MP 180677
MITRE PRODUCT

ECHO – Estimated Cyber Hardness by Observation

Project No.: 10AOH810-JF

Lexington Park, MD

Authors:

F. Lynam
F. DiBonaventuro
M. Mickelson
L. Oringer
S. Ricks

Peer Review

R. McInnes
M. McFarren
K. Kressel

October 2019

Abstract

This paper describes the basic elements of the ECHO model, which is the cyber intrusion and vulnerability model used for Trace analysis. The ECHO model is a simulation-focused formalization of other existing threat models, drawing from sources such as MITRE’s ATT&CK and the NSA’s Technical Cyber Threat Framework. A detailed derivation and sourcing of the pieces of the model are provided, as well as a detailed description of how to consider and apply the ECHO model as tailored to particular systems of interest. In practice, much of the ECHO model application is automated by Trace tools, however, an understanding of the model is an important aide to analysts intending to apply the model to a system under assessment. This paper argues that all potential vulnerabilities in any programmable logic system should be able to be fully described by the presented generalized model, and provides supporting discussion and references.

Overview

This paper provides technical details on an approach intended to exhaustively describe all possible adversarial network traversal and asset impact effects within a consistent, generally-applicable process flow model: the ECHO (Estimated Cyber Hardness by Observation) function. This approach is used to support analysis estimating the probability of executing any one process element. Use of this model within the Trace (Traversal-driven Risk Assessment of Composite Effects) analysis framework allows for an evidence-based approach to evaluating cyber risk; that is, constructing quantitative risk measures directly from how the system architecture aligns with the full universe of exploitable pre-existing flaws. This paper focuses primarily on the theoretical basis of the traversal model, its constituent elements and its necessary assumptions. Many of the caveats and tangential considerations identified here are addressed in other parts of the overall Trace / ECHO implementation, but are noted so that this piece of the puzzle can be used in other analyses if desired.

Cyber Weapon Analysis

The ECHO traversal model grew out of the need to anticipate high-capability, nation-state cyber activities that leverage zero-day (previously unknown) vulnerabilities and previously unseen tactics. The outputs provided by ECHO are built on observations from empirical data on the upper limits of technological possibilities, vice making any limiting assumptions about tactics or adversarial decision making. This paper attempts to capture certain assumptions the authors consider reasonable within their experience and subject matter expertise, however, for systems or scenarios where these assumptions are considered over-conservative (over-stating the resulting risk), additional analysis to better characterize other relevant dimensions of the problem may be warranted.

This paper introduces the necessary basic concepts to understand the intent and nature of the ECHO traversal model, followed by a deep discussion of the specific content of the model itself and provides brief notes about assumptions and necessary context. A complete listing of the threat concept catalog with detailed descriptions is included at the end of the paper.

Fundamental Observations

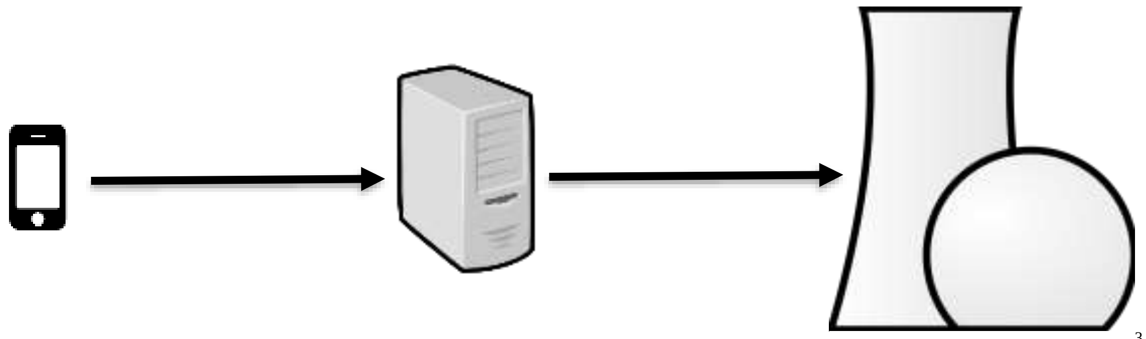
It has previously been observed that vulnerability discovery rates converge on a mean rate of approximately one month between vulnerabilities of consequence in a large-scale developmental test environment (after an initial "infant mortality phase").[1] While theoretical models could be envisioned which provide an elegant explanation for this phenomenon, this observation stands regardless of its cause. Should future data indicate a more complex relationship between resourcing and discovery rate, such a function should be easily accommodated without substantive change to ECHO.

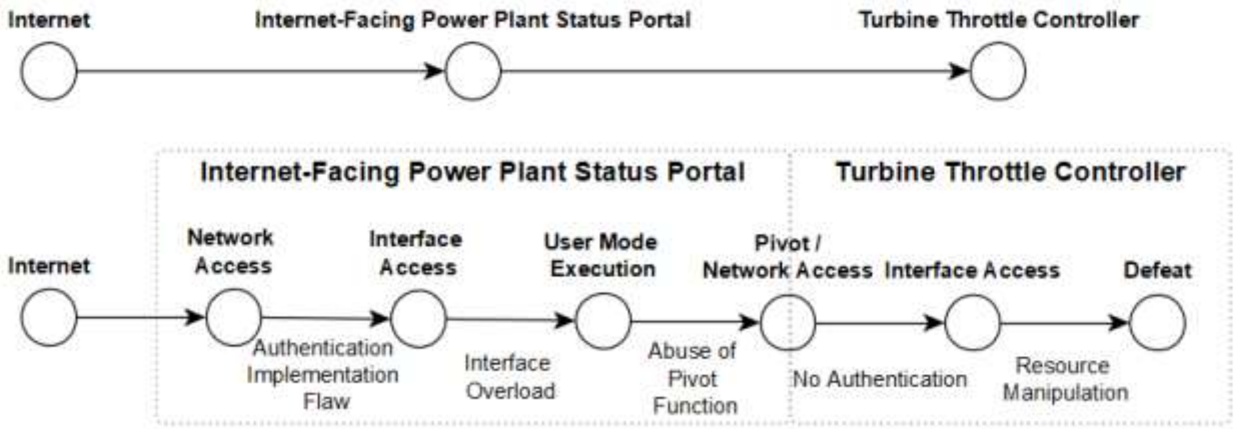
The analysis performed in development of ECHO suggests that among these vulnerabilities, the same flavors show up in the same proportions over time regardless of the software or logical system in play; there is some distribution of types of vulnerabilities that has some degree of consistency over time and technology. These "flavors" and their functional relationship are the focus of this paper.

