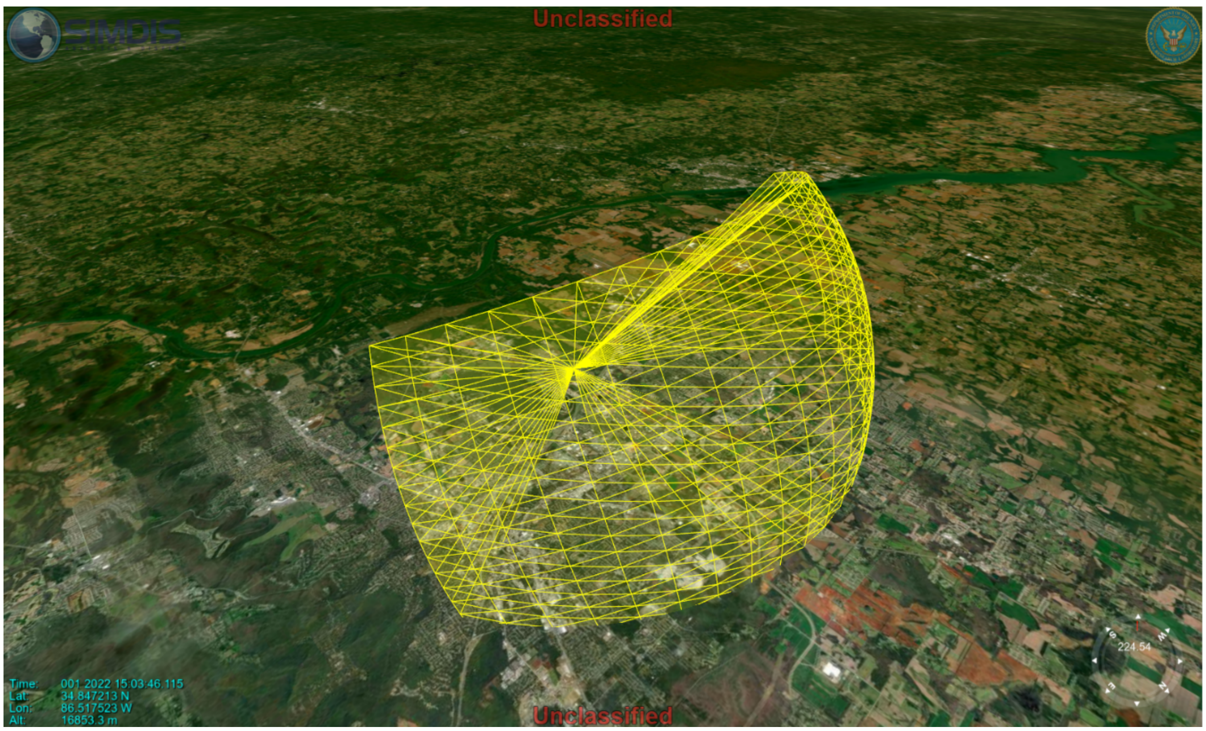

Figure 1. Image of Radar's Field of Regard in SIMDIS Visualization Environment

Components

Radar

The radar's primary role is to detect and track objects that enter its field of regard. This is done using the definition of the field of regard previously included in this paper. At each time step, the current position of each incoming ballistic threat is checked. Consider a single incoming threat. First, the Euclidean distance between the radar and that threat is calculated. If that distance is less than the configured radius that defines the max distance from the radar at which an object is visible, then the elevation angle of the threat's current position is calculated using trigonometric functions. If that angle falls within the configured elevation angle bounds, then the azimuth angle of the threats current position, relative to the orientation of the radar, is calculated. If it is determined that this azimuth angle falls within the configured azimuth bounds for the radars field of regard, then the object is visible to the radar. The current state of this object is subsequently passed from the coordinator to the radar to which it is visible.

Once the radar has received the current position of a given incoming ballistic threat, it then determines whether this is a new threat or one that it is already tracking. Currently this is being done by assigning each threat a target ID before passing it to the radar. Many systems include an additional consideration commonly referred to as identification friend or foe (IFF). As all the objects currently being simulated are either incoming threats or interceptors launched by the system it is assumed that anything that hasn't been launched by the system is a threat, and IFF is handled simply by assessing an object ID.

