Cyberphysical Security Through Resiliency: A Systems-Centric Approach

Cody Fleming, Iowa State University
Carl R. Elks, Virginia Commonwealth University
Georgios Bakirtzis, Stephen C. Adams, and Bryan Carter, University of Virginia
Peter A. Beling, Virginia Tech
Barry Horowitz, University of Virginia

Cyberphysical systems require resiliency techniques for defense, and multicriteria resiliency problems need an approach that evaluates systems for current threats and potential design solutions. A systems-oriented view of cyberphysical security, termed Mission Aware, is proposed based on a holistic understanding of mission goals, system dynamics, and risk.

Cyberphysical systems (CPSs) are often defended in the same manner as IT systems—by using perimeter security. Multiple factors make such defenses insufficient for CPSs, but resiliency shows potential in overcoming these shortfalls. Techniques for achieving resilience exist; however, methods and theory for evaluating resilience in CPSs are lacking. We argue that such methods and theory should assist stakeholders in deciding where and how to apply design patterns for resilience. Such a problem potentially involves tradeoffs between different objectives and criteria, and these decisions need

Digital Object Identifier 10.1109/MC.2020.3039491
Date of current version: 4 June 2021

to be driven by traceable, defensible, and repeatable engineering evidence. Multicriteria resiliency problems require a systems-oriented approach that evaluates systems in the presence of threats as well as potential design solutions once vulnerabilities have been identified. We present a systems-oriented view of cyberphysical security, termed *Mission Aware*, that is based on a holistic understanding of mission goals, system dynamics, and risk.

CPSs[1] are increasingly the subjects of cyberattacks and threats. From microelectronic chips to operating systems (OSs), data networks, and wireless network protocols, threats and exploits proliferate at increasingly high rates, caused by adversaries, from nation-state actors to hackers.[2,3] Unlike IT systems, where vulnerabilities can lead to loss of information or privacy, vulnerabilities in the highly integrated information processing and physical control technology intrinsic to CPSs could have public health and safety consequences.[4–6]

Perimeter-based security approaches (such as firewalls and encrypted communication channels) show some success in protecting CPSs. However, a purely perimeter-based defense is asymmetric because attackers have the advantage of choosing the point of attack and access to many new kinds of vulnerabilities. In the context of a CPS, perimeter security tends to be agnostic to the system's purpose, its required service or mission, and the functional behaviors of its cyberphysical aspects. Securing individual components is important in some contexts, but CPSs are vulnerable to compromises in the interactions among components, even in the absence of what would traditionally be viewed as an individual component attack.[7] They are also vulnerable to supply chain and insider attacks.[8]

Instead of attempting to react to, or predict, adversaries' specific capabilities, attack resilience is the ability of a system or domain to withstand attacks or failures and, in such events, reestablish itself quickly.[7,9] The goal of resilience (particularly for a CPS) is to proactively ensure the safety of the system by maintaining state awareness and physical system control. By first focusing on the safety of CPSs, engineers and analysts can bound or focus the problem in ways that are challenging for pure IT systems.

There are many approaches to attack resilience, and it is currently an open problem and an active field.[8,10,11] While many of these techniques are successful, each has a particular implementation and associated costs as well as operational expenses that involve both financial and performance tradeoffs. For a given solution, system designers and owners must understand the tradeoffs among costs, performance degradation, complexity, and improvements in resilience. To make matters worse, there is a combinatorial number of different solutions when one considers all of the possible solutions for a given function along with the collections of functions that are found in a CPS. The design of a CPS engenders a potentially intractable decision problem.

We then have the following questions: What set of resilience-based solutions can be used, where in the CPS should these solutions be deployed, and in what combination?

We argue that a systematic, tractable, and rigorous method is needed to support decision making for implementing CPS resilience solutions. Designers of CPSs must be able to manage the complexity of the decisions themselves and to understand, and balance, the benefits and costs of resilience solutions. We have developed a framework called Mission Aware cybersecurity that aims to manage complexity through general systems theory, framing CPS cybersecurity as a safety-control problem. Mission Aware supports decision making through the use of three fundamental concepts: 1) CPS modeling based on systems theory and top-down hazard analysis, 2) automated vulnerability assessment via mining of attack databases, and 3) reusable design patterns, many of which exist in the literature and some of which have been developed by the authors. To explain and demonstrate these concepts, we develop an example based on an application to an unmanned aerial vehicle (UAV) performing a tactical reconnaissance mission.

MANAGING COMPLEXITY THROUGH ABSTRACTION

Rather than beginning with tactical issues of how to protect a system against attacks, a strategic approach begins with questions about what essential services and functions must be secured against disruptions and how these disruptions can lead to unacceptable loss. The specific implementation details will be used later to reason more thoroughly about only a subset of all of the possible vulnerabilities, that is, only those combinations that can lead to specific undesirable outcomes. We argue that any resilience approach should move "top down," from general to specific, from abstract to concrete, and from system-level goals and hazards to component-level behaviors and their interactions.

