Designing Deceptions for Protecting Industrial Control Systems

Neil C. Rowe[1]

[1] U.S. Naval Postgraduate School, Monterey, CA 93943-5201, USA
ncrowe@nps.edu
phone: 1-831-656-2462

Abstract. Industrial-control systems (ICSs) are especially vulnerable cyberattack targets because of their role in critical infrastructure, their infrequently updated software, and their frequent need to run uninterrupted. For them, deception is a valuable active-defense method under cyberattack, as another line of defense after access controls and system monitoring. We provide here a general theory for planning deception defenses for ICSs. The theory enumerates the rich variety of options that can be considered for attack targets, deception locations, deception tactics, adversarial material goals, and adversary psychological goals. Our approach is to learn a model of attacker goals and priorities using reinforcement learning from the actions taken by an attacker or a class of attackers from the same source, and their intended targets. Goals and targets are chosen from a list of possibilities in the system, and priorities are inferred from a set of psychological "adversary variables". We then select a set of deceptions most compatible with those goals and priorities. Implementing such deceptions can benefit from placing them in multiple layers for "defense in depth".

Keywords: Industrial control systems, Cyberattacks, Defense, Deception, Discouragement, Planning, Honeypots, Layers, Psychological modeling, Frustration, Reinforcement learning

This paper is a chapter in Q. Zhu, Z. Lu, P. Yu, and C. Wang (Eds.), *Foundations of Cyber Deception - Modeling, Analysis, Design, Human Factors and Their Convergence*, Springer Nature, 2025.

1 Introduction

Defensive deception is a powerful technique for defending computer systems and digital devices since it is often unexpected (Rowe and Rrushi 2016). The subtype of cyber-physical systems (CPSs) called industrial control systems (ICSs) are particularly vulnerable targets of cyberattacks due to their predominant function as critical infrastructure, their frequent use of old software for compatibility with hardware, and the difficulty of updating them to fix vulnerabilities due to the need to keep them running continuously (Ackerman 2017) (Kayan et al. 2022). ICSs thus greatly benefit from

additional defense methods including cyberdeception since traditional access controls and intrusion detection are insufficient to defend them (Sohl et al. 2015). Deception can also be especially effective in defending ICSs because many have large networks of uncommon devices and sensors in which attackers can become confused, and their controls are designed for specialists familiar with the underlying physics or chemistry, unlike many attackers. Although they can use defensive deception methods common to other digital systems, ICSs thus present new opportunities for deception related to their processes that have no counterpart in traditional operating systems. This chapter will survey ICS deception planning opportunities in the context of an experimental network which we are developing.

2 Previous work

A classic distinction is between defensive deception to encourage an attacker, as with honeypots trying to collect attack intelligence, and defensive deception to discourage an attacker, as with real ("production") systems trying to defend themselves (Rowe 2024). Most previous work has used encouragement deceptions to collect data on attacks, but we will focus here on discouragement deceptions for production systems. Deceptions tend to be more convincing on production systems because they can show real rather than simulated activity. Nonetheless, honeypots can test useful encouragement ideas as well.

(Franco et al. 2021) surveyed previous work with honeypots for ICSs and Internet-of-Things (IoT) networks. IoT designs are related because they also use large numbers of networked processors. However, ICS devices often provide more complex services than IoT nodes, and require more processing.

Honeypots can be considerably more useful when they use deception, since attackers avoid known honeypots because attackers know honeypots collect their data and thwart exploitation attempts (Rowe 2019). Previous work has examined ideas of building networks of honeypots (honeynets) for ICSs since honeypots can attract more attacks when they are present in large numbers ("honeypot farms"). Some relevant work is (Urias et al 2018), (Abe et al. 2018), (Cifranic et al. 2020), and (Dutta et al. 2020). Honeypots can be placed on broad networks, local networks, or in hardware (You et al. 2020). Honeypot farms can use a wide range of deceptions to attract a broad range of attacks.

Our previous work has built and tested honeypot types and tactics for defending ICSs. We showed that honeypots are best implemented with cloud services (Atadika 2019) as supported in (Dodson et al. 2020). After preliminary experiments, we have focused on the IEC 104 protocol for ICSs, since it is increasingly popular, and the HTTP protocol, the most popular protocol with our attackers (Foley et al. 2022). We have been using a design with a modified general-purpose honeypot Conpot as a front end to a GridPot electrical-grid honeypot on the back end (Rowe et al. 2020), which was later hardened with a user-interface front end on a separate machine and a separate logging site (Meier et al. 2023). Hardening was essential to collecting data on the more serious attacks since it prevented erasing the log or terminating the virtual machines used to run the honeypot. Further experiments showed we got similar traffic

independent of where in the world the honeypot was situated, whether it was a virtual machine, and what details it claimed about its operations (Rowe et al. 2022).

Deception for discouragement is appropriate in defending real infrastructure including ICSs. Its goal is to convince an attacker that attacking a system will be costly and unlikely to succeed. Discouragement deceptions are not easy to test because they should involve real systems and real attacks to accurately measure effectiveness. Nonetheless, we can implement them and compare them to similar systems that are not using discouragement, if we are patient enough to get a representative sample of attacks.

3 Example layered deception plan for a sophisticated adversary

Effective digital defensive deception requires advance planning, much like deceptions for defensive military operations (Clark and Mitchell 2019). We focus in this chapter on planning for protecting an ICS against a sophisticated adversary with many resources, such as a nation-state or "advanced persistent threat". We cannot stop such an attack indefinitely, given sufficient time and resources for an attacker to try many methods. However, it often suffices to delay them. Delays give defenders time to figure out the type, methods, sources, and targets of an attack, and that may suffice to divert or block the attack's most damaging aspects.

Military defenses often use multiple layers (or "lines") to delay attacks considerably, a concept called "defense in depth". Since ICSs are often analogous critical infrastructure, defensive ICS deception can also use multiple layers for "deception in depth" (Landsborough et al. 2024). Each layer can use different deceptions, choosing them for the setting and from an ontology (Basan et al. 2022). A layered deception permits testing different attacker skills at each level, providing valuable intelligence about the different methods they use. However, (Clark and Mitchell 2019) also suggests that intelligent adversaries are most effectively fooled by a consistent plausible "story", as for instance "the site is well hardened" or "the site is logging your activities". A useful plausible story for discouragement should be consistent through the levels of defense, reinforce prior attacker beliefs, and should encourage them to leave.

Consider an example of a layered deceptive defense to discourage an attacker of a real ICS network at a military facility. An initial line of defense could try to persuade adversaries that the ICS site is not worth attacking by mislabeling it as a less desirable target by its name and owner. An example would be labeling a water-treatment plant as a cistern system for collecting rainwater for gardening, something not critical to a military facility. However, the site must still run legitimate ICS protocols consistent with its cover story so that the adversary's network mapping can detect them. Legitimate users of the network would be told the correct resources to use.

A second line of defense could provide a confusing interface to the real network containing the ICSs. It could use cryptic descriptions of its resources with numbers and codewords to make it difficult for adversaries to recognize what is there. Legitimate users would be given a document to decipher these descriptions. Attackers could be quickly identified from their exploratory behavior.

A third line of defense could try to decoy adversaries from the legitimate targets another way, by providing files with explicitly false information like incorrect network maps, faulty instructions for using systems, and false data files allegedly obtained from the network. The more detailed a false story that is constructed, the more time that sophisticated adversaries will waste exploring the fakes. As a side benefit, the defender will obtain valuable information about adversary methods and goals from which to harden defenses.

A fourth line of defense could try to waste attacker time and prove attacker reconnaissance intentions by offering some minimal ICS network nodes that look promising but provide few services (Cifranic et al. 2020). This idea goes back to (Cohen 1999) and is used in several commercial tools. It provides a quick way to identify reconnaissance. Its disadvantage is that network scanning tools like Nmap can quickly reveal minimal nodes.

A fifth line of defense could provide false error messages to discourage exploration, while legitimate users would be given secret codeword to permit normal operation. False error messages have the advantage of working best against alert and sophisticated attackers that examine their interactions carefully, like advanced persistent threats. Many kinds of false error messages can be provided (Rowe and Rrushi 2016 chapter 9). Since messages are verbal rather than architectural, they can deceive in a different way even if adversaries have figured out the others.