[1] M. McQueen, W. Boyer, M. Flynn, G. Beitel, "Time-To-Compromise Model For Cyber Risk Reduction Estimation," Quality of Protection Workshop, ESORICS, Idaho National Laboratory, September 2005

Combining the predictability of vulnerability discovery rates with the predictability of vulnerability type distributions allows ECHO to quantify the time (or work factor) associated with the discovery and subsequent exploitation of any useful vulnerability in a system. These factors can be expressed as probabilities and combined into quantifications of end-to-end weapons. This allows ECHO to provide a time-varying function for probability of exploitation of any single vulnerability, facilitating the determination of the time (or work factor) associated with some chain of effects compromising a larger system's mission or objective when applied in a topological vulnerability analysis.[2]

For example, consider an attack on a power plant which causes a turbine to operate at unsafe speeds:


[3]



In this example, an internet-facing power plant status portal secured for use only by authorized personnel is compromised by manipulating a hash collision in the outdated hashing function used to secure user accounts. This exploit allows a malicious user access to the portal, which they then exploit to execute arbitrary code via some code-injection technique. That attack then leverages some built-in ability of that server intended to allow users to request status of plant systems. Instead of a status request, the malicious user crafts a command message to a throttle controller to open the throttle to full. The throttle controller, as in many industrial control systems, is designed to respond to those commands: that's how the throttle is normally commanded, via messages over the network. This results in an overspeed of the turbine, at the very least tripping it off via some out-of-band safety feature, but potentially permanently damaging it or harming personnel in the vicinity.

[2] S. Noel, M. Elder, S. Jajodia, et al, "Advances in Topological Vulnerability Analysis," Conference for Homeland Security, 2009

[3] Images used under Creative Commons license

All of the specific vulnerabilities in that story are special cases of the general concepts within the ECHO threat concept catalog, and they mesh together into a chain of events via application of the ECHO traversal model. Being able to describe those ideas in a general manner via ECHO allows for the prediction of similar vulnerabilities that leverage the same basic ideas, but that may manifest in slightly different ways. Maybe the initial access is attained because passwords are passed in the clear during the login process. Maybe the web server HTTP GET message receipt is overloaded instead of the portal scripts. Each of these point vulnerabilities could be resolved if they were publicly known, but just because they haven't been discovered or disclosed yet doesn't mean they weren't already there the whole time. ECHO provides a means to explicitly measure and account for the residual risk of someone exploiting vulnerabilities we don't know about, which are the very ones we'd expect our greatest adversaries to exploit in a time of crisis.

By creating a model defined around automated construction of effects chains, the complete potential space of cyber weapons can be exhaustively described, and the relative credibility or adversary work factor of exploiting those effects can be explicitly quantified. In practice, this state space has been found to be so large that manual path construction is insufficient to meaningfully represent the cyber attack surface.

Further research should be performed to better characterize potential variance of mean time to vulnerability discovery as a function of the underlying technology or resourcing level. Additionally, it should be explored more thoroughly whether different systems or system features lend themselves to different distributions of threat concepts. Theoretical explanations for these observable phenomena should be explored and tested. For example, this phenomena would be consistent with a chaotic distribution of vulnerabilities across the undecidable space of a Turing machine, where factors which increase the rate of vulnerability discovery could be viewed as simply reducing the complexity of completing partial solutions of the halting problem. The method contained herein for estimating objective cyber risk via evidence-driven analysis should continue to evolve as a better understanding under of the fundamental effects at work is developed.

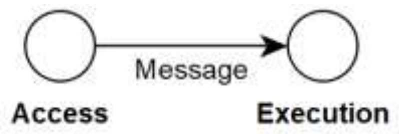
Adversarial Network Traversal

A literature review of cyber analysis models and cyber attack case studies was performed to establish a common set of generalized pieces of attacks, referred to here as cyber threat concepts. These concepts are intended to encompass any technique or tactic that could be effectively employed by an adversary to achieve both near-term and long-term cyber offensive objectives. These were then incorporated into a model for treatment as event categories in a follow-on data analysis. The integrated process flow which includes each of these threat concepts is described here. Notes on sources and details of the threat concepts themselves are included at the end of this paper.

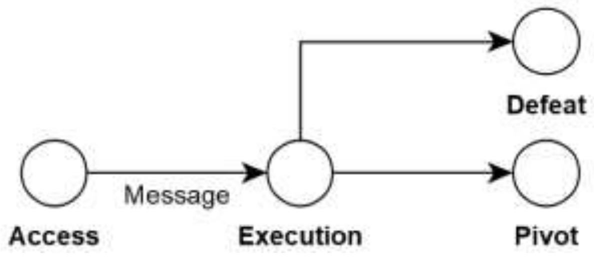
Fundamental Model of Adversarial Cyber Interaction

The interaction at a logical interface with a cyber device is fully described by the implementation of that device. However, a complete model of these systems often does not exist, particularly for large, complex or legacy systems. It would require an exhaustive model not just of the software in use, but of the libraries used by that software, of the operating system those libraries interface with, of the chip-level machine code executed by that operating system, and of the transistor-level logic within that processor. This level of fidelity, while theoretically attainable, is not feasible. ECHO provides a means to integrate those “cyber quanta” at a useful level of emergent properties.[4]

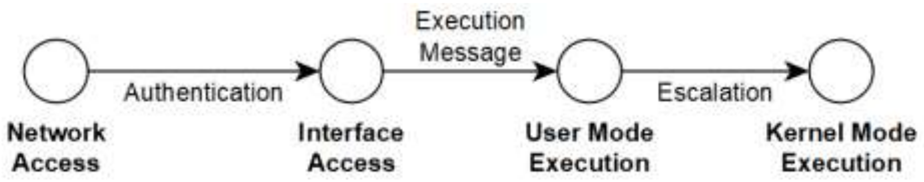
Cyber interactions can be fully described by a communication model, where a message is received and some logically-driven process ensues. That message can be a set of symbols input by a human operator, or information transmitted over a network, or sound converted into data via a microphone. This interaction is modeled by ECHO as traversal through a simple process flow. The process starts with some actor having a presence external to the device in question, represented by “access.” It ends with that actor having an effect internal to the device, represented by “execution.” That transition from being on the network to taking control is made through the process sending a message in this fundamental cyber interaction model:



Notionally, this model completely bounds all possible cyber attacks, as all potential attack paths can be exhaustively described through simple application of this model to all devices in a network. Adding the ability to connect these together, either to pivot deeper into a network or to defeat a particular function, allows this fundamental concept to facilitate the development of attack paths for use in topological analysis:



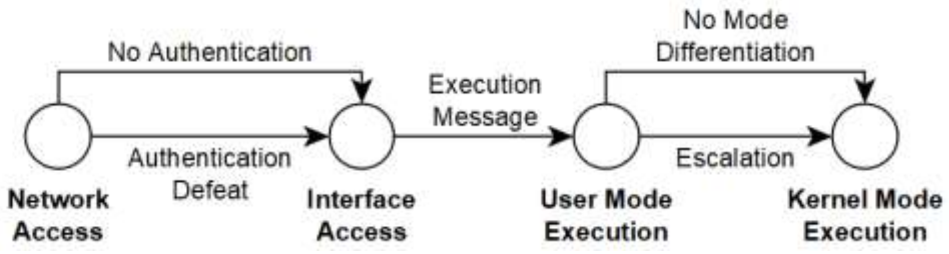
However, this level of abstraction is insufficient to provide meaningful insight into mitigation options or to provide reasonable estimates of residual risk in an actionable manner. Based on the ECHO threat concepts, this basic interaction has been broken into a more detailed and useful generalized model. To understand that model, we’ll first focus on the internal effect states and ignore the external connections for the moment:



This model now contains all of the internal effect states found in the final model. The shown states break the idea of “access” into the difference between the ability to talk to a device (network access) and the device actually listening (interface access), and break “execution” into the difference between

[4] K. Jabbour, S. Muccio, “The Science of Mission Assurance,” Air Force Research Laboratory, Journal of Strategic Security 4, no. 2 (Summer 2011)

the device doing what a normal process is allowed to do (user mode execution) and the device doing whatever you say (kernel mode execution). The shown transitions demonstrate some common obstacles in a cyber effect process. Not all interfaces accept messages from just anyone talking to them (authentication), and not all devices will perform any arbitrary function in response to an external command (escalation). The presence or absence of these functional features needs to be accounted for by the analysis:



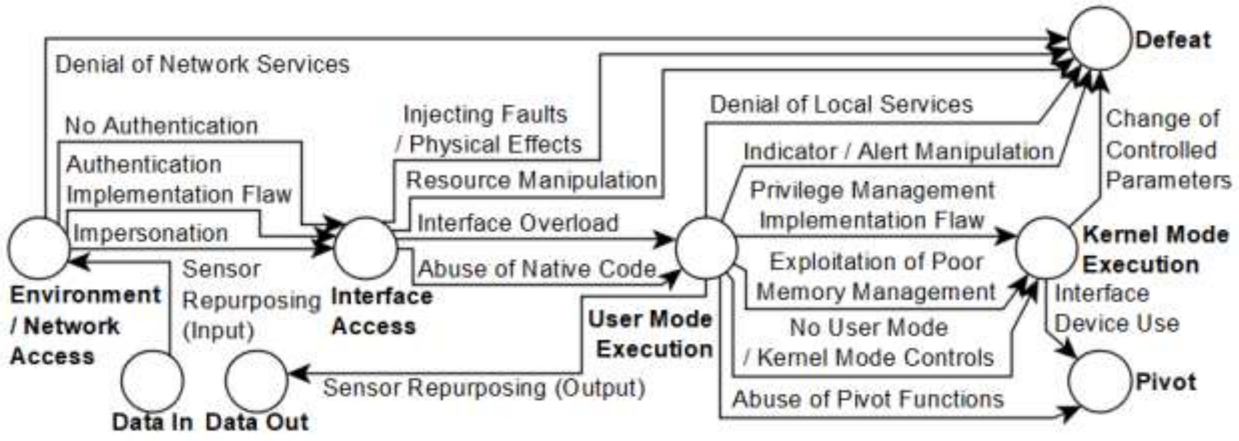
This model provides for multiple potential state transition paths. Not all of these paths apply to every device; if a system has no authentication, there is no need to defeat the authentication. Part of the ECHO assessment process trims out these inapplicable paths by asking basic system engineering questions about the presence or absence of these functions: Is there interface authentication? Is there user / kernel mode differentiation? The answers to these relatively simple questions provides critical information for understanding the time involved in leveling cyber effects on the component in question.

For example, often a typical embedded system such as a motor controller may require no authentication and have no user / kernel mode differentiation, happily turning on and off a motor based on any message traffic that says to do so that happens to pass by its interface. In such a case, the probability of traversing between the network access and interface access states in the overall traversal diagram is 100%. However, if authentication is provided and no known vulnerabilities or exploits exist, the maximum probability of successful traversal between states can theoretically be described as a function of the respective vulnerability discovery rate for the underlying technology and the time invested in discovering and developing a novel vulnerability and exploit.

A key objective of ECHO is to retain the complete bounding provided by the fundamental cyber interaction model shown above, even as additional layers are added to the process flow. Above, that is accomplished for the authentication transition because the two paths “No Authentication” and “Authentication Defeat” exhaustively describe all the possible transition modes between the first and second states with no overlap. The threat concept catalog was designed to provide exhaustive coverage of all possible transitions in this way.

The ECHO Traversal Model

This section provides a detailed discussion of the ECHO traversal model, depicted in its entirety here:


Defeat
Denial of Network Services
No Authentication
Authentication Implementation Flaw
Impersonation
Injecting Faults / Physical Effects
Resource Manipulation
Interface Overload
Abuse of Native Code
Denial of Local Services
Indicator / Alert Manipulation
Change of Controlled Parameters
Privilege Management Implementation Flaw
Exploitation of Poor Memory Management
No User Mode / Kernel Mode Controls
Abuse of Pivot Functions
Kernel Mode Execution
Interface Device Use
Pivot
User Mode Execution
Environment / Network Access
Sensor Repurposing (Input)
Interface Access
Sensor Repurposing (Output)
Data In
Data Out

The ECHO traversal model consists of various intermediate states of compromise connected in a process flow via threat concepts. It is solely the design and intended use of the underlying technology which dictates applicability of any individual threat concepts to a particular system under assessment. This section touches on each state in the complete traversal model, focusing on states with outgoing threat concepts and discussing the criteria for why and how each outgoing threat concept applies to a particular asset or interface and what differentiates them.