One of the powerful ways to manage complexity is by using hierarchical abstraction and refinement. By starting at a high level of abstraction with a small list of hazards or goals and simple models and then refining that list and its associated models with more detail at each step, the stakeholders can be more confident about the completeness and consistency of the analysis. The reason is that each of the longer lists of causes (refined hazards or causes) and more complex models (refined behaviors, analysis, and simulations) can be traced to the small starting list and models. With this approach, high-fidelity modeling, analysis, and simulation are needed on only a subset of the CPS to provide assurance of correct behavior during deployment. By beginning with unacceptable or undesirable outcomes at the top (and not all possible outcomes), this approach reduces the total state space that one might need to explore at the lowest levels of abstraction.

Using the abstraction techniques described previously, we can leverage historical vulnerability and weakness databases more effectively by basing search parameters on strategically relevant parts of the system. Historical vulnerabilities associated with those relevant system components help identify potential threats to the system and, consequently, motivate the choice of particular resilience strategies to use in response to those threats.

UAV-mission use case

We use as an example a UAV within a tactical reconnaissance mission that requires the vehicle to produce data about the terrain, human activity, and aerial traffic within a particular area of interest. This particular mission involves identifying and localizing possible uncontrolled fires. Consequently, there is a pressing need that the UAV, the sensors that collect the data, and the data all maintain an acceptable level of performance in spite of potential adversarial actions. This mission is complex in that it involves a diverse set of components and technologies that are subject to a variety of potential threats. At the same time, UAV-based reconnaissance is a familiar scenario in many domains.

SYSTEMS AND GRAPH THEORY FOR SAFETY AND SECURITY

Mission Aware involves an early systems engineering process that identifies a high-level set of system objectives and unacceptable losses that represents system owners, operators, and other stakeholders.[12,13] Assuming that one has a high-level, comprehensive set of unacceptable outcomes, Mission Aware then involves constructing a model of the system from a control perspective based on the System-Theoretic Accident Model and Processes (STAMP) framework.[14] Specifically, we identify the controllers, the actions available to them, and the way in which those actions potentially lead to mission losses.

STAMP is an accident-causality model that captures accident-causal factors, including organizational structures, human error, design and requirements flaws, and hazardous interactions among nonfailed components.[15] In STAMP, system safety is reformulated as a system-control problem rather than a component-reliability issue: accidents occur when component failures, external disturbances, and/or potentially unsafe interactions among system components are not handled adequately or controlled. The safety controls intended to prevent such accidents are embodied in a hierarchical safety-control structure, whereby commands or control actions are issued from higher levels, and feedback is provided from lower levels.

There is an important difference between STAMP and traditional hazard analysis techniques, such as failure modes and effects analysis and fault tree analysis. The latter primarily focus on system failure (for example, system reliability) as a function of individual component failures in the system, which is quantifiable via physical failure rates. Other potential causal factors, such as complex software errors and unsafe component interactions, often are not thoroughly considered. There is less technical agreement on quantifying security probabilities, where the origin of failure is from an adversarial act, a design vulnerability, or a misinterpretation of security requirements. For this reason, our approach stresses systematic methods to aid in the design and selection of resilience measures that are agnostic to the underlying probability of a successful attack.

The Mission Aware framework (Figure 1) systematically encodes the following from the mission level all of the way down to the hardware and component levels:

1. the unacceptable outcomes of the mission
2. the hazardous states that can lead to those outcomes
3. the control actions that could lead to hazardous states and the circumstances under which those actions can create hazardous states
4. the combinations of causes that can lead to hazardous control actions.

This process allows for full top-to-bottom and bottom-up traceability, which supports evaluation of the cascading effects of specific changes to hardware, software, the order of operations, or other classes of behaviors on the potential outcome of a mission. This information is then used to identify vulnerable areas appropriate for resiliency or other security solutions. The pieces of information collected in the STAMP-based analysis are encoded into the system models (Figure 2), which are then used for further analysis and updated iteratively.[12,13]

The specification graph (S-graph) combines diverse types of "states," or nodes, to represent the system operating in its mission environment. Valid decision behaviors of the operator, hazardous conditions, and mission outcome nodes are encoded as truth tables, which perform the standard Boolean algebra on critical combinations of states in the system. Physical state nodes represent the set of variables that influence (or are influenced by) mission outcomes, while actuator and sensor nodes represent the system's ability to manipulate and measure those physical states, respectively. The sets of control actions or transition conditions between nodes are represented by edges between controllers or between a controller and its actuator(s). By using this graphical representation, we capture the traceability from the STAMP-based analysis between mission outcomes and individual component interactions, behaviors, or vulnerabilities.