A sixth line of defense could be a confusing or obfuscating user interface to a device, as contrasted with the confusing interface to a network in the second line of defense. This can be a visual deception rather than a verbal or architectural one, and may have a renewed chance of working.

A seventh level of defense could transfer a persistent attacker to a safe "sandbox" environment simulating a device, and either feed them fake data about the state of the device or simulate fake states consistent with the adversary commands. The GridLab-D simulator of an electric grid from Pacific Northwest Laboratories is an example we have used in our honeypot research (Rowe et al. 2021). Even if an adversary could infer they have been transferred to a simulation, it will take them a while to recognize it, and in the meantime, they will provide useful data about their methods and goals.

4 Deception planning

We now present a defensive deception strategy for ICSs. (Rowe and Rrushi 2016), (Han et al. 2018), and (Pawlick et al. 2019) provide taxonomies of broadly applicable deception methods for cyberspace on which we build here. ICSs can use most of these, but they have additional deception capabilities due to their mission of controlling devices. A few are specific to programmable logic controllers (Morales et al. 2020), but we can get some generality if we take a broader perspective. With ICSs, attackers generally want to set switches and dials to interfere with or stop normal operations. Deceptions can fool them into thinking that they have done this and achieved goals.

4.1 A procedure for deception planning

Fig. 1 shows our defensive-deception discouragement-planning procedure for local-area ICS networks and their systems.

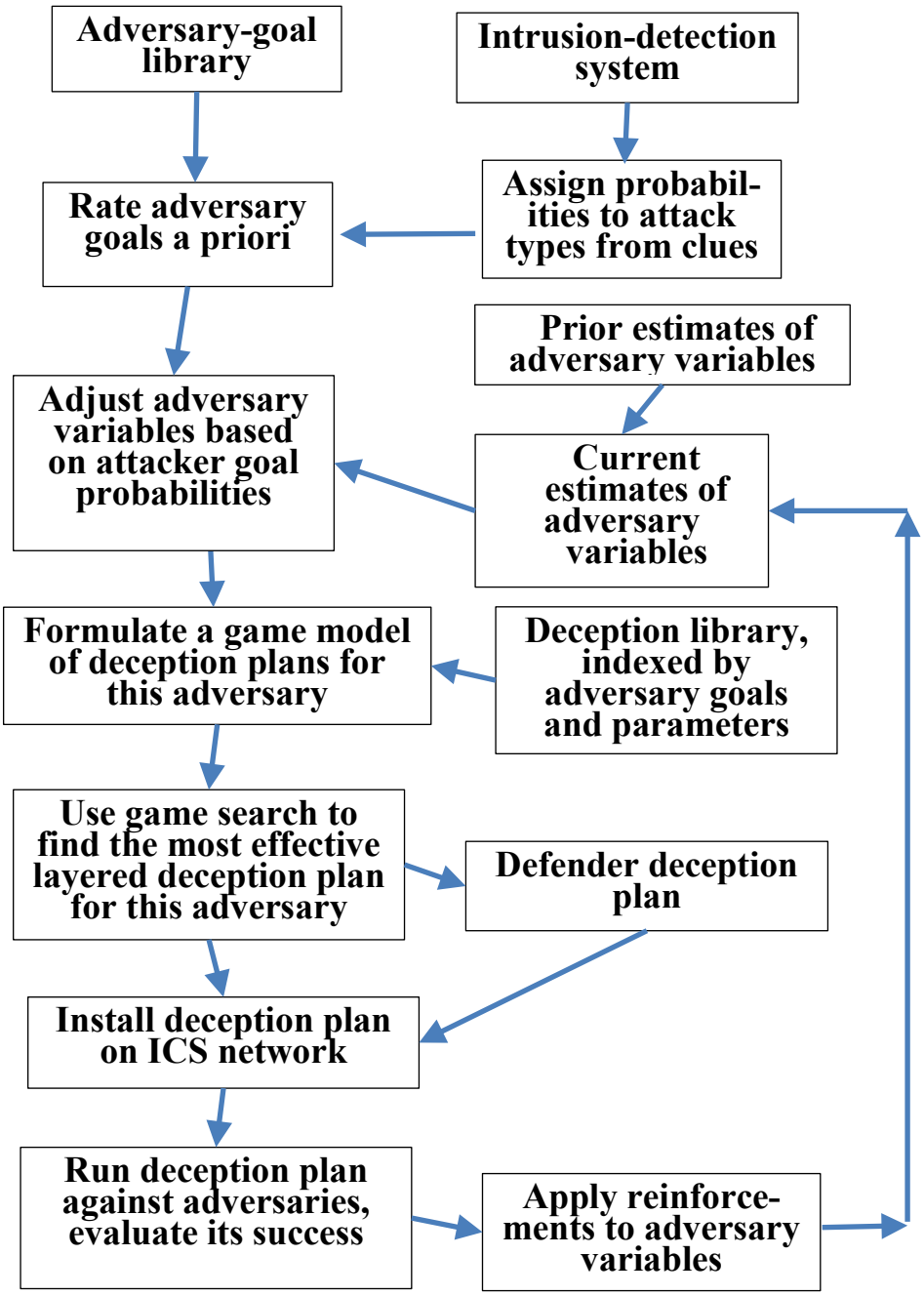

Fig. 1. Discouragement-deception planning flowchart for defending ICS systems.

The overall procedure for ICS defensive deception planning can also be summarized as follows:

1. Use an attacker-goal library to rate possible attacker goals for those possible cyberattacks, including both physical and psychological goals.
2. Use an intrusion-detection system to identify cyberattacks in activity on the local-area network of the ICS (Qassim et al. 2020).
3. Obtain estimates of adversary variables from prior values and recent network activity.
4. Modify adversarial variables based on conditional probabilities of goals.
5. Use a game model to propose and rate possible deception tactics for the defender based on a deception library.
6. Use game theory to build a good deception plan, and implement it in the local-area network.
7. Run the deception plan against attackers, and monitor the results.
8. Calculate reinforcements on adversary variables, and apply them.

4.2 Attacker detection

Defenses must first distinguish normal traffic from attack traffic. Standard ways are access monitoring, log inspection, signature-based intrusion detection, and anomaly-based intrusion detection. These must be specialized to handle ICSs since ICS protocols differ significantly from those of other information systems. ICS attacks may provide specialized clues such as attempting to set device parameters to unusual values, trying to interact with devices in nonstandard ways, or trying to violate physical constraints (Yahya et al. 2020) (Rrushi 2022). However, ICSs are also increasingly using Web (e.g., HTTP) and remote-desktop (e.g., SSH and RDP) protocols for easier network management, and also can be susceptible to the many attacks on them. An advantage that ICSs have is that attacks are easier to distinguish than with most digital systems since ICS traffic is often very regular, as attack traffic such as enumeration of networks, requesting unusual information, and setting of parameters to unusual values can be easy to recognize.

4.3 Attacker goals and deception venues

A key issue is what attack activities on an ICS merit deception. Overuse of deception makes it easier to detect (Rowe & Rrushi 2016), so it is desirable to deceive sparingly and only at important steps in an attack. Table 1 and Table 2 show, in the first two columns, a set of attacker goals for ICSs proposed in (Ackerman 2017), followed by our assessment of their deception possibilities. These provide options for step 2 in the deception-planning procedure of section 4.1.

Table 1. Evaluation of broad options for deception in ICS defense, part 1.