The threat concepts are designed to be mutually exclusive groupings, which together provide exhaustive coverage of the set of all possible vulnerabilities. They are primarily differentiated by two factors: whether they are abusing an intentional or unintentional design feature, and which state they facilitate transition to. Intentional design features are capabilities built into a system by the designers to accomplish a desired task, and sometimes these may be subverted or abused for other intents. Unintentional design features are flaws in the construction of a system, gaps in how it’s built or assembled which may cause the system to function in potentially novel ways never originally envisioned by the designers.

The concept of negative requirements helps to illustrate the idea of unintentional design features. A device may have a requirement such as “receive two one-digit numbers, add them and output the result.” It does not have the negative requirement “reject two-digit numbers.” A system meeting the requirement may be designed, tested and operated without any two-digit inputs ever showing up. However, without the negative requirement, if an adversary provides a two-digit input, the system may fail in an unpredictable way. Formal design methods, like the methods used to design a bridge, are a tool that can be used in software design as well to demonstrate that all possible negative requirements, like “nothing except one-digit inputs,” are safely handled. Here, the term “formal verification” will be used as a placeholder for either formal methods or any other mechanism which may be devised that can exhaustively prove the completeness of coverage over all negative requirements.

Environment / Network Access

Like any cyber attack path, the ECHO traversal model starts with an adversary having access to some data interchange that the asset under assessment collects data from. That might be the surrounding environment, such as ultrasonic information received by a microphone, or from a connected network, such as an embedded serial bus. From here, threat concepts lead to three further states: Defeat, Data In and Interface Access.

The **Defeat** state is the state where some functionality that the asset provides is no longer able to be executed. This can sometimes be attained with nothing more than access to a network simply by denying that access by leveraging intentional or unintentional design features, such as through flooding. Here, that idea is captured by the *Denial of Network Services* threat concept.

The **Data In** state is the state where some pre-planted payload receives some information, such as command and control data, through some channel being used by that payload in an intended or unintended manner, such as an accelerometer being actively polled in order to be used as a microphone. This is reached via *Sensor Repurposing*, even if that sensor is something normally not considered a "sensor" being subverted in a manner not necessarily "repurposing" it entirely, such as a network interface card listening on an innocuous-seeming port.

The **Interface Access** state is the state where the intended device logic (not a pre-planted payload) is actively interpreting and listening to incoming messages. This is reached naturally in systems without authentication schemes, captured here with the *No Authentication* threat concept. Where authentication is in place, it can be defeated by leveraging intended functionality via the Impersonation threat concept and by leveraging unintended functionality via the *Authentication Implementation Flaw* threat concept.

Given an interface with an authentication mechanism, unless that mechanism has been formally verified to lack any exploitable flaws, some flaw (such as a timing or state dependency) may exist which can be abused to compromise the integrity of the mechanism. If no such verification has been performed, the interface is susceptible to an *Authentication Implementation Flaw*.

In contrast, *Impersonation* relies on obtaining shared secrets through some means. For example, methods exist to estimate the computational time necessary to brute force forged certificates or otherwise cryptographically break different encryption and authentication schemes. Alternatively, authenticating secrets could be obtained by a priori knowledge, such as compromise of a SSH key repository for a network using a common SSH infrastructure. Computation of the former is not discussed here, but as a time-dependent probability, the result could certainly be easily incorporated into this analysis. The latter must be addressed in the analysis via an integrated understanding of the authentication infrastructure for a given interface. To this end, the interface is susceptible to Impersonation if it is credible to assume that the adversary has obtained appropriate compromised credentials through phishing or other means.

Interface Access

The **Interface Access** state is the progression of interaction with a device to the point where the device is actively interpreting, parsing or responding to incoming data. Four outgoing threat concepts apply: *Injecting Faults* and *Resource Manipulation* to achieve the **Defeat** state or *Interface Overload* and *Abuse of Native Code* to achieve the **User Level Execution** state.

Injecting Faults is inputting data that is structured in a manner which causes the as-built software to fail. *Interface Overload* is a code injection attack leveraging some unforeseen gap in the intended functionality, such as SQL injection. If an interface has not been formally verified to be robust to invalid inputs across all obtainable states, it is susceptible to both *Interface Overload* and *Injecting Faults*.

Resource Manipulation is abusing the trust in some network resource to insert falsified data to elicit an anticipated response, such as an overtemperature alarm in a propulsion system which executes a safety shutdown at an inopportune moment. If an interface relies on data from an external network resource, it is susceptible to *Resource Manipulation*.

Abuse of Native Code is using intended **User Mode Execution** functionality designed to be accessible from the interface, such as JavaScript or SNMP firmware updates, to perform some allowed executable function on the system. If an asset interface contains allowed states which can execute code, then it is susceptible to *Abuse of Native Code*.

User Mode Execution

User Mode Execution is the state of progression of device compromise where an outside actor is able to execute arbitrary code with limited or otherwise restricted privileges in some protected mode on a device. From **User Mode Execution**, paths exist to achieve the **Defeat**, **Kernel Mode Execution**, **Pivot** and **Data Out** states.

The asset is susceptible to **Defeat** via *Denial of Local Services* if local computing resources are necessary for critical functions the device provides. The asset is also susceptible to **Defeat** via *Indicator and Alert Manipulation* if critical functions could be undermined by providing operator indications which are not representative of the underlaying data or system state. **Defeat** via *Indicator and Alert Manipulation* is intended here to be explicitly different from **Defeat** via *Resource Manipulation*, which is limited to falsified data on an external data bus. While the operator and operational effects may be the same in some cases, these threat concepts are importantly distinct in their technical dimension. Further, it may be possible to generate fake alerts or indications (e.g. “skull and cross bones”) that prompt desired operator actions that are not possible to generate by simply falsifying bus data.

If the **User Mode Execution** is designed to allow for interaction with external interfaces, then the system is susceptible to abuse of those functions to pivot deeper into the network without a need to escalate to kernel mode. Likewise, if the user mode has authority to control external data systems, such as speakers or radios, then the **Data Out** state can also be attained without the need for kernel mode access.