STAMP can generate many hazardous scenarios.[15,16] We pick one simple example to illustrate traceability in the S-graph and the way in which this traceability then provides a focus for threat modeling and design.

For the UAV example, mission stakeholders defined the mission, its goals, unacceptable outcomes, and other relevant operational insights. Let us assume that the most pressing loss is the loss of trustworthiness of the reconnaissance information chain based on the mission description. This scenario can be illustrated as a situation where fire services believe there is no fire near Waypoint A, when the UAV actually has imaged an area far from Waypoint A. The aircraft first navigates to an unrequested region and then captures information about that area by activating its imaging payload. This scenario involves otherwise "correct" control actions (the flight control system maintains stable control and the payload activates), but the UAV images the wrong area and sends this information to the operator. Incorrect mission decisions could result from the failure of the UAV to image fires in a vulnerable area.

This example illustrates the notion that STAMP and the S-graph can capture coupling and interdependencies among multiple functions and controllers within a system. Among many possible lower level causes of this scenario, the navigation function of the UAV becomes critical, illustrating the coupling among the navigation components, the active control of the aircraft control surfaces, and the timing of when the imagery payload is activated.

Here we map this informal example scenario to respective nodes and edges in the S-graph (Figure 2):

- mission loss (orange node): inappropriate allocation of suppression resources by fire service
- hazard (orange node): a combination of (incorrect) latitude–longitude values plus activation of payload
- unsafe control action (gray nodes): 1) payload activated out of sequence with respect to 2) manipulation of control surfaces, leading to preceding hazard
- causal factors (attitude sensor node): inaccurate and/or delayed UAV location information

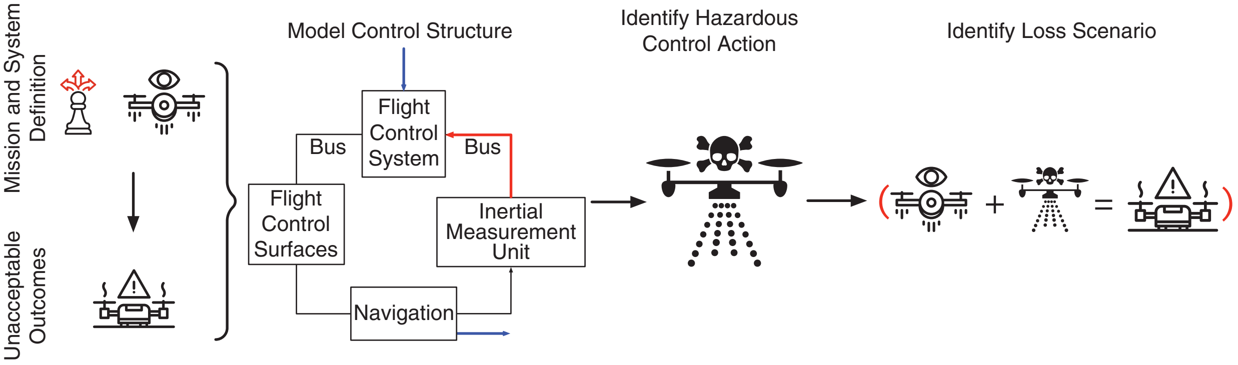

FIGURE 1. STAMP-based modeling and analysis begins with defining the system and the mission it performs. Then, by identifying unacceptable outcomes, the system's control structure and hazardous control actions help define the conditions that can lead to loss scenarios. If the conditions that lead to losses are identified, then we can implement safeguards in the form of resiliency or other solutions to prevent those conditions from occurring.

- resulting focus of threat modeling (see the next section): GPS and other navigation equipment/software.

At this point in the methodology, the model is agnostic to the initial cause of the scenario, but it can be further augmented with a concrete threat model. In turn, the mission-level requirements and threat model could better inform resilience or hardening of defenses.

Threat modeling

The S-graph—as produced by the STAMP analysis—results in finding and annotating the mission-critical subsystems in the initial model. From this analysis, a threat model naturally emerges in the sense that analysts and designers can use the S-graph model to produce a list of the subsystems that, if exploited, could cause mission degradation. At this stage, however, the details present in the S-graph model are not at the right abstraction to further inspect if and how these subsystems could be exploited. Furthermore, a single list of elements is insufficient to produce metrics that can augment and inform a threat model, such as attack surfaces or exploit chains.