Attack goal	Target	Attack difficulty	Possible deceptions	Deception difficulty	Deception desirability
Do reconnaissance	ICS network	Easy	Decoys, honeypots	Easy	High
Steal credentials	ICS network	Moderate	Fake traffic	Moderate	Moderate
Analyze packets to system to get intelligence	ICS network	Difficult	Fake traffic	Moderate	Moderate
Replay network traffic	ICS network	Moderate	Honeypots	Moderate	Moderate
Inject packets	ICS network	Difficult	Honeypots, false data	High	Moderate
Spoof packets	ICS network	Difficult	Honeynets	High	Moderate
Gain remote access	Controllers	Moderate	Honeynets	High	High
Modify data to/from controller	Controllers	Difficult	Honeynets	Moderate	Moderate
Modify configuration	Controllers	Difficult	Honeypots	High	Moderate
Modify control algorithms	Controllers	Difficult	Honeypots	High	Moderate
Modify data to affect control	Controllers	Moderate	Honeynets	Moderate	High
Modify controller firmware	Controllers	Difficult	False data	High	Moderate
Modify I/O data of controller	Controllers	Moderate	False data	Moderate	Moderate
Escalate privileges	Work-stations	Difficult	Decoys, honeypots	Moderate	Moderate
Gain remote access	Work-stations	Easy	Honeypots	High	Moderate
Copy sensitive data	Work-stations	Moderate	Honeypots	Easy	Low
Modify data	Work-stations	Difficult	Honeypots	Moderate	Moderate
Modify configuration	Work-stations	Difficult	Decoys, honeypots	Moderate	Moderate
Send commands to controller	Work-stations	Moderate	Honeypots	Moderate	Moderate
Maintain persistence	Work-stations	Moderate	False data	Moderate	Moderate
Do denial of service	Work-stations	Easy	Delays	Easy	Low

Table 2. Evaluation of broad options for deception in ICS defense, part 2.

Attack goal	Target	Attack difficulty	Possible deceptions	Deception difficulty	Deception desirability
Escalate privileges	Application and SCADA servers	Difficult	Decoys, honeypots	High	Moderate
Gain remote access	Application and SCADA servers	Difficult	Honeynets	Moderate	Moderate
Copy sensitive data	Application and SCADA servers	Moderate	Honeypots	Easy	Low
Modify data	Application and SCADA servers	Difficult	Honeypots	Moderate	Moderate
Disrupt process communications	Application and SCADA servers	Moderate	False data	Hard	Moderate
Disrupt user interface	Application and SCADA servers	Moderate	False data	Hard	Moderate
Maintain persistence	Application and SCADA servers	Moderate	False data	Moderate	Moderate

4.4 Deception architecture

Deception methods will be an overlay on existing systems and must cooperate with the operating system. An issue is where a deception should be implemented, for following step 5 of section 4.1. **Fig. 2** shows a generic ICS architecture with eight numbered locations, extending the three of (Lin et al. 2016).

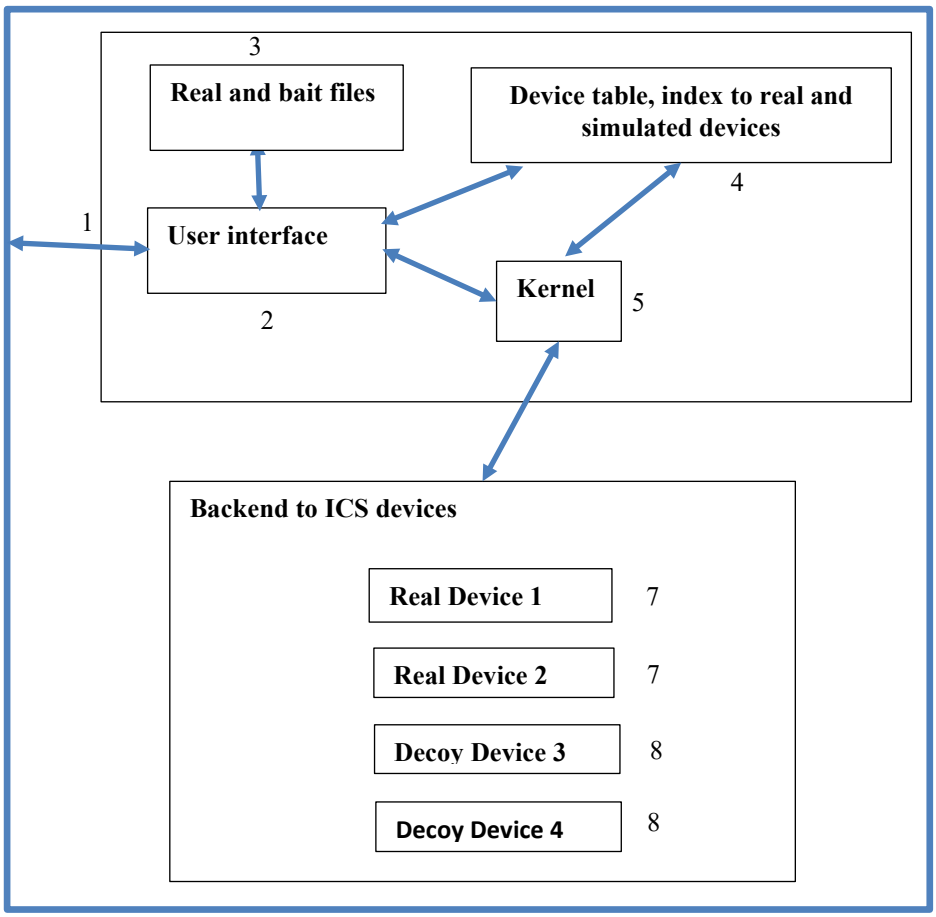

Fig. 2. Block diagram of a generic ICS with possible deception locations.

We assess the deception locations in **Fig. 2** as follows:

1. External interface: Can provide some broad deceptions such as fake networks.
2. User interface: Deceptions are relatively easy to implement here since they can imitate normal functions of the interface. On the other hand, most user interfaces are friendly, and deceptive behavior may be implausible.
3. Real and bait files: Useful as delaying tactics, though unlikely to help against advanced persistent threats that have time to study them.
4. Device table: Deceptive nodes are easy to implement if the table's device addresses include decoys or honeypots as well as regular devices. However, honeypots must be installed for each simulated device, and this may require work since overly similar simulations will be easy to spot as deceptions.
5. Kernel: Besides sending users to decoys or honeypots, the driver can implement other tactics such as delays, unnecessary passwords requests, false error messages,

or flooding of the user with information. These should not be overdone to remain plausible. This also requires modifying highly secure features of a system, and can only be done if planned at an early stage in designing a kernel. Since a kernel must be minimal, deception should be triggered by external decisions as in the user interface.

6. Interface to devices: Changing the software that interacts with backend ICSs requires modifying their drivers. This is simpler than modifying the operating system, but requires understanding of specialized software. Putting deceptions here does enable monitoring them to avoid overuse or underuse of deceptions.
7. Real devices: These are essential for operations and will keep an attacker interested, particularly an advanced persistent threat. However, the devices also must be hardened so they cannot cause major harm or easily spread an attack; and as this may require modifying vendor software, it may be difficult.
8. Decoy devices: These should simulate real systems as much as possible. Deceptions can be very effective, but require work in the design and implementation to be plausible. This may not be difficult for many devices, since some like thermostats are simple. However, timing (such as the speed of response) is important to simulate accurately as well.

4.5 Game theory for multilayered deception

Multi-layered deceptions as in section 3 can serve as effective obstacles to attacks by persistent attackers, particularly where each deception provides a different challenge, and the attacker must solve all the challenges to gain access. The tradeoff is that the more deceptions there are, the more easily they can be detected and the less effective they will be, particularly when they are similar. It will help to use significantly varied and unexpected deceptions that are hard to connect to one another. So if for instance we flood an attacker with information through the user interface to slow their attack, we can send them to decoy simulations of devices through the device driver if they manage to get there in a later phase of their attack. Simulations are conceptually different from information displays, so it may be difficult for the attacker to connect the two deceptions.

Another clue that reveals deception is inconsistency on similar tasks. So if we flood a human attacker with information on one attempt to query a device, we should also flood them with an attempt on a different device. However, inconsistency is not much noticed by automated attacks, or even a human controller reviewing their output. It will help to identify in advance the activities by the defender that most require consistency. Otherwise, maximally different responses to similar situations may prevent attackers from seeing them as an overall deception strategy.

The tradeoffs with deceptions are best planned with game theory (step 6 in section 4.1) using the conditional probability of detection of each deception given the detection of previous deceptions (Rowe et al. 2024). These conditional probabilities can be based on observations of human subjects, including non-attackers.

5 Adversary goals

Planning of discouragement deceptions can benefit from knowledge of the adversary's goals. We identify here the possible material and psychological goals, and discuss how they can be modeled.