The other key role of the radar is to provide an uplink to launched interceptors. In many of the systems considered for this model, the radar component of the battery also transmits, via radio frequency (RF), continuously updated targeting information to the interceptor missiles. This is repeated throughout the flight of the interceptor missiles until the missile is within a certain range of the intended target. For the purposes of this model all that is communicated is an updated intercept point. Once the desired range from the target is reached, the interceptor uses the real time position of the intended target to self-guide for the remainder of the flight.

Command and Control (C2)

The C2 serves two purposes. These are launching interceptors and the handling of interceptors currently in the air. Upon receiving the current state of an incoming threat that has not been fired upon, the C2 progresses towards a potential interceptor launch in two phases, namely the discrimination phase and launch command phase. The discrimination phase involves two key decisions. The first concerns determining if a given object, within a radar's field of regard, is a threat or an asset. If this object is a threat, the second involves assessing whether it is likely to impact a defended asset. As mentioned in the previous section, a full IFF protocol is not utilized by this model. Furthermore, this assessment is undertaken by the radar, and, as such, the C2 need only check the radar's assessment.

Assuming that an incoming data point is determined to have been observed from an incoming threat, the next step is to determine its projected trajectory and, in doing so, its projected impact point. The trajectory projection algorithm used by the C2 obtains this projected trajectory by fitting small segments of ellipses to the path of the incoming threat using its observed position, velocity vectors, and acceleration vectors. This curve is then mapped to a logarithmic surface derived by characteristics observed from the threats flight, and subsequently mapped to the earth frame that the system inhabits. This yields a set of positional points intended to predict the flight of the incoming threat. Along the curve this set of points represents, the point where the height is zero is taken to be the projected impact point. If this point falls within the radius of a defended asset, then the threat is determined to be targeting that defended asset.

Once it has been determined that an object is a threat targeting a defended asset, the C2 determines that an interceptor launch is warranted and enters the launch command phase. To effectively destroy the incoming ballistic threat, the C2 must determine the optimal intercept point as well as the optimal launch time. To determine the optimal intercept point, the C2 first performs a probability of kill assessment. At each of the points previously calculated to determine the projected trajectory of the incoming threat, the C2 calculates the following:

$$\mathrm{P(kill)} = \mathrm{P(detect)P(launch)P(hit)P(destroy)} \qquad (1)$$

where P(detect) is the probability that the incoming threat is accurately detected, P(launch) is the probability that the launch of an interceptor will be successful, P(hit) is the probability that the interceptor will hit the incoming threat, and P(destroy) is the probability that the incoming threat will be intercepted in such a way as to adequately destroy it.

For this model, it is assumed that the incoming threat has been accurately detected, and that the launch of an interceptor will be successful. As such, it will always be the case that P(detect) = 1 and P(launch) = 1. The probability that the interceptor will hit the incoming threat, P(hit), is taken to be the probability that the particular point taken from the trajectory projection is accurate, and the probability that the interceptor will accurately hit that point. It is written as follows:

$$\mathrm{P(hit)} = \mathrm{P(trajectory)P(targeting)} \qquad (2)$$

The probability that the incoming threat will be destroyed in such a way as to adequately destroy it, P(destroy), is an assessment of the conditions under which the interceptor will destroy the threat. The interceptor used in this model destroys its intended target with an airburst. The intended result is that the interceptor will arrive parallel to its target and detonate in such a way that it in turn detonates the onboard warhead of its intended target. This case yields a P(destroy) = 0.9. However, there are multiple other possible cases as shown in figure 2. This model considers two others. The interceptor can be oriented towards its target with a relative angle closer to perpendicular when it detonates its airburst. This yields a P(destroy) = 0.65. In this case, it is still possible to destroy the incoming threat's onboard warhead, but this is less certain than in the first case. The interceptor can also arrive behind its target with a relative angle closer to parallel. In this case, the airburst is most likely to bisect the ballistic threat, which results in the warhead continuing on an altered trajectory, causing potential damage where it lands. This is not considered an adequate destruction of the incoming threat. There is a significantly lesser likelihood that, in this case, the onboard warhead is destroyed along with the missile. In this case, P(destroy) = 0.1.

To determine the probability of destruction, the possible impact points are considered as a set of points on a sphere of radius one with the center taken to be a positional point from the projected trajectory. Dependent on this point and the projected orientation of the threat relative to the orientation of a given launcher, the probability that the interceptor will hit each of these points on the sphere is calculated. Each of these points is weighted by the value, 0.9, 0.65, or 0.1 corresponding to which of the above cases it reflects. A counting method is then used to determine the overall probability of destruction.