For these reasons, the S-graph must be modified to include particular implementation choices of software, network, and hardware that a designer is considering at the earlier stages of formulation (Figure 3). This extra information that is added to the S-graph is used to map subsystem elements to attack vector databases. Attack vector databases contain attack patterns, weaknesses, and vulnerabilities. Databases of this type, such as Common Attack Pattern Enumeration and Classification (CAPEC) and Common Weakness Enumeration (CWE), have little in the form of quantitative information but a lot in textual descriptions of exploits and their solutions. Therefore, to properly map entries related to the system under examination, it is necessary to add specific details

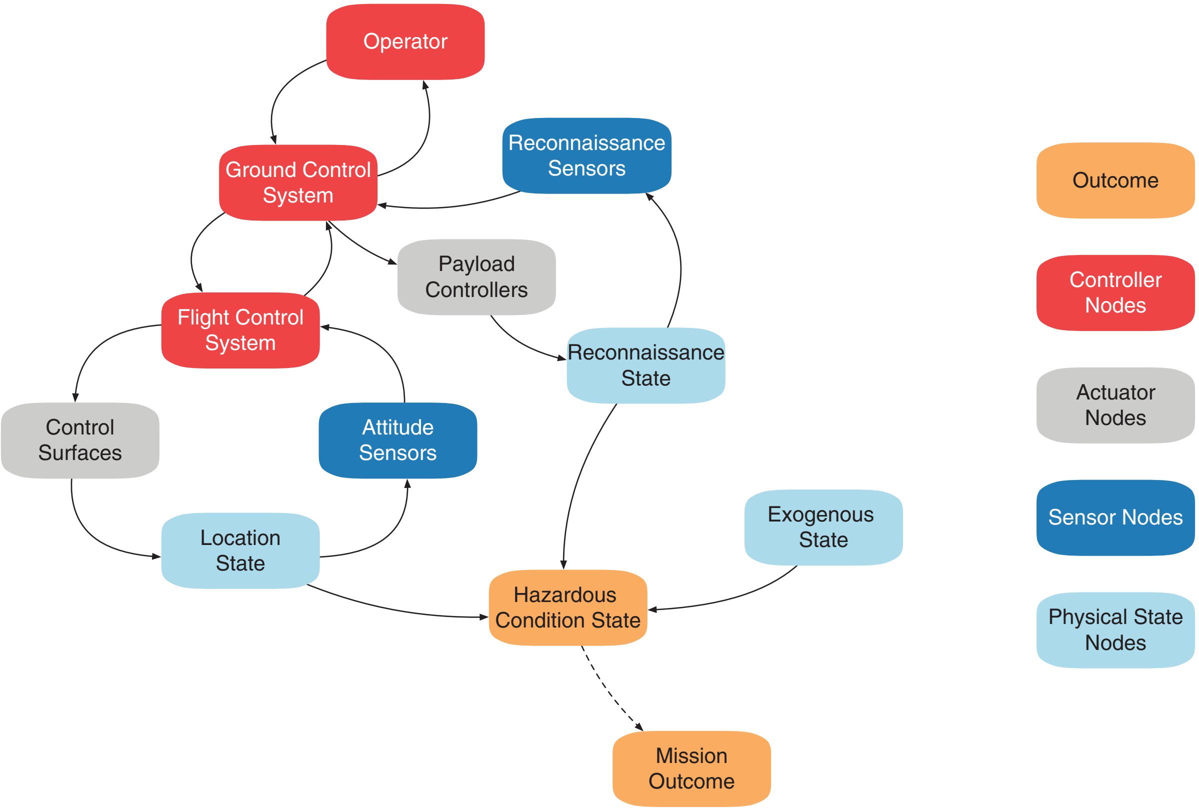

FIGURE 2. A portion of the specification graph (S-graph) for the reconnaissance UAV mission. The S-graph represents the system's functional control structure in combination with the physical states that determine the presence or absence of undesirable outcomes. By using multiple "types" of nodes in the graph, we can combine behavior, consequences, and control structure in the same model.