5.1 Material goals

Material goals aim to change the physical state of an ICS. For power plants, most material goals will be based on the physics of electromagnetism; for a chemical plant, most will be based on chemistry. Some more specific material goals for ICSs than those of **Table 1** and **Table 2** are:

- Disabling the ICS hardware. This could be the overall goal of nation-state adversaries, consistent with the primary warfare objective of disabling a nation-state's ability to wage war. However, it is difficult to achieve because ICSs have many safeguards to keep them running. But successful damage can have a long-term effect much like that of munitions.
- Disabling the ICS software. This could be done by modifying software executables or possibly input data. It could have the same effect as disabling the hardware while being easier for the adversary. Operations could be restored by the defender from backup copies of software or data, although it could take time.
- Denial of service of the ICS software. This could slow operations considerably while not disabling anything. This could be useful for adversaries who want to send a political signal.
- Modifying the ICS functionality, as by changing switch settings or installing malware in controllers. For instance, the Industroyer2 attacks on Ukraine in April 2022 apparently attempted this with the IEC 104 protocol (Zafra et al. 2022). Modifications could be targeted, or it could be random to create confusion.
- Changing control parameters in the ICS. For instance, decrease power output of a generator. Effective changes require specialized knowledge of the physics or chemistry of the ICS.
- Reprogramming the ICS software to change its operation. This is difficult because it would require access to the management system, which is usually more protected than the ICS itself. It also likely requires exploiting non-ICS vulnerabilities, which are likely to be better protected than ICS vulnerabilities because of the more networks they could affect.
- Thorough reconnaissance of the ICS to enable later attacks. This could involve sending crafted packets to test responses, or systematically modifying settings to see what they do. Most of what we saw on our ICS honeypots (Meier et al. 2023) was packet traffic of this type.
- Making money. Ransomware can do this; since ICSs are often critical infrastructure, holding them hostage can be very effective. Also, some information-warfare organizations pay bounties to employees to attack sites of adversaries.

Since material goals often seek a change of a physical or chemical system, actions to accomplish them can be seen as "derivative changers", changes to the rate of some parameter, analogous to Newton's Second Law which implies that forces are velocity changers. For instance, Stuxnet attempted to increase the speed of centrifuges, and Industroyer 2 attempted to change switch states while enumerating them. The appearance of more or stronger derivative-changing commands than normal on a system, as by data injection, is suspicious and can be detected by deviations from normal behavior (Miao and Dong 2021). "Second-derivative" commands that alternate first derivatives of opposite directions, such as in turning on and off power repeatedly, can damage equipment by creating strong stresses. They can be mounted by injecting a quickly varying signal into an ICS.

While predicting the effect of derivative changes in unfamiliar domains may seem daunting, simplified models of ICS processes may suffice for deception planning. This has been called "naïve physics" for physical processes (Smith and Casati 1994), and "naïve chemistry" for chemical processes (Gollmann et al. 2015). So for instance, causing centrifuges to run much faster than normal, or flooding a chemical reaction with petrochemicals, will likely do something bad even if an attacker does not know exactly what. That is because they considerably exceed normal parameters and measures for which the processes are designed. Five subcases of "derivative attacks" can be identified:

- First-derivative attacks: These increase or decrease numeric parameters such as the applied power from correct settings to interfere with an ongoing process. An example is reducing the power on a generator so it is not supplying its required level of power, or increasing the heat in a chemical reactor. As (Gollmann et al. 2015) points out, modifying a chemical process can damage more than the equipment, possibly the chemicals produced or to the output compliance conditions about safety or pollution, and such damage can be done by changing the first derivative of a parameter at critical times.
- Second-derivative attacks: These increase or decrease a numeric parameter and then shortly thereafter change it in the opposite direction to stress a system. An example is repeatedly heating and cooling a chemical reactor to create unwanted chemicals.
- First-derivative attack with boundary condition: These increase or decrease a numeric parameter until it exceeds a safe value, such as by increasing the power supplied to a transformer until it catches fire.
- Second-derivative attack with boundary condition: These alternate increasing and decreasing a parameter until the change exceeds safety constraints, as with rapidly changing the direction of a rotating shaft until it breaks. These attacks require identification of safety limits for first derivatives.
- Transit attacks: These change directions of flow within the ICSs, such as opening valves normally kept closed in a water-treatment plant (Lucchese et al. 2023) or supplying electric current to an unusual location.

5.2 Psychological goals

Psychological goals satisfy personal and emotional needs of the attacker. These are more important with amateur attackers than paid professionals, but all attackers have them and they can often be exploited. Possible psychological goals are:

- Accomplishment of a job. Information-warfare operatives may enjoy attacking adversary countries and feel it is patriotic.
- Confidence and self-worth. Attackers like to feel they know what they are doing and can achieve results reliably and efficiently.
- Social rewards: Attackers may work in teams and enjoy the praise of their colleagues.
- Obedience to authority. Attackers may enjoy satisfying their bosses as a means of gaining self-worth or material benefits.
- Revenge for perceived slights. Terrorists are often motivated this way.
- Fantasies of power and control. Adolescents often have such needs, and entertain fantasies of controlling people as well as engaging in pointless exhibitions like petty vandalism.
- Stress reduction. If an attacker becomes frustrated with a system, they may feel better if they go away and do something else.

Psychological goals can be inferred from what is attacked and how. Some work has attempted automated inference of such goals (Shinde et al. 2021). Psychological goals do not necessarily make sense, as terrorists may be motivated by unreasonable hatreds.

5.3 Adversary mental states relevant to discouragement

Defenders can affect adversary mental goals and goal-directed behavior by influencing their emotions and inducing mental states. Some mental states defenders should consider encouraging are:

- Disappointment: Defenders want adversaries to feel disappointed after interacting with them, so they will be less likely to return. But if they are too disappointed, they may take it as a challenge and want to return with more sophisticated attacks.
- Unhappiness: Defenders want adversaries to feel unhappy when they cannot fully compromise our systems, as this will discourage them from continuing and encourage them to find easier targets. But as with disappointment, we do not want to make them so unhappy that they take that as a challenge.
- Trust/mistrust: Defenders want adversaries to trust them so adversaries are more likely to believe their deceptions. For instance, we may want adversaries to think that a resource is unavailable even when it is not. However, it may also be useful to encourage distrust in impatient adversaries looking for easy targets, who may abandon an attack site once they see evidence of deception.

- Irritation and anger: Defenders may want to irritate adversaries, as with false error messages, to show that interacting with them will be unpleasant. However, if defenders make them too irritated, they can become angry and retaliate vigorously.
- Frustration: A good way to create irritation and anger is to create unexpected obstacles in attempting to achieve an apparently easy goal. For instance, getting onto an ICS and implanting exploits is often easy with today's infrequently updated and vulnerable ICSs. Then if an attacker meets unexpected obstacles, they can become frustrated. Frustration is particularly useful to encourage in attackers as it will both discourage them from continuing and encourage them to stay away.
- Betrayal: Advanced persistent threats are expecting deception and are not fazed by it. However, many amateur attackers recognizing deception against them feel betrayed. This may cause an unpredictable range of behaviors, including terminating interactions abruptly, publicizing the deception, and redoubling attack efforts. Because of this unpredictability, encouraging feelings of betrayal is not very useful against most adversaries.
- Superiority: Many adversaries have high opinions of themselves, and it may be useful to encourage this. Then they may underestimate defenders, and be fooled by a complex multi-layered deception.
- Fear: Unlikely to be induced because many adversaries have high opinions of themselves and their skills.
- Sanguinity: Not useful for discouragement because we generally want attackers to feel unhappy, but it could be useful for an encouraging honeypot.

5.4 Psychological adversary variables

Mental states of adversaries can be inferred by their actions. For instance, if the adversary can elevate privileges of an account or change switch settings, they have high confidence in their abilities and low frustration; if they log out after a short session, they have low confidence and high frustration; or if they have ten times opened fake files and discovered only gibberish, they have a high threshold for frustration and may be professionals of an intelligence-gathering organization that requires them to search exhaustively. Hidden Markov models and reinforcement learning are classic machine-learning models for inferring mental states, and a variety of neural-network architectures can be trained to infer them as well.