Transition from user mode to kernel mode is captured via three paths: *No User Mode / Kernel Mode Controls*, *Exploitation of Poor Memory Management* and *Privilege Management Implementation Flaw*. The lack of user mode / kernel mode controls provides free motion, and applies in systems without such differentiation. *Exploitation of Poor Memory Management* and *Privilege Management Implementation Flaws* are both abuses of unintended flaws in aspects of the design of the system, and relate to formal verification. *Exploitation of Poor Memory Management* is applicable solely in cases where the underlying operating system and processing infrastructure lack formal verification. *Privilege Management Implementation Flaws* applies where the specific implementation of the mechanism used to segregate between the execution modes lacks formal verification.

Kernel Mode Execution

Upon attaining **Kernel Mode Execution**, two paths remain: the ability to **Pivot** via *Interface Device Use* or **Defeat** via *Change of Controlled Parameters*. By definition, kernel mode privileges constitute arbitrary control of system devices, such as network interfaces, and allow for generation of effectively arbitrary output on those devices (within the physical constraints of the device design). Any device which is able to output data to a network is vulnerable to this threat concept upon kernel mode escalation: the ability to pivot deeper is a given once **Kernel Mode Execution** is possible. Likewise, in some cases, key operating parameters of a system or network are protected by kernel mode controls and the system is therefore vulnerable to defeat upon kernel mode escalation: the ability to defeat any critical functions this asset provides is now a given.

Summary of Edge Applicability

The following table provides a summary of the threat concepts in the ECHO traversal model and shows how to rapidly identify their applicability to a particular asset or interface. Some of these questions may be easily resolved by understanding the overall system network architecture. Others may require understanding essential fundamentals about the technology or functionality of the asset. Functional relationships indicated here can be replaced with functional dependency models incorporated into a larger analysis if so desired.

#	Question	Applicable Threat Concept if Yes
	These questions will likely be answered by the system architecture	
1	Does the asset output data on any interfaces?	Interface Device Use
2	Does the asset perform a critical function?	Change of Controlled Parameters
	These questions are about how the asset interfaces with the outside world, and should be asked for each interface	
3	Is this an interface to the physical environment (speakers, RTDs, microphones, pumps, etc)? **If no, sub-parts are no.**	
3.a	Can this interface receive data from any of these systems?	Sensor Repurposing (Input)
3.b	Can this interface send data (control or otherwise) to these systems?	Sensor Repurposing (Output)
4	Does this interface transmit or receive data as part of a function of this asset? **If no, sub-parts are no.**	
4.a	Could loss of timely access to that data (or ability to transmit it) result in functional failure?	Denial of Network Services
4.b	Could the asset respond undesirably if fed false data from a trusted network resource on this interface?	Resource Manipulation
4.c	Could critical functions be undermined by providing operators indications which are not representative of the underlaying data or system state?	Indicator / Alert Manipulation
5	Does this interface lack any authentication mechanism? **If yes, sub-parts are no.**	No Authentication
5.a	Has the specific authentication implementation not been formally verified?	Authentication Implementation Flaw
5.b	Is it credible that the adversary has obtained appropriate compromised credentials?	Impersonation
6	Does the asset normally execute code contained in input data, such as loading a web page or parsing an SQL query? **If yes, sub-parts are yes.**	Abuse of Native Code
6.a	Does the asset data ingestion path lack formal verification that it contains no paths which execute code?	Injecting Faults, Interface Overload
	These questions are about how the asset functions, and should be asked once for each asset	
7	Are local computing resources necessary for critical functions the device provides?	Denial of Local Services
8	Does the system have no privilege differentiation, such as between user and kernel mode? **If yes, sub-parts are no.**	No User Mode / Kernel Mode Differentiation
8.a	Does the operating system and processing platform lack formal verification of the memory management scheme?	Exploitation of Poor Memory Management
8.b	Does the specific privilege control implementation lack formal verification?	Privilege Management Implementation Flaw
8.c	Is the user mode execution designed to allow interaction with external network interfaces?	Abuse of Pivot Functions

Simplifying Assumptions

To allow application of the observations supporting ECHO consistently via model-based analysis, this paper makes a number of contextual assumptions to allow a focus on describing the totality of the problem space. This problem space can the be reduced or trimmed as appropriate by other analyses. These assumptions are discussed here.

There are two fundamental models of adversary approach to a cyber attack: active intrusion and cyber weapons. Active intrusion involves real-time adversarial decision making during execution of an attack path, and can be broken into two further modes: use of an existing toolkit or active exploit development in real time. A cyber weapon is a pre-planned attack, where each step in the attack path is identified, developed and tested before an attack is ever launched.

The cyber weapon model is intended for consideration against highly planned and targeted cyber effects from high-capability adversaries, and conservatively bounds the risk posed by an active intrusion, or access-based model of risk. Because the definition of a preplanned attack is that an adversary has sufficient time to plan for better decision-making than possible during an active intrusion, this model may overstate the risks in an access-based analysis. Modeling of active intrusions is, however, unnecessary if planned cyber weapon risk can be satisfactorily accounted for. If residual risk of cyber weapons cannot be managed to an acceptable level, further analysis specifically of active, access-based intrusions would be a valuable risk reduction measure.

Fundamentally, the analysis method is not meant to simulate an adversary’s development cycle. Instead, it’s meant to be able to measure the limits of that development cycle across the entire attack graph. That lifecycle has been expanded in many forms, but can be generalized into three distinct stages: reconnaissance, development and deployment. For ECHO, this lifecycle is simplified to solely expressing limitations on the minimum length of the development stage. A high capability adversary is assumed to have sufficient resources to perform adequate reconnaissance on the design of the target system as necessary to develop a cyber weapon. Likewise, a high capability adversary is assumed to have sufficient resources to provide adequate testing to demonstrate that any defensive measures (technical or operational) in the design of the target are able to be evaded and that the weapon will be effective (pending changes to the design of the target system). Development time for effects such as path traversal for payload implantation, data exfiltration, command and control or remote triggering can be estimated using ECHO given appropriate information regarding the effect path.

The target under assessment is assumed to be targeted by the adversary. A risk transference approach to security is often considered credible in the commercial sector, where sufficient security is measured simply in terms of not being the “softest” target. For adversarial and combat interactions at either the strategic or even tactical level, any risk transferred away from the system under assessment is transferred to another critical campaign function. Such risk transference is not a viable solution to cooperatively secure mission objectives across an integrated battlespace.