The above probability of kill assessment is performed for each positional point calculated in the projected trajectory, relative to the position of each launcher. Directly following this, the C2 assesses the optimal launch time to hit each of these points from each launcher. This is done by calculating a launch delay value for each. This launch delay being the time the launcher will wait from the time that the incoming threat was observed at a given position to launch an interceptor at a potential intercept point in that threat's projected trajectory. To calculate this launch delay, the C2 uses the observed velocity of the incoming threat and integrates this over the projected trajectory up to a given potential intercept point to obtain the estimated time until the threat reaches that point. It then estimates the path the interceptor must take to reach this point as well as the velocity at each point along this path, and, similar to the incoming threat, integrates over the points in this path at these velocities to obtain the time it will take the interceptor to reach the potential intercept point. The C2 then subtracts the time until the interceptor reaches the potential intercept point from the time it will take the threat to reach the same point to obtain the time the launcher must wait before launching on this point.

This launch delay is also calculated for each point in the projected trajectory relative to each launcher. The C2 now has, for each point in the projected trajectory, relative to each launcher, a probability of kill value and a launch delay value. It selects from this set the potential intercept point with the highest probability of kill value for which the corresponding launch delay value is greater than zero and takes this point as the optimal intercept point. It selects the launcher from which this information was taken to be the optimal launcher, formulates the optimal intercept point and launch delay into a launch command, and sends this to the optimal launcher.

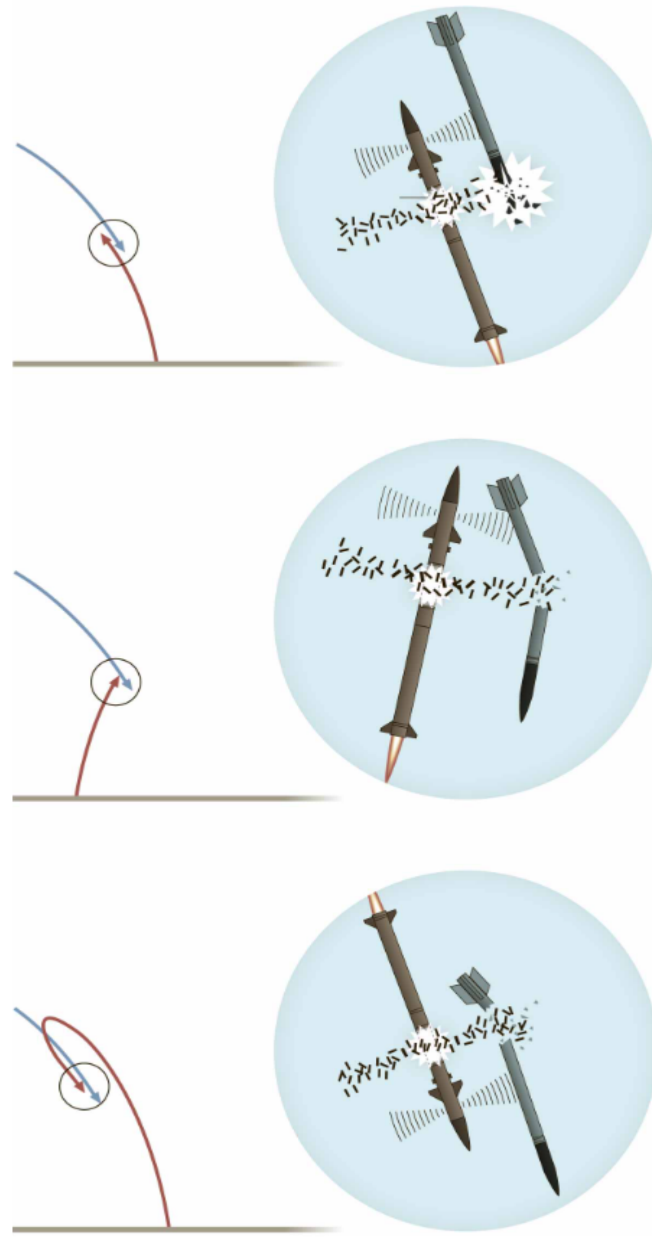

Figure 2. Image of Three P(destroy) Cases. (New York Times, 2013)