Although sections 5.1 and 5.2 listed a variety of attacker goals, the number of mental states relevant to achieving them is more limited. We have thus argued (Landsborough, et al. 2024) that they can be modeled by a small set of "adversary variables" representing key aspects of the mental state of an adversary that affect their actions, and these variables can be used to plan the deceptions most likely to succeed against them. **Table 3** shows our proposed adversary variables. These are used at steps 3, 4, and 5 of the deception-planning procedure of section 4.1. The rightmost column represents possible predictions when the deceivee measures high on the adversary variable. However, predicting the actual outcome requires estimating costs of the options for the deceivee such as time wasted on fruitless tasks, and using a decision tree to estimate the weighted cost of options. The table should apply to automated attacks as well, since after finding a

rare vulnerability in a system, human judgment is necessary to decide what to do next, and then the variables apply to that human.

Table 3. Adversary variables useful for predicting mental states of an adversary.

Name of adversary variable	How inferred	Correlations to other variables	Predictive use
Sophistication of the adversary	Ratio of advanced actions to basic actions	Lack of surprise	Superiority
Confidence of the adversary in their methods	Consistency in following a plan without digressions	Sophistication	Trust
Interest more in intelligence gathering than sabotage	Ratio of sabotage-related actions	None	Rate of attempts to modify the system
Adversary estimate of reliability of the target system	Number of anomalous events observed by user	Sophistication, alertness	Acceptance of deceptions as unreliability
Trust in information shown about the attack target	One minus ratio of redundant confirming actions to normal actions	Sophistication, alertness	Likelihood of ending the interaction
Alertness of adversary to the target	Whether they notice obvious inconsistencies	Sophistication	Ineffectiveness of deceptions
Surprise of the adversary at the target	Increase in idle time	Lack of sophistication and confidence	Likelihood of ending the interaction
Adaptability of adversary to the target	How well they find alternatives to obstacles created by deceptions	Sophistication, alertness	Ineffectiveness of deceptions
Impatience of the adversary	Count of steps after encountering obstacles	Frustration, lack of confidence, desire to disconnect	Likelihood of ending the interaction
Desire of adversary to disconnect without achieving their goals	Fraction of connections ended without achieving goals	Impatience	Choice of deceptions
Frustration level of the adversary	Ratio of unnecessary sabotage to normal actions	Lack of trust	Increased length of interaction
Interest of the adversary in financial gain	Ratio of sabotage to normal actions, and sending ransom notes	Confidence, sophistication	Length of reconnaissance

6 Deceptive tactics for foiling adversary goals

(Rowe and Rrushi 2016 chapter 4) lists a range of defensive deceptions against general cyberattacks. The options for ICSs can be rated differently since the ICS environment usually tries to maintain ongoing processes rather than do new things. Some good tactics for ICSs for accomplishing steps 5-7 of the deception planning in section 4.1, roughly in order of decreasing usefulness, are:

- T1, false displays of system state: If an attacker tries to do something dangerous, it can be a useful deception for the defender to merely simulate what happens. For instance, increasing the power in many processes could cause overheating, which can be simulated with increases on a thermostat icon. Most industrial processes have a "digital twin" simulation for testing and analysis, and a display of the twin could deceive the attacker. False displays can be effective for ICSs because their specialized nature (Rowe and Rrushi 2016 chapter 11) means inconsistencies in a simulation are more likely to be missed by most attackers. A particularly useful false display is for a target system to imitate an obvious honeypot, since attacks try to avoid honeypots (Maesschalck et al. 2024).
- T2, fake controls: Attackers want to control systems, so an interface they can manipulate as in (Ramirez et al. 2022) is appealing. It can look like the controls of a real ICS, with knobs and switches that appear to change the ICS, though they do not. They can also log what the attacker does for future analysis.
- T3, fake resources ("decoys"): These include useless nodes, processes, and files to waste attacker time. ICSs tend to have many nodes and features, and it is easy to make some extra ones. Fake resources can also be generated dynamically to answer attacker interests, as a more active defense (Yang et al. 2020). Files can be made more realistic by simulating plausible resource use (Sutton et al. 2019).
- T4, false error messages: These can be used as excuses not to do something dangerous. Attackers can take error messages seriously, particularly for an unfamiliar and specialized system like an ICS, since they may mean goals are unachievable (Rowe and Rrushi 2016 chapter 9). False error messages can also thwart automated attacks because they cause unexpected interruptions, made worse if followed by requests to do new things for which the automated attack is unprepared.
- T5, fake crises: These can be triggered by defenders by interjecting messages that an intruder has been detected, or reporting that the ICS is out of control. This is useful in fooling intended saboteurs, but is too dramatic to be used routinely.
- T6, flooding the adversary with data: This is denial of service in reverse. It can be done by generating fake data or using historical data with dates changed. It is useful against sophisticated attackers who may waste time trying to understand the data.
- T7, obfuscatory controls: An interface can be made hard to understand so that the attacker cannot usefully control it. Controls can be poorly labeled with code names and numbers, so an attacker has few clues about how to proceed. Real systems designed for a limited set of specialized users often lack adequate labeling, and many ICSs require specialized technical knowledge to understand, so incomprehensible

interfaces are plausible. The interface can also require confusing preconditions before anything happens.

- T8, misleading controls: A deception unique to ICSs is to have the knobs and switches do something different from expected. A knob labeled “power” could decrease power if turned clockwise, contrary to the usual expectation, or switches labeled “on/off” could be inconsistent in which direction is “on”.
- T9, process interruptions and delays: The system can stop responding for random periods of time for no obvious reason, or can respond very slowly as in “tarpits”. This will confuse the attacker or encourage them to think they have disabled the system.
- T10, decoy financial records: These encourage adversaries who seek monetary gain.

These deceptions work best when tied to high-priority adversary goals. Some examples:

- Adversaries want a feeling of control, and many want to disable things. So simulate disablement events that the attackers likely want. An example would be to simulate the turning off an entire power plant by terminating all protocols connecting to it. It could be made further convincing if preceded by dramatic fluctuations of the dials and lights of the display.
- Adversaries want to do reconnaissance, and they may get bounties for the nodes that they find. So simulate an ICS network with so many nodes that it will take a long time to explore, perhaps even an unbounded time if nodes are generated whenever an adversary queries a new address. Most nodes can be decoys to confuse the adversary; this might seem suspicious, but will not impede an automated attack.
- To frustrate adversaries who need to feel in control and having achieved persistence on systems, simulate an unreliable system with inconsistent behavior to give them a sense of failure to control things. For instance, terminate access to resources randomly, and ask for passwords repeatedly.
- Similarly, to frustrate adversaries who need to feel in control, provide complex visual displays that are difficult to decipher and waste their time. Use ambiguous labels and abbreviations to require adversaries to use considerable trial and error to accomplish anything.

Another consideration in choosing deceptions is their compatibility with high and low values of the adversary variables. **Table 4** summarizes the compatibility of these deception tactics to the adversary variables in **Table 3**.

Table 4. Compatibility of deception tactics with high and low values of the adversary variables.

Adversary variable	Deception tactics for high values of this variable	Deception tactics for low values of this variable
Sophistication of the adversary	T6, T7, T8, T9	T1, T2, T3, T4, T5
Confidence of the adversary in their methods	T1, T2, T3, T4, T8	T5, T6, T7
Interest of adversary in intelligence gathering versus sabotage	T3, T6, T10	T4, T5, T7, T8
Adversary estimate of reliability of the target system	T1, T2, T3, T4	T8, T9
Trust in information shown about the attack target	T1, T2, T3, T4, T5, T6	T8, T9
Alertness of adversary to the target	T6, T7, T8, T9	T1, T2, T3, T4, T5
Surprise of the adversary at the target	T3, T4, T6	T1, T2, T3, T4, T10
Adaptability of adversary to the target	T6, T7, T8, T9, T10	T1, T2, T3, T4, T5
Impatience of the adversary	T4, T5, T6, T7, T8, T9	T1, T2, T3
Desire of adversary to disconnect without achieving their goals	T4, T5, T6, T7, T8, T9	T1, T2, T3
Frustration level of the adversary	T4, T5, T6, T7, T8, T9	T1, T2,T3
Interest of the adversary in financial gain	T3, T10	

7 Reinforcement learning of adversary variables

Choosing among these options can try to optimize defender benefits by assigning costs and benefits to both attacker and defender tactics, and then using decision theory (if the attack plan is fixed), game theory (La et al. 2016; Rowe et al. 2024), or stochastic modeling (Betancourt et al. 2022) to determine the best tactics. Deception methods are also just one kind of active-defense method; alternative methods to consider are moving-target defenses than modify configurations and software (Ma et al. 2022) and active searching for attackers (Ajmal et al. 2021).