Target under assessment is assumed to be the subject of a full investment at the complete capability of the adversary. While it may seem intuitive that mean vulnerability discovery time (and consequently weapon development time) would be proportional to applied resources, as cited above, observable evidence suggests that after some upper limit, the time taken is invariant with additional resources. Adversary investment at that upper limit is assumed by this analysis. Consequently, the probability of a successful weapon development is primarily a function of time and technology for a given adversary tier.

This analysis is solely constrained to measuring the development time for an effective cyber weapon which is expected to successfully execute an effect. Analysis to assess the probability of the insertion of a weapon into a supply chain or the probability of compromising an insider should be addressed separately and those probabilities provided as an input to topological analysis.

Relevant Assumptions for Path Analysis

The adversary should be assumed to develop all effects in parallel. The probability of developing a weapon which can execute across a path is the Boolean AND of the probability of successful discover and development of each effect in that path.

End effects must be defined by some outside method. ECHO provides a threat concept catalog meant to bound the technical aspects of cyber effects, but does not necessarily bound the human or operational impact of those effects. Some other analysis is necessary to exhaustively define that space and connect those mission impacts to the technical effects defined by the threat concept catalog. The path analysis mechanism should ensure the ability to account for all sets of paths which reach each end effect, as well as all forking path weapons requiring multiple concurrent end effects.

Path analysis should not be constrained to single paths. Notionally, an optimized investment strategy for weapon development would involve simultaneous investment in multiple credible paths, and an effective weapon would be able to employ multiple techniques to maximize probability of success (as observed in STUXNET).[5]

Common mode vulnerabilities must be handled. If two similar systems are in series in a single weapon path, the ability to defeat one of those systems is also the ability to defeat the other. The path analysis mechanism must account for common modes of vulnerability when using ECHO to compute a particular path.

Of particular note, this analysis is focused on understanding the risk imposed by unaddressed software flaws not yet known to the system owner, also known as zero-day vulnerabilities. However, the residual risk of un-remediated known vulnerabilities can be explicitly determined by setting the appropriate state transition probability to 100% and finding the marginal change to overall risk due to zero-day effects elsewhere in the system under assessment. Likewise, the residual risk of including functionality clearly subject to potential abuse can be characterized in a similar fashion.

[5] N. Falliere, L. Murchu, E. Chien, “W32.Stuxnet Dossier,” Symantec Security Response, Version 1.4, February 2011

Essential Modeling Considerations

The traversal model is described here as a “process flow.” It’s important to remember that cyber weapon development is really an agent-driven process, someone is making choices about which vulnerabilities to invest in discovering and compiling together into a weapon. The ECHO traversal model is designed to integrate with the Trace process, which makes no attempt to decide which vulnerabilities might be invested in. ECHO produces a model that isn’t a decision process at all, in fact the options available might all sum to an expectation value greater than 1. The ECHO traversal model simply describes what all the vulnerabilities someone could invest in look like. This allows a Trace analysis to cover the entire space of possible weapon paths, and describe how likely any one of those might be to succeed.

Also important to understanding the modeling of cyber weapon paths is an idea called “monotonically increasing intrusion depth.”[6] This is the idea that a weapon doesn’t need to attack boxes it has already compromised; that well-orchestrated attack paths won’t backtrack or crisscross themselves. This means that the weapon process itself cares about what its past states are, and cannot be treated as “memoryless.” In contrast, other process models such as a Markov process are only appropriate for situations which are truly memoryless, where a future step doesn’t care about any past steps. It’s essential to employ a process model that’s representative of our understanding of reality.

The ECHO Threat Concept Catalog

Each of the threat concepts in the traversal model are listed here in a simplified tabled, followed by detailed descriptions. Each threat concept is provided a simple, one-sentence definition and some discussion and examples. In the research performed, a large number of more nuanced threat concepts were also developed, but pending further research, these have been conceptually grouped into the simplified model presented in this paper.

[6] P. Ammann, D. Wijesekera, S. Kaushik, “Scalable, graph-based network vulnerability analysis,” 9th ACM Conference on Computer and Communications Security (CCS 2002), Washington DC, USA, November 2002

#	Threat Concept	Brief Description
1	Abuse of Native Code	Use intentional design features to inject code
2	Abuse of Pivot Functions	Use intentional design features to pivot
3	Authentication Implementation Flaw	Exploit a design flaw in how a system manages authentication to bypass the need for credentials
4	Change of Controlled Parameters (Free)	Modify the operation of a system to undermine a critical function
5	Denial of Local Services	Disrupt the internal resources, but not the network
6	Denial of Network Services	Disrupt the network, but not the internal resources
7	Exploitation of Poor Memory Management	Exploit programming errors to change system memory
8	Impersonation (Sometimes Free)	Use valid credential to authenticate a message
9	Indicator / Alert Manipulation	Falsify indications via local malicious code
10	Injecting Faults	Falsify network traffic to induce a predictable response not intended to be there
11	Interface Device Use (Free)	Send messages out
12	Interface Overload	Input data an interface is designed to handle, but that it handles in an unexpected manner
13	No Authentication (Free)	An interface doesn't authenticate messages
14	No User Mode / Kernel Mode Differentiation (Free)	A system doesn't have privilege controls
15	Privilege Management Implementation Flaw	Exploit a design flaw in how a system manages privileges to escalate to kernel mode
16	Resource Manipulation	Falsify network traffic to induce a predictable, programmed response
17	Sensor Repurposing (Input, Free)	Use sensors for data insertion, command and control
18	Sensor Repurposing (Output, Free)	Use sensors for data exfiltration

Notes on Catalog Development

The following sources were used, among others, in the development of the content of this catalog. Relevant examples from several of these sources are provided where possible.