Launcher
The role of the launcher is, as the name implies, to launch interceptors. Upon receiving a launch command from the C2, this launch command is processed. The launcher executes incoming launch commands in the order of their launch delay value. Once the system time reaches the optimal launch time of a given interceptor, the interceptor is then launched to engage its target at its optimal intercept point. Additionally, the launcher also plays a role in the launcher selection occurring on the C2. Depending on the current state of the launcher it will communicate its status to the C2. The launcher has a predetermined number of interceptors onboard and available for launch before it must be reloaded. If the launcher is depleted of interceptors, or if all interceptors are already allocated for impending launches, it will communicate to the C2 that its status is empty. The C2 will then look to the other launcher's sets of optimal intercept points to find a suitable alternative if possible. If none are available, it will wait for a status of standby from the original launcher, indicating that the launcher has available interceptors, and subsequently send the original launch command. Once a launcher has exhausted all of its onboard missiles, it will then wait out its reloading time. The launcher, upon completing this reloading time, will then communicate to the C2 that its status is once again standby and that it is once again available to receive launch commands.

Data Pipeline

Networking
As mentioned previously in this paper, all components included in a given simulation form isolated nodes on a network. To facilitate network communication, the model takes advantage of Docker's MACVLAN network configuration. When a component sends a packet it checks the destination component's IP address and constructs the packet to be sent. Each packet has a header that once received is used to identify the packet and unpack the data within.

The architecture of this network is such that each radar can send and receive to and from the C2, as can each launcher. However, there is no direct communication between any radar and any launcher. To facilitate the flow of data across this network the model incorporates five packet types as shown in table 1. These are the launcher status and launch missile command mentioned above, as well as the track object, track dropped, and update targeting packets. As mentioned previously, the launcher status packet and the launch missile command packet are utilized during the process leading up to the launch of interceptors. The launcher status packet is sent at a fixed rate of 20 Hz, from the launcher to the C2. It contains the status, either standby or reloading, and a launcher ID value. The launch missile command is sent from the C2 to the launcher, only as needed, when it has been determined that a launch is warranted. It contains the launch delay in seconds, the optimal intercept point as a three-dimensional cartesian product in meters, the target ID of the incoming threat, the self ID of the interceptor, and the launcher ID of the intended launcher.

Table 1. Description of Modeled Packets

Modeled Packets					
Header	**Packet Name**	**Description**	**Source**	**Dest.**	**Rate**
0x24	Launcher Status	Status of Launcher	Launcher	C2	20Hz
0x25	Launch Missile Command	Command for Launcher to Fire Interceptor	C2	Launcher	N/A
0x2A	Track Dropped	Notification of new object in field of regard	Radar	C2	5Hz
0x2D	Track Object	Updated state of a tracked object	Radar	C2	20Hz
0x2F	Update Targeting	Update targeting data for launched interceptor	C2	Radar	20Hz

The track object and track dropped packets are both sent from a radar to the C2. The track object packet communicates the state data of a particular object as observed by the radar at a rate of 20 Hz. It contains the position, velocity, and acceleration values of a single incoming ballistic threat each as a three-dimensional cartesian product in meters, meters per second, and meters per second squared respectively. It also includes the time stamp at which this information was observed and the target ID for the incoming threat. One of these packets is sent by a given radar for each individual threat currently in the field of regard of that radar. The track dropped packet is sent at a rate of 5 Hz and is used to notify the C2 that a previously tracked threat is no longer observable by the radar that is sending this packet. It contains the target ID of the threat leaving the field of regard and the time stamp at which it was last observed by the radar. Finally, the update targeting packet is sent by the C2 and is used to send the updated targeting information to the radar to facilitate the missile uplink. It includes the self ID for the interceptor, the target ID, and the updated optimal intercept point and is sent at a rate of 20 Hz while an interceptor is in the air.

The construction of each of these packets is fairly consistent. As mentioned previously, each has a header that makes up the first eight bits of the packet and is unique to that packet type. The next eight bits give the packet length in bytes. The data being communicated in the packet requires a variable number of bits depending on the number of fields and the data type of each field. The last sixteen bits of the packet give the CRC-16 checksum value.