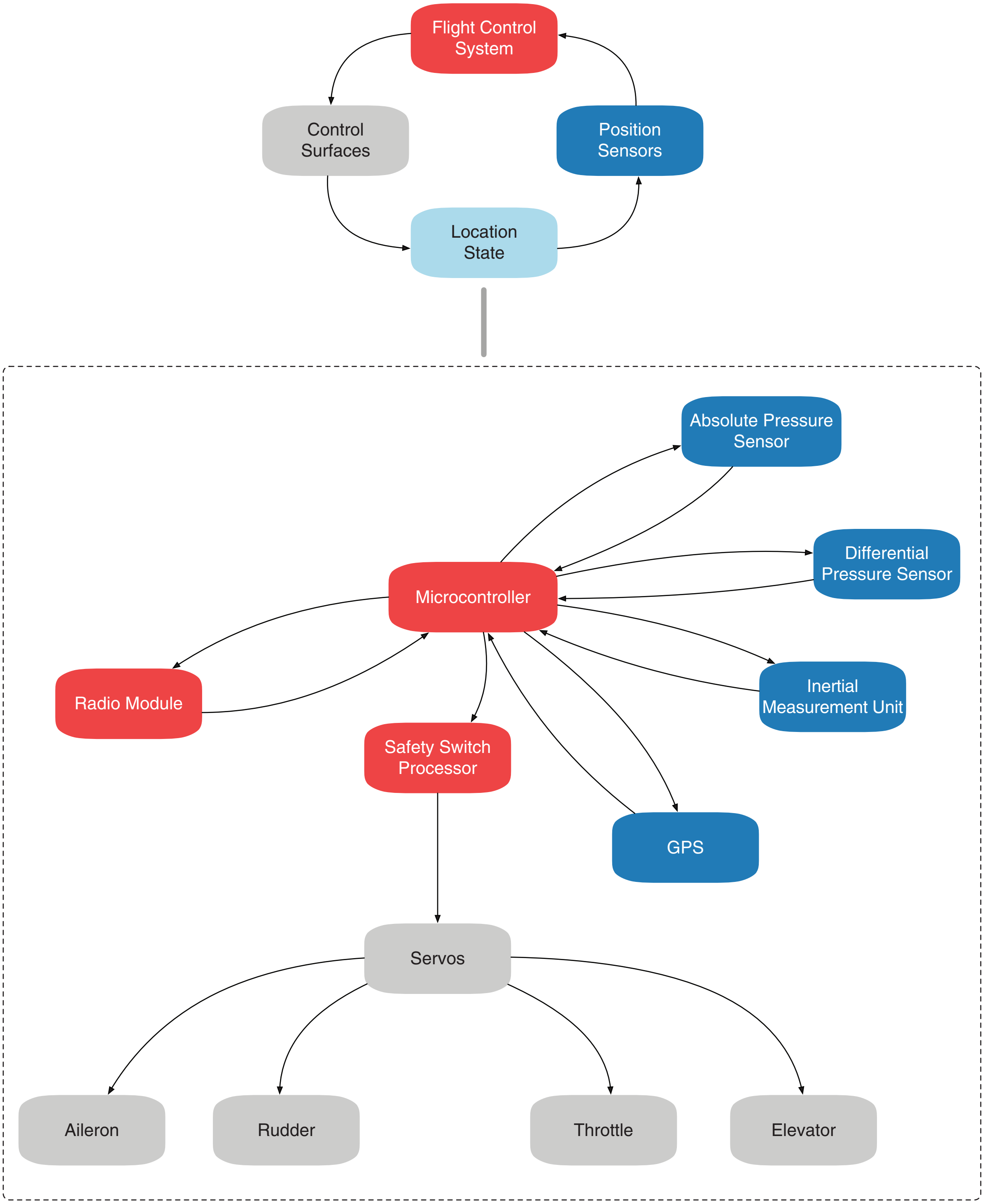

FIGURE 3. The S-graph is an initial formulation of subsystems that are critical to the mission. To develop a concrete threat model, these critical subsystems must be further decomposed to a system architecture, which is one of many possible designs. The reason for the necessity of the decomposition is twofold: 1) to have a system model at the same level of detail necessary to match to attack vectors and 2) to produce metrics such as the system's attack surface and exploit chains.

to the model on system implementation choices. Such choices could be, for example, the type of hardware platform, OS, or network connectivity or the designation of private or public networks. By adding extra keywords to the model, the S-graph is now able to associate with attack vector databases like CAPEC and CWE.[17]

This additional information in the S-graph assists in semiautomating the process of finding possible exploits as well as constructing the attack surface by locating attacks on the entry point of a given subsystem. Tools and visualization methods using natural language processing can be used to aid this process.[18,19] As the system is refined, implementation choices with associated details emerge. It is at this phase in the lifecycle development where potential exploits or attack surfaces for the system architecture are captured.

A critical subsystem might not be immediately accessible by attackers, but a series of attacks starting from the network-accessible elements of the system and continuing by exploiting a series of connected subsystems can manage to reach and exploit a critical subsystem. For example, the microcontroller of a UAV might not be immediately attackable, but there exists a possible series of attacks that start with the radio modules that does lead to the microcontroller being exploited. Another more direct example is the exploitation of the GPS that is used to provide positioning to the flight controller. The mapping between the system model and attack vector databases in this case produces Common Vulnerability and Exposure (CVE)-2016-3801, which describes a vulnerability in the MediaTek GPS drivers for Android One devices. This specific CVE may not directly affect a UAV system; however, it is an instantiation of CWE-264, Permissions, Privileges, and Access Controls, which is not immediately produced by the information added by system designers but can be (automatically) identified because of the hierarchical nature of attack vector data. This broader class of weaknesses gives system designers, integrators, and operators a basis from which to be aware of the risk associated with specific implementation choices.