Abbreviation	Data Source	Organization
CAPEC	Common Attack Pattern Enumeration and Classification	MITRE
ATT&CK	Adversarial Tactics, Techniques and Common Knowledge	MITRE
ATT&CK[7]	NSA/CSS Technical Cyber Threat Framework	NSA
CAR	Cyber Analytics Repository	MITRE
AFRL	Air Force Research Laboratory Avionics Cyber Vulnerability Assessment and Mitigation Manual	Air Force Research Laboratory
Papp	Embedded systems security: Threats, vulnerabilities, and attack taxonomy[8]	CrySyS Lab
CVE	Common Vulnerabilities and Exposures	MITRE
N/A	A Taxonomy of Cyber Attacks on SCADA Systems[9]	UC Berkeley
BCIT IEL	British Columbia Institute of Technology Internet Engineering Lab	BCIT
NVD	National Vulnerability Database	NIST
CWE	Common Weakness Enumeration	MITRE

1. Abuse of Native Code

- An adversary takes advantage of intentional design features to inject malicious code.
- These features are often intended to reduce costs, increase quality, and expand capability, but also provide opportunities for lateral movement, or developing and inserting/hiding malware.
- Examples include hardware/software added to support debugging and testing, Joint Test Action Group (JTAG) ports and console ports, resetting a system to default conditions that enable default access, taking advantage of systems which use unsecured memory by replacing

[7] Portions of the NSA/CSS Technical Cyber Threat Framework are derived from MITRE's ATT&CK threat model, and no substantive differences relevant to this analysis were noted in review.

[8] D. Papp, Z. Ma and L. Buttyan, "Embedded systems security: Threats, vulnerabilities, and attack taxonomy," Privacy, Security and Trust (PST), 2015 13th Annual Conference on, Izmir, 2015, pp. 145-152

[9] B. Zhu, A. Joseph and S. Sastry, "A Taxonomy of Cyber Attacks on SCADA Systems," *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*, Dalian, 2011, pp. 380-388

executing code, leveraging network connectivity between a corporate LAN and a control system network, use of an API, use of normal software update mechanisms, use of an un-secured embedded device to send/retrieve data, exploitation of a lack of input validation (range checking, interlocks, authentication, path manipulation...), a device rolled back to vulnerable BIOS image, manipulation of set points or flight/mission data and configuration information or interference with sensor data often exploiting a lack of rigorous authentication or cryptography, such as leveraging the implicit trust a server places in the client, man-in-the-middle attacks, or server to server communication paths that have not been analyzed from a security perspective because processes "trust" other systems because they are behind a firewall, using a normal system function which is not documented to the customer but was included by the developer for testing or maintenance, and which the adversary has advanced knowledge of due to reverse engineering or intellectual property theft, exploiting a weakness in input validation or buffer management using crafted input data submitted using an interface functioning to process data input.
- An adversary uses functions of an application to achieve a malicious objective not originally intended by the application, without injecting malicious code.
- See also: CAPEC 242, 152, 126, 210, 113, 137, 390, 436, 440, AFRL, Papp, TARA T000005, T000322, T000335, T000340

2. Abuse of Pivot Functions

- An adversary takes advantage of intentional design features to directly communicate with other devices.
- Examples include normal use of a router to traverse allowed logical paths, reuse of a redirection page on a webserver, or leveraging a retransmission feature in a token ring network.
- See also: CAPEC 175, 240, AFRL, ATT&CK, TARA T000324

3. Authentication Implementation Flaw

- An adversary exploits limitations and assumptions in the mechanisms a target utilizes to manage identity and authentication external to the system.
- Examples include inherent weaknesses of an authentication mechanism, flaws in the authentication scheme's implementation, evading or circumventing an authentication mechanism to access protected data without authentication ever having taken place, manipulation of an application's information to change the apparent credentials or similar information, or engaging a device using algorithms or security policies that it supports but that are weaker than those it normally uses.
- See also: CAPEC 114, 115, 225, 172, Papp, ATT&CK

4. Change of Controlled Parameters

- This threat concept is reasonably assumed a certainty once the kernel mode execution state has been attained.
- An adversary modifies the operation of a system to undermine a critical function.
- Examples include disabling critical services the asset provides to the network, operating a system outside of safe parameters in critical cyber-physical control systems or improperly engaging weapons at an inappropriate time on a weapons platform.

5. Denial of Local Services

- An adversary depletes a local resource within the asset to the point that critical functionality is affected.
- Examples include locally exhausting virtually any local resource necessary for the target's operation, inducing account lockout, or remotely transmitting specially crafted network packets sent to the web interface of a PLC that result in resource exhaustion causing the device to restart.
- Sub-concepts include physical destruction (component fails to perform its intended function due to a tangible physical consequence, sabotage, mechanical degradation, physical anomaly, etc, which may or may not be a total failure) and gray failure (component fails to perform as intended, but remains minimally responsive, creating a discrepancy between the state of the required system functionality (down) and the apparent state of the component (up) where a component will appear "alive" to monitoring, even if it has failed).
- See also: CAPEC 119, 131, TARA T000342, T000156

6. Denial of Network Services

- An adversary disrupts the network resources of a target, without degrading the internal resource availability of the target.
- Examples include rapidly engaging in a large number of interactions with the target, inducing account lockout, monopolizing or disabling the network or a network resource, exploiting a weakness in rate limiting or flow control in management of interactions, or continually engaging a specific resource such that a given resource is tied up and not available to a legitimate user, blocking traffic associated with reporting to prevent remote analysis.
- End effects include loss of network connectivity, indication data or timing data.
- Sub-concepts include disruption of input, network alert suppression and disruption of output.
- See also: CAPEC 125, 161, 607, 227, TARA T000156

7. Exploitation of Poor Memory Management

- An adversary exploits programming errors to change system memory.
- Examples include input parsing vulnerabilities leading to buffer overflow problems, memory management problems such as using pointers referring to memory locations that have been freed, manipulating an application's interaction with a buffer in an attempt to read or modify data they shouldn't have access to or retrieving or providing more input than can be stored in the allocated buffer, resulting in the reading or overwriting of other unintended program memory.
- Notably, buffer attacks are distinguished in that it is the buffer space itself that is the target of the attack rather than any code responsible for interpreting the content of the buffer and in virtually all buffer attacks the content that is placed in the buffer is immaterial.
- See also: CAPEC 123, ATT&CK, AFRL, Papp

8. Impersonation

- This threat concept may be reasonably assumed a certainty after initial system access depending on the design of the system.

- An adversary bypasses an authentication mechanism using valid credentials.
- Credentials may be obtained from multiple sources, such as improperly controlled printed password lists, hardcoded passwords, passive listening or compromised assets with shared credentials.