For an example of this in practice consider the launch missile command packet. As described above the first eight bits give the header; in this case the header is 0x25. The next eight bits give the length of the packet; in this case, the length is 384 bits. Next, taking up thirty-two bits each, is the launcher ID target ID and self ID of the outgoing interceptor. Each of these are sent as integers. Following these, taking up sixty-four bits each, are the x, y, z coordinates of the optimal intercept point as well as the launch delay, each sent as floats. Finally, the last sixteen bits gives the CRC-16 value calculated each time depending on the packet contents.

Shared Memory Communication

In addition to the network traffic between components, it is also necessary for the battery to receive and send data to and from the MATLAB Simulink model handling the calculations for incoming ballistic threats and outgoing interceptors. To facilitate this, the model employs a coordinator situated between the Simulink model and the set of radars and launchers. This coordinator handles both this data flow as well as the timing for both the model and the components in the battery. At simulation start, the model creates four shared memory namespaces for each radar and two for each launcher. For the radar, one pair of shared memory namespaces handles the sending of data and the other handles the received data. For the launcher namespace, the coordinator only ever needs to receive data. For each namespace pair, one namespace handles the transit of data, while the other is used for a handshake between the coordinator and a component to indicate that the data has been transmitted and to verify that the data has been received.

Once these namespaces have been created and connected to by their respective components, the coordinator starts the system time at the current time. It then begins receiving state data from the Simulink model. For each object being simulated it performs the checks mentioned previously to determine if the object is visible to the radar. If it is, it then maps the data from the earth reference frame to the reference frame of the radar that the object is visible to. The coordinator then uses one of the radar's shared memory namespaces to store the current state data of the object, and it uses the other to indicate that there is new data in shared memory. Once the radar in question reads the data out of the first shared memory namespace, it indicates in the other namespace that it has done so. This process happens in reverse to receive launch data from the launchers and missile uplink data from the radars. Once each of the namespaces have been checked once and the necessary data received or transmitted, the coordinator starts the next time step and repeats the process.

To communicate with the Simulink model, the coordinator uses MATLAB's engine application programming interface (API) for python. The simulation is controlled using multiple MATLAB scripts. As the Simulink model runs, it logs the states for each threat or interceptor to a mat file, and an additional MATLAB script is used to retrieve this data and feed it to the coordinator.

Example of Interceptor Launch

To further illustrate how the pieces of this model fit together, the following is an example of the process from incoming threat state data generation to the interceptor launch and subsequent destruction of the incoming threat. A full breakdown of system processes is shown in the system diagram in figure 3. Consider an incoming ballistic threat with Target ID, 1001, interior to the field of regard of Radar 1 at time step, t. The current position, orientation, velocity, and acceleration relative to the earth frame as well as the Target ID, 1001, and current time, t, is passed from the Simulink model to the coordinator. It is determined by the coordinator that the incoming threat is interior to the field of regard of Radar 1, and therefore is visible to Radar 1. The position, orientation, velocity, and acceleration are then mapped to the reference frame of Radar 1. Once this has been done, this state data in the Radar 1 frame is placed into the data transfer shared memory namespace of Radar 1 and the handshake shared memory namespace is set to the transmitted state.