Our experience with real ICS honeypots has been that most malicious activity is exploratory (Ramirez et al. 2022), and can be deceived by simple methods like random error messages. Defensive planning is primarily useful against the rarer sophisticated adversary who has time to make a plan specific for the target site, and can adjust to our attempts to deceive. For such adversaries, reinforcement learning is a simple way to dynamically find their weaknesses (as step 8 in the deception plan of section 4.1). For instance, many Chinese attacks attempt espionage; so to deceive them, offer them plenty of what looks like valid intelligence, and see if they like it by looking for more. Many Russian attacks attempt to sabotage; so to deceive them, give them plenty of

evidence of suddenly crashed processes after malicious-appearing adversary commands, and see if they try to do more.

After deceptions, reinforcements can also update the estimates of the adversary variables based on adversary choices and deception success or failure. Reinforcement learning also has the advantage that it is self-correcting. After a false error message for instance, we could decrease the estimated perception by the adversary of the reliability of the system, and increase the estimated frustration of the adversary. It simplifies matters to treat adversary variables as probabilities on a range of 0 to 1. To keep variables in that range, we can ignore increments that would force them outside the range. Alternatively, a common method is to set the probability of choice p to $c_D * \mathrm{p}$ for a decrease and to $\mathrm{p} + c_D * (1 - \mathrm{p})$ for an increase, where p is the value of the adversary variable, D is a particular deception or system response, and c_D is the reinforcement constant specific to deception D.

For ICSs and a deception plan focused on discouragement, a high rating should be given to deceptions that caused an attack to stop. Lesser but positive ratings should be given to deceptions that cause an attack to waste time, as by retrieving bait files or visiting decoy network nodes, with the rating proportional to the time wasted (Reti et al. 2022). Negative ratings should be given to deceptions which the attacker ignored. From these ratings we should subtract the values to the attacker of achieving partial goals, such as legitimate nodes they have found, legitimate documents they have stolen, and how much they have disrupted actual operations. Data to estimate these values can be collected from intrusion-detection systems, log files, and packet captures.

Reinforcements can be applied with attenuation to deceptions earlier in the attacker's session. For instance, if the adversary disconnected after a second error message about network availability, this should reinforce the decision to issue a first error message about it, so that the first error message should be issued more often by the defender in similar situations in the future. However, it is usual that the reinforcement should be attenuated by multiplying by it by a fraction for each step backward since earlier actions are less related to outcomes. Multiple reinforcements to the same action can be averaged, so that if at one time the defender's deception worked well and at one time it worked badly, the average will assign a more neutral effect to the deception. Reinforcements can be averaged over all attacks so very different attacks will still generate separate reinforcements for the deceptions that worked with them.

Data for reinforcement learning can be enhanced with "data farming". This means generating synthetic data based on real data and training with it. (Haynes et al. 2023) tried this method by generating new variants of ICS attacks for Log4j and IEC 104 permutation attacks using an evolutionary algorithm. To implement this in a more general way, we need a simulation of the attacker-defender game that we can run many times to generate automated outcomes. We need some starting data of attacker behavior from which to enumerate a set of attacker actions. We then create sequences of those actions that we have not seen previously, choose random sets of defender actions, and rate the outcomes based on the achievement of defender goals minus the achievement of attacker goals. This then creates new data for reinforcement learning.

A secondary issue affecting planning of multilayer deceptions is how often deceptions should be used. Adversaries that are high on the sophistication parameter such as

advanced persistent threats will be expecting obstacles including deceptions, so we can use as many deceptions as we like without expecting them to react differently, since they are low on the "trust" parameter concerning us. But amateur attackers may experience a major change to the trust variable when they realize we are deceiving them, and they may redouble their efforts or seek revenge against us. We should try to avoid this situation because it may lead to new circumstances against which we are unprepared to defend, so we may not want to use multilayered deceptions against such attackers.

8 Case study: Experiments with building maintenance systems

8.1 The equipment

We have recently been exploring these ideas in the context of heating, ventilation, and air-conditioning (HVAC) systems for buildings, a type of ICS underestimated for security vulnerabilities even though its manipulation can make a building uninhabitable and disable its computer systems. We have been working with our Facilities Management department to learn about their controls. They have centralized control over all the buildings on our campus, using the BACnet protocol for the lower levels of device control.

To foil attacks on facilities, a honeypot designed for discouragement is appropriate. We are currently experimenting with an Automated Logic fan-coil unit (Automated Logic n.d.), obtained as excess hardware from Facilities Management, which exemplifies their hardware and software. The unit contains an actuator, a fan motor, a temperature sensor, a temperature sensor, and a thermostat. It is controlled by a Web-based graphical user interface WEBCTRL (Automated Logic 2019) that shows the state of the equipment, and it reports data to a centralized collection point using the BACnet-over-IP protocol.

We could block all unauthorized attempts to control the fan, but this would give an attacker quick feedback and enable them to switch their targeting to a more vulnerable device. A better way to control attackers would be to simulate cooperation with attacks by false representations in the graphical user interface. For instance, if an attacker tries to turn off a running fan, we should just simulate turning it off by substitution of a visualization of a nonrotating fan in the interface, while keeping the fan actually on (Colvin 2023). Similarly, if an attacker tries a second-derivative attack by turning on and off the fan repeatedly to try to damage it, we can simulate it stopping and no longer responding to commands. Appearing to cooperate with user sabotage first plays to their sense of power as discussed in section 5.2, and then increases their frustration if they discover later that their actions actually failed. People are unlikely to question what graphics shows them, and are more likely to blame themselves for failures.

Similar honeypot ploys can be used for heating and air-conditioning units where attackers will have similar goals such as increasing the heat or cooling rate. We can use naïve-physics models (Smith and Casati 1994) to simulate the temperature change when following attacker commands; all we need for deception is a temperature gauge showing the simulated temperature, while in reality the temperature remains the same.

8.2 Example reinforcement-learning scenario

Fig. 3 shows part of the state diagram for an attacker-defender game on the equipment described in the last section, assuming the attacker has access to the controller machine which displays a table of devices and their associated measures. We assume there is a fan, a temperature gauge, and a thermostat.

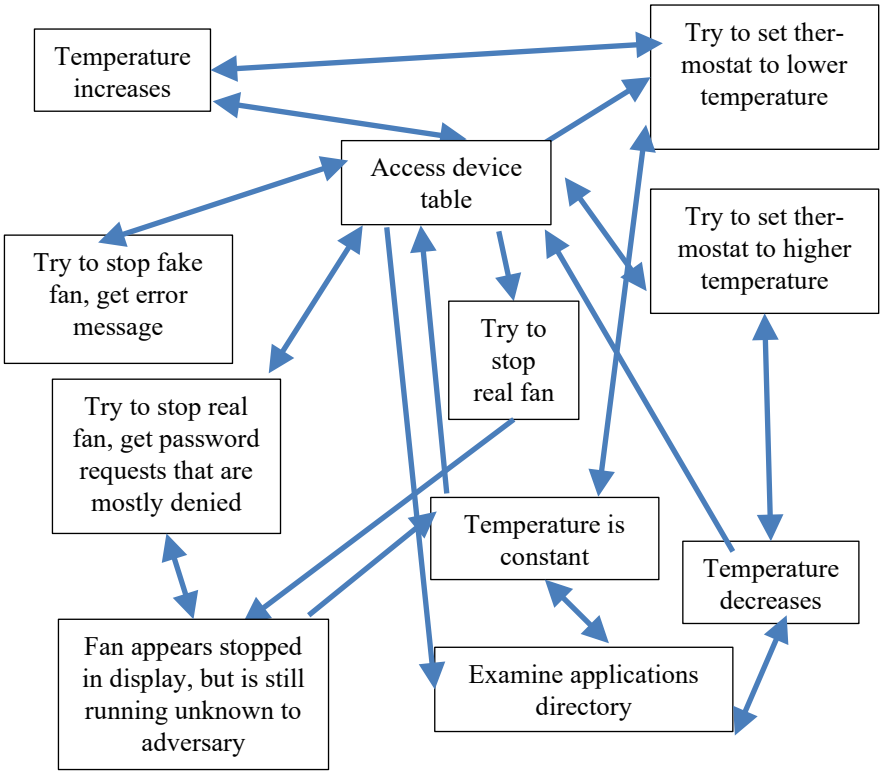

Fig. 3. State diagram for some deceptive interactions with a fan, thermostat, and temperature gauge in an HVAC (heating, ventilation, and air-conditioning) system.