This threat modeling analysis can be conducted at various times in the system lifecycle as different levels of system and component details emerge from the preliminary design. The benefit of this type of threat modeling is that it is systematic and traceable. Analysts and designers can augment system requirements, modify the system architecture, or implement secure design principles to reflect the findings of the threat posture analysis. The stopping point is when a complete threat model is drafted with respect to the proposed system architecture containing attack vectors that map to system elements. The attack surface resulting from the analysis informs analysts where attacks can enter the system and outlines exploit chains. By producing a complete threat model based only on system models, it is possible to incorporate security analyses earlier in the lifecycle and, therefore, inform security design decisions at a stage where design changes have much less of a cost impact. Such actions could be security focused, such as opting for more secure hardware and software platforms, or resilience focused, where redundancy, component diversity, and recovery principles are employed to secure the architecture.

DESIGN PATTERNS FOR RESILIENCE

As suggested in the 2018 National Academy of Engineering workshop on cyberresilience, system-level patterns for implementing resilience are needed.[11] In the Mission Aware approach, system enhancements take the form of reusable design patterns (physical, software, or procedural) that are intended to increase resilience. The set of all possible design patterns is large for even a simple system, and it is possible that several of these potential patterns would have little effect on the resilience of the system. Nonetheless, one can describe most resilience design patterns in terms of several well-known principles that follow from system self-healing and self-protecting attributes from autonomic computing: 1) diversification, 2) redundancy, 3) randomization, and 4) system policy adaptation.[20] Design patterns using these principles adapt the system to increase the effort required to successfully attack and compromise it. Note that adaptation might create unintentional effects that degrade system performance—an observation that reinforces the need to explicitly model the requirements or objectives of the system.

Redundancy is a common design paradigm in which multiple critical components are used to perform the same function so that if one fails, another can take its place. For a resilience system, diverse redundancy requires that two or more components that perform the same function be diverse with regard to a common attack pattern. It is imperative that the components be different because, even with redundancy, a common source of failure (such as a successful attack) could cause all of the redundant components to fail. This design pattern mitigates the ability for a single successful attack to compromise all redundant components.

A redundant configuration of the flight control system in the UAV example mitigates the risk of vehicle loss caused by the failure of a single controller. However, if a supply chain attack successfully embeds a Trojan horse onto the controller, then redundancy is not sufficient. While the system is resilient to natural failures, this single attack can compromise all controllers. The diverse redundancy solution is to procure each controller from a different supplier, thus mitigating the risk of an insider supply chain attack.

Similar to diverse redundancy, verifiable voting[8] uses redundant components to confirm the output of a system. Each component is a voting mechanism implemented in software or hardware and should be simple enough to be secured. If the voting mechanisms do not agree on an output, it is likely that an attack or some other fault has occurred. Verifiable voting allows for the system to remain in use when under attack. If it can be confirmed that only one voting mechanism is compromised, the results from the other voting mechanism can still be used. For example, a UAV's primary source of information may be a standard camera. The data from the camera are transmitted to a media server and then relayed via a wireless signal to interested parties. The media server is vulnerable to an insider attack. A verifiable voting solution to this vulnerability is to install a secondary camera payload with lesser but acceptable performance and a second media server in a separate location. The redundant system monitors the same information but mitigates the risk of attack by supplying redundant information to the interested parties.

Physical configuration hopping[8] is another design pattern for resilience derived from the idea of redundant systems. As in diverse redundancy and verifiable voting, several redundant components are given the same objective in a system. However, when implementing physical configuration hopping, control and execution are randomly moved among the redundant components. Physical configuration hopping can be combined with diverse redundancy to further mitigate the risk of an insider supply chain attack. In the UAV, physical configuration hopping can be implemented by randomly hopping control of location monitoring among the redundant flight controllers. Virtual configuration hopping is similar to physical configuration hopping; however, hops occur among virtual components instead of physical components.

EVALUATING RISK AND TRADEOFFS

As with any other design choice, resilience requires the ability to characterize and evaluate the tradeoff between potential gain in resilience and the cost of a proposed design. The cost of a design pattern can take on many notions, including the financial expense of acquiring and installing hardware, the cost of increased complexity, and the cost of operational degradation. While some of these amounts can be measured and quantified, calculating the gain in resilience is more difficult and remains an open question. The benefits of implementing security or resilience solutions include eliminating or reducing the possibility of attack or mitigating and containing the results of a successful attack. In particular, reasoning about resilience should be defined as a function of three variables or dimensions: 1) the severity of mission-level outcomes, 2) the complexity of the attack vectors needed to achieve the outcome, and 3) the cost and complexity of mitigating such attacks.

Consequence can only be determined (and ranked) by the owners and other stakeholders of the system, and these outcomes must be agnostic to implementation details, architectures, or threats. The key step is then providing a clear mapping between component vulnerabilities and system-level consequences, and vice versa. We provide one approach to obtain this information and mapping in the section "Systems and Graph Theory for Safety and Security," utilizing STAMP concepts and the S-graph. This represents one of the dimensions necessary to consider when designing resilience into systems.