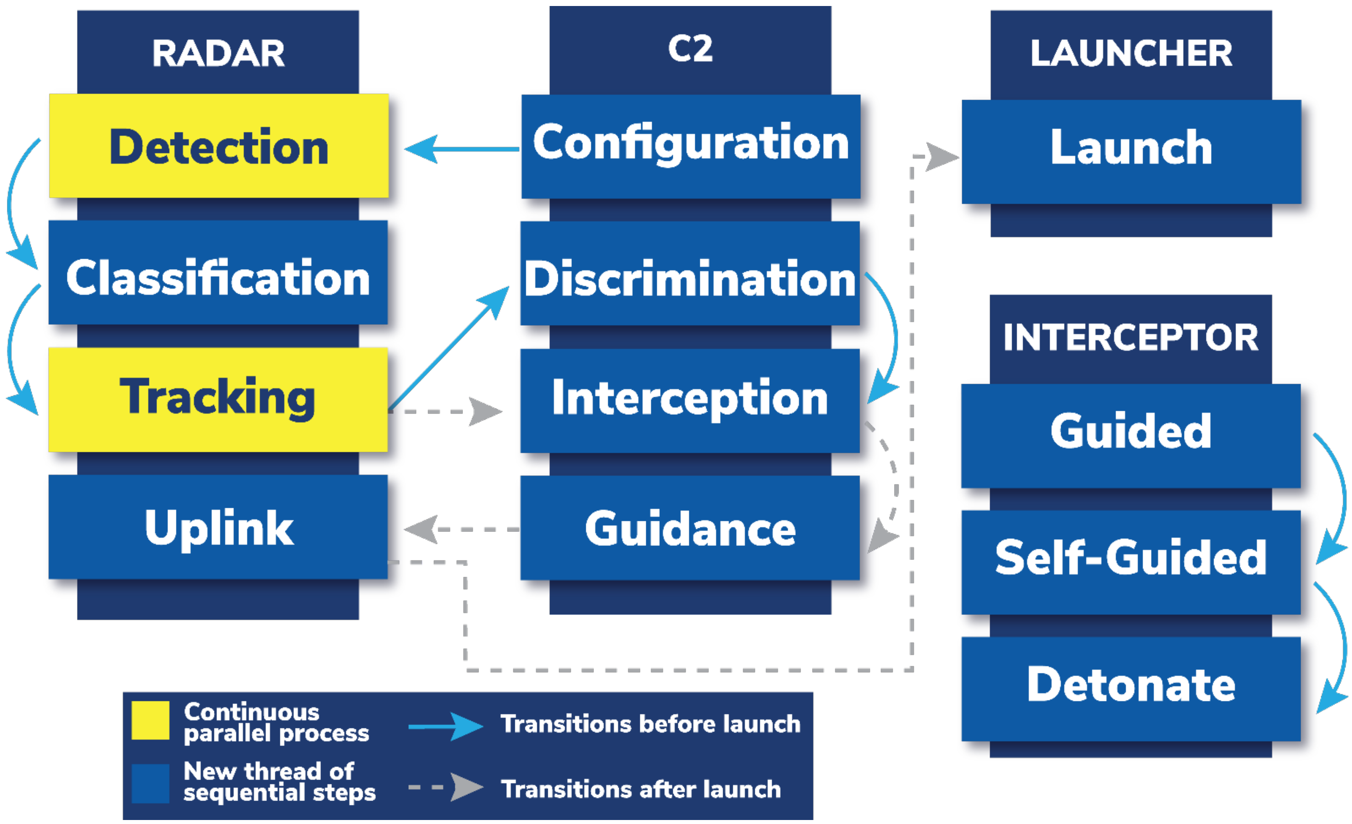

Figure 3. System Diagram Depicting Full Process Flow

Radar 1 then checks its handshake shared memory namespace and sees that it is in the transmitted state. It then reads in the state data of Threat 1001 from the data transfer shared memory namespace and sets the state of its data transfer shared memory to received, to allow subsequent data to be placed into its data transfer shared memory. Assume that Radar 1 has already observed data from Threat 1001 and is already tracking this threat. Radar 1 then prepares to transmit this state data to the C2 by creating a track object packet and assigning the state data to the pertinent fields. This packet is then transmitted from Radar 1 to the C2 unit.

Upon receiving the track object packet from Radar 1, the C2 uses the Radar ID to obtain the position and orientation of Radar 1 to be utilized in later calculations. Assume that the C2 now has enough successive state data to determine that Threat 1001 is targeting Defended Asset 1. The C2 calculates the projected trajectory and impact point of Threat 1001 and determines that the impact point for Threat 1001 falls within the radius for Defended Asset 1. The C2 now determines that the launch of Interceptor 2001 upon Threat 1001 is warranted. As such the projected trajectory calculated by the C2 is now mapped to the earth frame using the position and orientation of Radar 1 obtained previously. This projected trajectory, now in the earth frame, is used in a probability of kill and optimal launch time assessment. The optimal intercept point for Threat 1001 and the launch delay and optimal launcher for Interceptor 2001 are obtained. Let the optimal launcher be Launcher 1. The Threat ID, Interceptor Self ID, Launcher ID, optimal intercept point, and launch delay are formulated into a launch missile command packet. The C2 checks the latest status update from Launcher 1 and sees that it is on standby and as such transmits the launch missile command packet from the C2 to Launcher 1.