9. Indicator / Alert Manipulation

- An adversary provides falsified indications or alerts to the operator to result in an operator action that degrades or defeats the mission, vice having a direct electronic effect on the mission and without necessarily altering network traffic providing those indications.
- Examples include an attack which manipulates the fuel indication which may lead an operator to conclude that insufficient fuel is present to complete a mission and take consequent actions to ensure safety of flight (such as landing).
- See also: CAPEC 527, 268, ATT&CK

10. Injecting Faults

- An adversary causes a fault in a device using disruptive signals or events which propagate through a control chain to cause further faults or failures in upstream equipment.
- Examples include packet and input crafting attacks exploiting parsing vulnerabilities in protocol implementations or other programs, replaying previously observed packets or packet fragments to cause protocol failures or deadlocking, manipulating characteristics of system data structures in order to violate the intended usage and protections of these structures. Notably, vulnerabilities and therefore exploitability of these faults often exists primarily due to ambiguities in the design of system interfaces.
- See also: CAPEC 624, 255, Papp, TARA T000333

11. Interface Device Use

- This threat concept is reasonably assumed a certainty once the kernel mode execution state has been attained.
- An adversary uses existing external interface hardware within normal parameters to communicate with other connected devices, via either a custom command and control protocol or a normal communications channel.
- Examples include bypassing a firewall by compromising secure settings, nimplementations mimicking well-known protocols, Base64 encoding the message body of an HTTP request, adding junk data to protocol traffic, using steganography, or interfacing over an input or output that isn't typically associated with communications, such as electrical power systems or sensors, moving data in our out of locations that are not visible to the user, such as embedded system memory, or over channels that lack network monitoring, use of commonly open ports, protocols associated with the port or a completely different protocol, communications over Link 16, Common Data Link (CDL), ACARS, ADS-B, Mode A/C/S transponders, Traffic Alert Collision Avoidance System (TCAS), Distance Measuring Equipment (DME), Traffic Information Service-Broadcast (TIS-B), Global Positioning System (GPS), Instrument Landing System (ILS), VHF Omnidirectional Range (VOR), Marker Beacon, and VHF/UHF voice, or any other protocols that are well known and documented giving adversaries the technical details they need to analyze them and to potentially exploit them, internet access provided to passengers on an aircraft if

not implemented securely could potentially give an adversary access to aircraft avionics, or shared external satellite communications links used for both internet access and operational functions.

- Notably, largely due to how easy it is to define new protocols and use existing, legitimate protocols and network services for communication, detecting intrusion without prior knowledge of well-defined set of acceptable data is a difficult proposition over the long term.
- Sub-concepts include exfiltration via covert channels and native channels.
- See also: ATT&CK, AFRL, TARA T000167

12. Interface Overload

- This threat concept applies for any data-consuming interface which has not been formally verified as non-manipulable.
- An adversary uses existing external interface hardware within normal parameters to communicate with other connected devices, via either a custom command and control protocol or a normal communications channel.
- Examples include implementations mimicking well-known protocols, Base64 encoding the message body of an HTTP request, adding junk data to protocol traffic, using steganography, or interfacing over an input or output that isn't typically associated with communications, such as electrical power systems or sensors, moving data in our out of locations that are not visible to the user, such as embedded system memory, or over channels that lack network monitoring, use of commonly open ports, protocols associated with the port or a completely different protocol, communications over Link 16, Common Data Link (CDL), ACARS, ADS-B, Mode A/C/S transponders, Traffic Alert Collision Avoidance System (TCAS), Distance Measuring Equipment (DME), Traffic Information Service-Broadcast (TIS-B), Global Positioning System (GPS), Instrument Landing System (ILS), VHF Omnidirectional Range (VOR), Marker Beacon, and VHF/UHF voice, or any other protocols that are well known and documented giving adversaries the technical details they need to analyze them and to potentially exploit them, internet access provided to passengers on an aircraft if not implemented securely could potentially give an adversary access to aircraft avionics, or shared external satellite communications links used for both internet access and operational functions.
- Notably, largely due to how easy it is to define new protocols and use existing, legitimate protocols and network services for communication, detecting intrusion without prior knowledge of well-defined set of acceptable data is a difficult proposition over the long term.
- Sub-concepts include exfiltration via covert channels and native channels.
- See also: ATT&CK, AFRL, TARA T000167

13. No Authentication

- This threat concept is reasonably assumed a certainty for systems which lack an authentication mechanism.
- An adversary is able to freely communicate with the input interface of a target without the need to bypass any security mechanism.

14. No User Mode / Kernel Mode Differentiation

- This threat concept is reasonably assumed a certainty for systems which lack user mode / kernel mode differentiation.
- An adversary able to execute arbitrary code may do so without restrictions.

15. Privilege Management Implementation Flaw

- An adversary exploits limitations and assumptions in the mechanisms a target utilizes to manage identity and authentication internal to the system.
- See also: CAPEC 122, ATT&CK, AFRL, Papp, TARA T000331, T000161

16. Resource Manipulation

- An adversary manipulates one or more unsecured local or network resources without directly modifying the system under attack, affecting application behavior or information integrity.
- Examples include modification of information at the source or in transit, and includes modifications of web pages, email messages, file transfers, the content of other network communication protocols, timing information or sensor data, or remotely hosted configuration information. Notably, this often targets invalid assumptions that may be inherent in implementers of the protocol, incorrect implementations of the protocol, or vulnerabilities in the protocol itself, and thus does not apply to secure, authenticated communications.
- Sub-concepts include PNT manipulation (a fault, failure, input modification, or output modification achieved with regard to a sensor's position, navigation or time readings), mis-trained machine learning (affects on the output of a machine learning algorithm to include mis-training the system, re-training the system, or crafting "optical illusions" for the input) and data integrity manipulations (fault, failure, input modification, or output modification achieved by modifying the data consumed or produced by a system component).
- See also: CAPEC 148, 156, 272, 262, 176, TARA T000038, T000056

17/18. Sensor Repurposing (Input or Output)

- Any use of a sensor other than its designed purpose. Notably, every sensor may be a potential vector for both data input and data output. This threat concept can reasonably be assumed a certainty if the component is connected to the system.
- Examples include using a microphone as a speaker (and vice versa), using an accelerometer as a microphone, modifying thermometer readings with heat proximal to the sensor, and modifying pressure or volume readings with localized disturbances.
- Sub-concepts include optical, pressure / force, acoustic, thermodynamic, electromagnetic and wiring manipulation.
- See also: ATT&CK, TARA T000167