For a demonstration, we will use only the adversary variables given in **Table 5**, chosen from the variables in **Table 3**, with some example initial assignments.

Table 5. Parameters in the HVAC example with demonstration initial values.

Symbol	Parameter	Initial value for Table 6
p_d	Probability that adversary disconnects without achieving their goals	0.02
p_c	Probability that adversary tries to confirm display data	0.01
p_t	Probability adversary trusts data shown	0.05
p_s	Probability adversary is sophisticated	0.30
p_k	Probability adversary is confident about their methods	0.50
p_a	Probability that adversary is alert	0.30
p_r	Probability that adversary thinks the system is reliable	0.50
p_f	Probability that adversary is frustrated	0.01

We can update these values using a table of reinforcements applied to the adversary model, giving increments to parameters due to events (**Table 6**). The increment should be ignored if it makes the probability greater than 1 or less than 0. We have used plausible increments here, but they can be fit to data from real adversaries.

Table 6. Proposed reinforcements to adversary variables based on adversary events.

Event	p_d	p_c	p_t	p_s	p_k	p_a	p_r	p_f
Get error message	.01	.05	.01	.00	-.10	.10	.00	.03
Command fails	.03	.05	.02	.00	-.10	.05	-.05	.10
Command had opposite effect of expected	.07	.10	.05	.00	-.10	.05	-.05	.05
Nothing unusual found in examining display	.00	.00	.02	.02	.00	.03	.02	.00
Something unusual found in examining display	.10	.00	.02	.05	.00	.03	-.05	.00
Unusual response delay	.10	.00	-.05	.00	-.05	.00	-.02	.05
"Suspicious behavior" noted by system	.10	.00	-.10	.00	-.10	.05	-.05	.10
Adversary leaves	.10	-.05	.00	.00	-.20	.00	.00	.10

As a demonstration of how reinforcement works, **Table 7** gives an example interaction with deceptions based on **Fig. 3**, showing how its steps affect the adversary variables.

Table 7. Modifications to adversary variables for an example sequence of adversary actions with some defensive deceptions.

Attacker event and result	p_d	p_c	p_t	p_s	p_k	p_a	p_r	p_f
Start	.02	.01	.05	.30	.50	.30	.50	.01
Attacker views the device table, sees nothing interesting	.02	.01	.07	.32	.50	.33	.52	.01
Attacker sets the thermostat to a higher temperature; temperature decreases in device table, an unexpected result	.09	.11	.12	.32	.60	.38	.47	.06
Adversary tries to stop the fan; no effect in the display	.12	.16	.14	.32	.50	.43	.42	.16
Adversary tries to stop the fan a second time, gets long delay	.22	.16	.09	.32	.45	.43	.35	.21
Adversary gets password requests, system notes "suspicious behavior"	.32	.16	.00	.32	.35	.48	.32	.31
Attacker leaves	.42	.11	.00	.32	.15	.48	.32	.41

Observing the last row of the table, the overall effect of this sequence of events is to leave the attacker frustrated and thinking they have been deceived, less confident of themselves, but still thinking they are sophisticated because they did provide a good test of the system. If we see the attacker's IP address again, or an address in the same subnetwork, it will be good to avoid deceptions involving fakes (T1, T2, T3, T4, and T5 of section 6) since we inferred that this attacker is aware of deception and may tell colleagues in their organization. Instead, we could prefer tactics of flooding the attacker with data, obfuscatory controls, misleading controls, and process interruptions (T6, T7, T8, and T9).

9 Conclusions

The cybersecurity problems of industrial control systems differ in important ways from those of most information systems. This means their defensive tactics should be different, including their deceptive defenses, which should be especially sensitive to the environment in which they occur. We have identified a range of tactics in this chapter for better design of multilayered defensive deception methods for industrial control systems. These methods infer a set of "adversary variables" from adversary actions, which can be used to predict subsequent adversary activity. These can be dynamically modified by using reinforcement learning from attacker behavior over an extended period of interactions with a range of attackers. We have provided a variety of design tools that can provide interesting and varied deceptions for a range of attackers. Testing these tools is our next step, but testing is difficult since attackers are deliberately uncooperative, and testing against known attacks does not capture the dynamics of real attack campaigns. Nonetheless, much as when artificial neural networks receive repeated

feedback to improve their weights, and they can converge to surprisingly powerful consensus models, we expect our multiple methods of feedback can allow us to craft powerful defensive campaigns from repeated exposure to attacks.

Acknowledgements

This work was supported by the U.S. Department of Energy and the Defense Intelligence Agency. Thuy D. Nguyen, Julian L. Rrushi, Scott D. Colvin, and Angela Tan contributed ideas. Opinions expressed are those of the author and do not represent those of the U.S. Government.

References

1. Abe S, Tanaka Y, Uchida Y, Horata S (2018, July) Developing deception network systems with traceback honeypot in ICS network. ISCE Journal of Control, Measurement, and System Integration, 11(4): 372-379
2. Ackerman P (2017) Industrial cybersecurity. Birmingham, UK, Packt.
3. Ajmal A, Alam M, Khaliq A, Khan S, Qadir Z, Mahmu M (2021, September) Last line of defense: reliability through inducing cyber threat hunting with deception in SCADA networks. IEEE Access, 9.
4. Atadika M, Burke K, Rowe N (2019, August) Critical risk management practices to mitigate cloud migration misconfigurations. Proc. Intl. Conf. on Computational Science and Computational Intelligence, Las Vegas, NV, USA
5. Automated Logic (2019, May 9) WebCTRL V7.0. https://www.mindmeister.com/generic_files/get_file/10093951?filetype=attachment_file
6. Basan A, Basan E, Korchalovsky S, Bikhailova V, Ivannikova T, Shulika M (2022, November) The concept of the knowledge base of threats to cyber-physical systems based on the ontological approach. Proc. IEEE International Multi-Conference on Engineering, Computer, and Information Sciences, Novosibirsk-Yekaterinburg, RU.
7. Betancourt J, Gonzalex GP, Gonzalex SR, Cuellar , Gomez, C, Mariotti F, Montecchi L, Lollini P (2022, November) Modeling attacker behavior in cyber-physical-systems. Proc. Latin-American Symposium on Dependable Computing, Fortaleza/CE, BR
8. Cifranic N, Romero-Mariana J, Souza B, Hallman R (2020) Decepti-SCADA: A framework for actively defending networked critical infrastructures. Proc. 5th International Conference on Internet of Things, Big Data and Security - Volume 1: IoTBDS, 69-77
9. Clark R, Mitchell W (2019) Deception: counterdeception and counterintelligence. Sage Publishing.
10. Cohen F (1999) A mathematical structure of simple defensive network deceptions. all.net/journal/deception/mathdeception/mathdeception.html, accessed 15 Jan 2016.
11. Colvin S (2023, December) Implementing a high-interaction hybrid honeypot for facility automation systems. M.S. thesis, U.S. Naval Postgraduate School.
12. Dodson M, Beresford A, Vingaard M (2020) Using global honeypot networks to detect targeted ICS attacks. Proc. 12th International Conference on Cyber Conflict.
13. Dutta N, Jadav N, Dutiya N, Joshi D (2020) Using honeypots for ICD threats evaluation. In E. Pricop et al., (eds.), Recent Developments on Industrial Control Systems Resilience, Studies in Systems, Decision, and Control 255, Springer Nature, 175-196.