The next dimension, attack complexity, requires an analysis of possible adversaries and their available techniques. Of course, there is a risk in not identifying all available techniques, which we mitigate by focusing on 1) resilience and not prevention and 2) system-level consequence, not likelihood. Given that this dimension is based on difficult-to-quantify assessments of attacker profiles and the techniques available to them, attack complexity is (currently) developed from expert opinion based on historical evidence.

The last dimension, mitigability, is a function of the mission itself, its system architecture, and the design patterns available to handle a set of threats (that is, the information derived from the section "Design Patterns for Resilience"). Currently, this dimension is based on expert opinion because of the number of disparate factors that contribute to this measure. A scoring method could be applied to compare different resilience solutions; however, it is likely that the appropriate weights and scoring functions vary from design to design.

Based on the criticality of maintaining location integrity in the UAV mission and the vulnerability assessment revealing the threat of attackers exploiting permission, privileges, and access control vulnerabilities in the UAV, a possible resilience strategy would be physical configuration hopping in the flight control system. This solution would significantly increase attack difficulty and workload for the adversary because of the changing attack surface. Additionally, because the mission is dependent on visual imagery being correctly linked to its location, diverse redundancy and verifiable voting are natural candidates. Implementation could be an entirely different type of GPS device or even a different class of navigation components (for example, based on inertial navigation). This pattern does not necessarily prevent or mitigate attacks on the GPS but may make mitigation of a successful GPS attack more effective; the system can revert to another mode of navigation if the attack is detected.

In summary, the application of a pattern may increase the complexity of an attack required to yield an adverse outcome and thus make the outcome less likely, or it may make the attack more mitigable and thus make the outcome less severe. In some cases, a design solution may achieve both results, or multiple design patterns can be used to address the same risk, as the preceding example illustrates. In any of these situations, it is also required to think about cost. In the case of multiple redundant GPS units, it might not be the extra component that is "expensive," although that may also be true. Rather, one must design and assess the voting scheme, ensuring that it is itself secure from attack. Our position is that the monitoring functions for these design patterns can and should be made as simple as possible—the implication being that something simple is easier to secure (for example, through formal verification or more complete testing).

Given that there may be multiple vulnerabilities in critical pathways, with multiple combinations of available design patterns, the system owners must make a tradeoff between their perceived risk of attack and its impact against the increased resilience the solutions offer. Of possible future research directions, developing metrics to quantify these tradeoffs may be some of the most critical work for advancing the Mission Aware methodology and the resilience of CPSs as a field of study.

Critical CPSs are becoming much more common in daily life, and better ways of securing them are essential. The state of practice for securing a CPS is at a point where new systems-oriented methods and tools are needed to adequately protect critical systems against advanced threats.

ACKNOWLEDGMENTS

This work was supported by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract HQ0034-13-D-0004. SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. Department of Defense.

REFERENCES

1. D. Serpanos, "The cyber-physical systems revolution," *Computer*, vol. 51, no. 3, pp. 70–73, 2018. doi: 10.1109/MC.2018.1731058.
2. A. A. Cárdenas, S. Amin, and S. Sastry, "Research challenges for the security of control systems," in *Proc. HotSec.*, USENIX, 2008, p. 15.
3. G. Bakirtzis, G. L. Ward, C. J. Deloglos, C. R. Elks, B. M. Horowitz, and C. H. Fleming, "Fundamental challenges of cyber-physical systems security modeling," in *Proc. 50th Annu. IEEE-IFIP Int. Conf. Dependable Syst. Networks-Suppl. Vol. (DSN-S)*, 2020, pp. 33–36.
4. D. Goodin. "Patient dies after ransomware attack reroutes her to remote hospital." *ArsTechnica*, 2020. [Online]. Available: https://perma.cc/N48B-5XB2
5. M. Giles, "Triton is the world's most murderous malware, and it's spreading," MIT Technology Review, 2019. [Online]. Available: https://perma.cc/5QS8-U37C
6. D. Serpanos, "There is no safety without security and dependability," *Computer*, vol. 52, no. 6, pp. 78–81, 2019. doi: 10.1109/MC.2019.2903360.
7. S. Amin, X. Litrico, S. Sastry, and A. M. Bayen, "Cyber security of water SCADA systems—Part I: Analysis and experimentation of stealthy deception attacks," *IEEE Trans. Control. Syst. Technol*, vol. 21, no. 5, pp. 1963–1970, 2013. doi: 10.1109/TCST.2012.2211873.
8. R. A. Jones and B. M. Horowitz, "A system-aware cyber security architecture," *Syst. Eng.*, vol. 16, no. 4, pp. 401–412, 2012. https://doi.org/10.1002/sys.21206.
9. I. Linkov, D. A. Eisenberg, K. Plourde, T. P. Seager, J. Allen, and A. Kott, "Resilience metrics for cyber systems," *Environ. Syst. Decis.*, vol. 33, no. 4, pp. 471–476, 2013.
10. M. Pajic, J. Weimer, N. Bezzo, P. Tabuada, O. Sokolsky, and G. J. Pappas, "Robustness of attack-resilient state estimators," in *Proc. ICCPS*, pp. 163–174, Apr. 2014.
11. National Academies of Sciences, Engineering, and Medicine. *Beyond Spectre: Confronting New Technical and Policy Challenges: Proceedings of a Workshop*. The National Academies Press, Washington, D.C., 2019.
12. B. T. Carter, G. Bakirtzis, C. R. Elks, and C. H. Fleming, "A systems approach for eliciting