Launcher 1 receives the launch missile command packet from the C2. Launcher 1 then checks that it is the intended launcher by referencing the Launcher ID in the launch command packet. Upon verifying this, Launcher 1 then places the launch command data including the optimal intercept point, launch delay, Interceptor Self ID, and Target ID into the launch queue to await the optimal launch time. Once the optimal launch time is reached, Launcher 1 places the optimal intercept point, Interceptor Self ID, and Target ID into its data transfer shared memory and sets the state of its handshake shared memory to transmitted.

The Coordinator then checks the Launcher 1 handshake shared memory namespace and sees that data has been placed in the Launcher 1 data transfer shared memory. The Coordinator then reads in the launch command data from the data transfer shared memory and sets the state of the Launcher1 handshake shared memory to received. This launch command data for interceptor 2001 is then passed from the coordinator to the Simulink model using the Engine API for python. Simulink then sees that the object is an interceptor and begins generating state data using the 6DOF interceptor model to then send back to the coordinator along with the subsequent state data for Threat 1001 and the launch of Interceptor 2001 is complete.

Once interceptor 2001 has been launched, the process repeats as before to transmit state data on Threat 1001 and now Interceptor 2001 to the C2. Once it arrives in the C2 however, as Threat 1001 has already been fired upon, the process changes. As before, the C2 calculates the projected trajectory and optimal intercept point for Threat 1001 based on the updated state data. However, this data is now sent to Radar 1 using the update targeting packet. Radar 1 serves to facilitate the missile uplink by utilizing the same process as Launcher1 to transmit the updated optimal intercept point to the Coordinator and subsequently to the Simulink Model. This process repeats until Interceptor 2001 is within a preconfigured distance from Threat 1001, at which time Interceptor 2001 switches to self-guidance based on the true current position of Threat 1001. Interceptor 2001 guides itself internally for the duration of its flight until it is within range of the threat to detonate its payload, destroying the threat.

Data Collection

One of the end uses of this model is to provide more realistic data sets that provide generic operational data on the operations of a variety of real-world missile defense systems. To achieve this the simulation environment incorporates data loggers as well as packet captures (pcap). Each component in a given simulation logs its state data at each time step, as does the coordinator. Additionally, each packet transmitted on the network is captured in a pcap file.

What follows is a description of the data being logged by each component type. For all of these components and the coordinator, a time stamp is logged in addition to the other data. The radars, at each time step, log the current position, velocity, acceleration, and orientation of all incoming threats visible to it at that time. The radars will also log the IDs of all visible threats. The C2 logs the current status and number of onboard interceptors of each launcher as it is known to the C2 as well as the IDs of these launchers. The C2 also logs the position, projected impact point, and ID of each threat that it has received state data for. If an incoming threat is determined by the C2 to be targeting a defended asset, the optimal intercept point and whether or not a launch command has been sent during this time step will also be logged. The launcher, at each time step, logs its ID, current status and onboard missile count, as well as the optimal intercept point of any launch that has taken place in this time step.

The Coordinator is also employed in data logging. This is due to the fact that if there is a simulated bad actor that imposes some effect on the system, that reveals itself in the data logged from one or more components, then the Coordinator serves as the source of truth data for the interval of time during a cyber-attack. The Coordinator logs, at each time step, the current position, velocity, acceleration, orientation, and target of each threat or asset currently in the air. As this is not exhaustive to the data collected, the launchers themselves serve as the source for truth data on their respective current onboard interceptor counts. Similarly, the C2 serves as the source of truth data on whether a launch missile command was or should have been sent during a particular time step.

Simulated Cyber-Attacks

As mentioned previously, the purpose of this model is to facilitate analysis of the effects of cyber-attacks against missile defense systems. A sample malware attack has been developed and is being used to infect the components of this model during simulation. The end goal of this attack is to infect a launcher and interfere with or entirely prevent the launch of an interceptor. This allows for the collection of data from a system under attack without having to infect a physical system with malware. The data collection can then advise the improvement of the cyber-resiliency of real systems. This data can also provide the basis for testing an intrusion detection system (IDS) without having to gain access to classified information.

Throughout a simulation where the system is under attack, the malware attacks the system in three phases. During the first phase, the malware is placed on a radar, here referred to as a rogue radar. Once the malware begins its attack, the malware begins the network scanning phase wherein the rogue radar identifies the other nodes on the network as well as the vulnerabilities it can exploit to spread to these other nodes. This is currently being done by exploiting open ports and services on the various components. The malware is designed to have random timing of actions to prevent it from presenting obvious patterns during the network scanning phase. After other nodes have been identified, the malware begins the exploitation phase and subsequently the propagation phase where it exploits the component vulnerabilities to spread to the other components. Once on a new component it repeats the three phases until it reaches a launcher.