14. Foley B, Rowe N, Nguyen T (2022, December) Analyzing attacks on client-side honeypots from representative malicious Web sites. Proc. International Conference on Computational Science and Computational Intelligence
15. Franco J, Aris A, Canberk B, Selcuk A (2021) A survey of honeypots and honeynets for Internet of Things, industrial Internet of Things, and cyber-physical systems. IEEE Communication Surveys and Tutorials.
16. Gollmann D, Gurikov P, Isakov A, Krotofil M, Larsen J, Winnicki A (2015, April) Cyber-physical systems security – experimental analysis of a vinyl acetate monomer plant. CPSS '15: Proceedings of the 1st ACM Workshop on Cyber-Physical System Security, Singapore, SP
17. Han X, Kheir N, Balzarotti D (2018, July) Deception techniques in computer security: a research perspective. ACM Computing Surveys, 51(4) article 80.
18. Haynes N, Nguyen T, Row, N (2023, January) Creating synthetic attacks with evolutionary algorithms for proactive defense of industrial control systems. Proc. Hawaii Intl. Conf. on Systems Sciences, Maui, HI, US.
19. Kayan H, Nunes M, Rana O, Burnap P, Perera C (2022, September) Cybersecurity of industrial cyber-physical systems: A review. ACM Computing Surveys, 54(11s), Article 229
20. La Q, Quek T, Lee J, Jin S, Zhu H (2016, December) Deceptive attack and defense game in honeypot-enable-networks for the Internet of Things. IEEE Internet of Things Journal, 3(6)
21. Landsborough J, Nguyen T, Rowe N (2024, January) Retrospectively using multilayer deception in depth against advanced persistent threats. Proc. Hawaii Intl. Conf. on System Sciences, Hawaii, HI
22. Lin H, Alemzadeh H, Chen D, Kalbarczyk Z, Iyer R (2016, April) Safety-critical cyber-physical attacks: Analysis, detection, and mitigation. Proc. Symposium on the Science of Security, Pittsburgh, PA, US
23. Lucchese M, Lupia F, Merro M, Paci F, Zannone N, Furfaro A (2023, August) HoneyICS: A high-interaction physics-aware honeynet for industrial control systems. Proc. 18th Intl. Conf. on Availability, Reliability, and Security, Benevento, IT, Article 113.
24. Ma D, Tang Z, Sun X, Guo L, Wang L, Chang K (2022, November) Game theory approaches for evaluating the deception-based moving target defense. Proc. 9th Workshop on Moving Target Defense, Los Angeles, US
25. Maesschalck S, Giotsas V, Race N (2021, December) World wide ICS honeypots: A study into the deployment of Conpot honeypots. Seventh Annual Industrial Control Systems Security Workshop, Austin, TX US
26. Maesschalck S, Fantom W, Giotsas V, Race, N (2024, June). These are not the PLCs you are looking for: Obfuscating PLCs to mimic honeypots. IEEE Transactions on Network and Service Management, 21(3), 3623-3635.
27. Meier J, Nguyen T, Rowe N (2023, January) Hardening honeypots for industrial control systems. Proc. Hawaii Intl. Conf. on Systems Sciences, Maui, HI.
28. Miao P, Dong L (2021, July) Attack effect observer-based security control for cyber-physical systems subjected to false data injection attack. Proc. 40th Chinese Control Conference, Shanghai, CN
29. Morales E, Rubio-Medarno C, Doupe A, Shoshitsishvili Y, Wang R, Bao T, Ahn GJ (2020, October) HoneyPLC: a next-generation honeypot for industrial control systems. Proc. ACM SIGSAC Conf. on Computer and Communications Security, 279-291
30. Pawlick J, Colbert E, Zhu Q (2019, August) A game-theoretic taxonomy and survey of defensive deception for cybersecurity and privacy. ACM Computing Surveys, 52(4), Article 82

31. Qassim Q, Jamil N, Mahdi M, Rahim A (2020, August) Towards SCADA threat intelligence based on intrusion detection systems – A short review. Proc. 8th Intl. Conf. on Information Technology and Multimedia, Selangor, Malaysia, 144-149.
32. Ramirez R, Nguyen T, Rowe N, Meier J (2022, December) Classifying RDP remote attacks on user interfaces to industrial control systems. Proc. Intl. Conf. on Computational Science and Computational Intelligence, Research Track on Cyber Warfare, Cyber Defense, and Cyber Security.
33. Reti D, Elzer K, Fraunholz D, Schneider D, Schotten H (2022, November) Evaluating deception and moving target defense with network attack simulation. Proc. Workshop on Moving Target Defense, Los Angeles, CA, US
34. Rowe N (2019) Honeypot deception tactics. Chapter 3 in Al-Shaer E, Wei J, Hamlen K, Wang C (Eds.), Autonomous Cyber Deception: Reasoning, Adaptive Planning, and Evaluation of HoneyThings, Springer, Chaum, Switzerland, 35-45
35. Rowe N (2024) Cyber deception. Chapter in Encyclopedia of Cryptography, Security, and Privacy, 3rd edition, ed. Jagodia S, Samarati P, Young M, Springer Berlin, Heidelberg. doi.org/10.1007/978-3-642-27739-9_1766-1
36. Rowe N, Green J, Benn A, Drew S, Heinen C, Bixler R, Tan A, and Barton A (2024) Game-based testing for active cyberdefense and cyberdeception. Chapter in Cybersecurity – Cyber Defense, Privacy and Cyberwarfare, ed. Dimitoglou A, Deligiannidis L, Arabnia, H, Berlin, DE: De Gruyter
37. Rowe N, Nguyen T, Dougherty J, Bieker J, Pilkington D (2021, September) Identifying anomalous industrial-control-system network flow activity using cloud honeypots. Springer Lecture Notes, Proc. National Cyber Summit, Huntsville, Alabama.
38. Rowe N, Nguyen T, Kendrick M, Rucker Z, Hyun D, Brown J (2020, January) Creating effective industrial-control-systems honeypots. Proc. Hawaii Intl. Conf. on Systems Sciences, Wailea, HI
39. Rowe N, Rrushi J (2016) Introduction to cyberdeception. Berlin: Springer International Publishing.
40. Rrushi J (2022, May) Physics-driven page fault handling for customized deception against CPS malware. ACM Transactions on Embedded Computing Systems, 21(3), 1-36
41. Shinde A, Doshi P, Setayeshfar O (2021, May) Cyber attack intent recognition and active deception using factored interactive POMDPs. Proc. Intl. Conf. on Autonomous Agents and Multiagent Systems.
42. Smith B, Casati R (1994) Naive physics: An essay in ontology. Philosophical Psychology, 7(2) 225–244. doi:10.1080/09515089408573121.\
43. Sohl E, Fielding C, Hanlon T, Rrushi J, Farhangi H, Carmichael K., Dabell J (2015, October) A field study of digital forensics of intrusions in the electrical power grid. Proc. First ACM Workshop on Cyber-Physical Systems-Security and/or Privacy
44. Sutton A, Tahiri S, Bond B, Rrushi J (2019, December) Countering malware via decoy processes with improved resource utilization consistency. Proc. IEEE International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications
45. Urias V, Stout W, Van Leeuwen B (2018, October) On the feasibility of generating deception environments for industrial control systems. Proc. 2018 IEEE International Symposium on Technologies for Homeland Security, Woburn, MA, US
46. Yahya M, Sharaf N, Rrushi J, Tay H, Liu B, and Xu K (2020, December) Physics reasoning for intrusion detection in industrial networks. Proc. IEEE International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications.
47. Yang D, Mashima D, Lin W, Zhou J (2020, October) DecIED: Scalable k-anonymous deception for IEC61850-compliant smart grid systems. Proc. CPSS, Taipei, Taiwan.

48. You J, Lv S, Zhao L, Niu M, Shi Z, Sun L (2020, November) A scalable high-interaction physical honeypot framework for programmable logic controller. Proc. IEEE 92nd Vehicular Technology Conference, Victoria, BC CA
49. Zafra D, Leong R, Sistrunk C, Proska K, Hildebrandt C, Lunden K, Brubaker N (2022, April 25) Industroyer.V2: Old malware learns new tricks. https://www.mandiant.com/resources/blog/industroyer-v2-old-malware-new-tricks