ABOUT THE AUTHORS

CODY FLEMING is an associate professor at Iowa State University, Ames, Iowa, 50011, USA. His research interests include methods to assure the safety and security of controls systems and autonomy more broadly. Fleming received a Ph.D. in aeronautics and astronautics from Massachusetts Institute of Technology. He is a Member of IEEE. Contact him at flemingc@iastate.edu.

CARL R. ELKS is an associate professor in the Department of Electrical and Computer Engineering at Virginia Commonwealth University, Richmond, Virginia, 23284, USA. His research interests include the analysis, design, and assessment of dependable embedded cyberphysical systems of the type found in critical infrastructure. Elks received a Ph.D. in electrical engineering from the University of Virginia. He is a Member of IEEE. Contact him at crelks@vcu.edu.

GEORGIOS BAKIRTZIS is a Ph.D. candidate in computer engineering at the University of Virginia, Charlottesville, Virginia, 22903, USA. His research interests include model-based safety and security assessment for cyberphysical systems. Contact him at bakirtzis@virginia.edu.

STEPHEN C. ADAMS is a principal scientist at the University of Virginia, Charlottesville, Virginia, 22903, USA. His research interests lie at the intersection of systems and machine learning. Adams received a Ph.D. from the University of Virginia. He is a Member of IEEE. Contact him at sca2c@virginia.edu.

BRYAN CARTER is a graduate student in systems engineering at the University of Virginia, Charlottesville, Virginia, 22903, USA. His research interests include cyberphysical systems and systems analysis/modeling. Contact him at btc9an@virginia.edu.

PETER A. BELING is a professor in the Grado Department of Industrial and Systems Engineering and the Hume Center for National Security and Technology at Virginia Tech, Blacksburg, Virginia, 24061, USA. His research interests lie at the intersection of systems engineering and artificial intelligence. Beling received a Ph.D. from the University of California at Berkeley. He is a Member of IEEE. Contact him at beling@vt.edu.

BARRY HOROWITZ is the Munster Professor of Systems Engineering at the University of Virginia, Charlottesville, Virginia, 22903, USA. His research interests include cyber resilience and systems integration. Horowitz received a Ph.D. in electrical engineering from New York University. Contact him at bh8e@virginia.edu.

mission-centric security requirements," in *Proc. SysCon*, pp. 1–8, 2018.

13. B. T. Carter et al., "A preliminary design-phase security methodology for cyber-physical systems," *Systems*, vol. 7, no. 2, p. 21, 2019. doi: 10.3390/systems7020021.
14. W. Young and N. G. Leveson, "An integrated approach to safety and security based on systems theory," *Commun. ACM*, vol. 57, no. 2, pp. 31–35, 2014. doi: 10.1145/2556938.
15. N. G. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety.* Cambridge, MA: MIT Press, 2012.
16. N. G. Leveson, C. H. Fleming, M. Spencer, J. Thomas, and C. Wilkinson, "Safety assessment of complex, software-intensive systems," *SAE Int. J. Aerosp.*, v 2012. doi.org/10.3390/systems8030033.
17. G. Bakirtzis, B. J. Simon, A. G. Collins, C. H. Fleming, and C. R. Elks, "Data-driven vulnerability exploration for design phase system analysis," *IEEE Syst. J.*, pp. 1–10, 2019. doi: 10.1109/jsyst.2019.2940145.
18. S. C. Adams, B. T. Carter, C. H. Fleming, and P. A. Beling, "Selecting system specific cybersecurity attack patterns using topic modeling," in *Proc. TrustCom/BigDataSE*, pp. 490–497, 2018.
19. G. Bakirtzis, B. J. Simon, C. H. Fleming, and C. R. Elks, "Looking for a black cat in a dark room: Security visualization for cyber-physical system design and analysis," in *Proc. VizSec*, pp. 1–8, 2018. doi:10.1109/VIZSEC.2018.8709187.
20. S. Dobson, R. Sterritt, P. Nixon, and M. Hinchey, "Fulfilling the vision of autonomic computing," *Computer*, vol. 43, no. 1, pp. 35–41, 2010. doi: 10.1109/MC.2010.14.