Once a launcher has been infected, here referred to as a rogue launcher, the malware's payload is activated. Upon activation of the payload the rogue launcher begins sending illegitimate launcher status packets to the C2. In these illegitimate packets the rogue launcher's onboard interceptor count is falsified such that it appears to the C2 that the launcher has zero onboard interceptors. The legitimate launcher status packets are still sent at a rate of 20 Hz. As such the rogue launcher sends illegitimate packets at a higher rate in order to flood the C2 with bad information. As mentioned previously, once the C2 determines that an interceptor launch is warranted and has calculated the optimal intercept point and optimal launch time, it confirms that the launcher has the requisite number of onboard interceptors before sending the launch missile command packet to this launcher. If the launcher has no missiles onboard it will not send the launch missile command packet to this launcher.

Depending on the scenario there are several expected outcomes of this attack. If there is only one launcher or the other launchers present are out of missiles, then this attack can result in the delay of the interceptor launch until the receipt of a legitimate packet with the correct onboard missile count coincides with the confirmation that the launcher has enough interceptors to perform a launch. Alternatively, this could delay the launch long enough that the incoming ballistic threat reaches its target, and the launch is rendered irrelevant as the defended asset has been destroyed. These results could also be reached if all launchers in the battery are infected and are falsifying their onboard missile count. Another possible result is that the other launchers in the battery are not infected and have available interceptors but are not well suited to be the launch point for an interceptor. In this case, a launcher for which the optimal intercept point for a given threat has a low probability of kill, could fire upon a threat and miss. In this case the threat will not be intercepted and may destroy a defended asset.

CONCLUSION AND FUTURE WORK

The model described in this paper provides a necessary tool for readily attainable cybersecurity analysis and data collection for missile defense systems. Modeling all aspects of a system including the physical system, the individual components and their behaviors, and the network provides a high-fidelity simulation framework that can be used to evaluate generic vulnerabilities and effects of cyber-attacks on such a system. Additionally, the model can be used to create data lakes of both systems under normal operation and systems under attack. This data can be used to develop new AI/ML algorithms for detection of cyber-attacks targeted at missile defense systems when access to a real system or real data is not possible.

The future work on this model is twofold. There exist areas where the resolution of this model can be improved. In some cases, this opens up additional attack surfaces for analysis. For example, additional simulated objects other than incoming threats and launched interceptors would allow for a more realistic IFF protocol. This is a potential attack surface as the ability to use a system to attack assets friendly to it would be of great value to a bad actor. Additionally, the continued work with this model will include expanding the attacks against its existing capabilities to provide a fuller picture of the potential vulnerabilities of missile defense systems.

REFERENCES

Lambrechts, P., Bennani, S., Looye, G., & Moormann, D. (1997). *The RCAM design challenge problem description. In Robust flight control: a design challenge* (pp. 149-179). Springer Berlin Heidelberg.

Shapiro, I. I. (1958*). The Prediction of Ballistic Missile Trajectories from Radar Observations* (Vol. 129). McGraw-Hill.

The New York Times. (2013, March 20). *Angles of attack for Israel's Iron Dome Missile System*. The New York Times. https://archive.nytimes.com/www.nytimes.com/interactive/2013/03/21/world/middleeast/angles-of-attack-for-israels-iron-dome-missile-system.html?ref=middleeast

Nichols, J. O. (1977). *Analysis and compilation of missile aerodynamic data. Volume 1: Data presentation and analysis* (No. NASA-CR-2835). NASA.

Cancian, T. K. and M., Karako, T., Cancian, M., & Gluck, M. (2023, May 25). *Patriot*. Missile Threat. https://missilethreat.csis.org/system/patriot/

Pike, J. (n.d.). *Identification Friend or Foe*. Global Security. https://www.globalsecurity.org/military/systems/aircraft/systems/iff.htm

Pike, J. (n.d.-b). *Patriot Air Defense System*. Global Security. https://www.globalsecurity.org/military/library/policy/army/fm/3-01-11/ch5.htm#:~:text=FIRE%20UNIT%20TARGET%20ENGAGEMENT%20SEQUENCE