A Developmental Approach to Learning Causal Models for Cyber Security

Jonathan Mugan

21CT, Inc., Austin, Texas, USA
www.21ct.com

ABSTRACT

To keep pace with our adversaries, we must expand the scope of machine learning and reasoning to address the breadth of possible attacks. One approach is to employ an algorithm to learn a set of causal models that describes the entire cyber network and each host end node. Such a learning algorithm would run continuously on the system and monitor activity in real time. With a set of causal models, the algorithm could anticipate novel attacks, take actions to thwart them, and predict the second-order effects of those actions. Designing such an algorithm is a complex task because computer systems generate a flood of information, and the algorithm would have to determine which streams of that flood were relevant in which situations.

This paper will present the results of efforts toward the application of a developmental learning algorithm to the problem of cyber security. The algorithm is modeled on the principles of human developmental learning and is designed to allow an agent to learn about the computer system in which it resides through active exploration. Children are flexible learners who acquire knowledge by actively exploring their environment and making predictions about what they will find,[1,2] and our algorithm is inspired by the work of the developmental psychologist Jean Piaget.[3] Piaget described how children construct knowledge in stages and learn new concepts on top of those they already know. Developmental learning allows our algorithm to focus on subsets of the environment that are most helpful for learning given its current knowledge. In experiments, the algorithm was able to learn the conditions for file exfiltration and use that knowledge to protect sensitive files.

Keywords: machine learning, automation, automation assurance, autonomic computing, verification and validation, cyber resilience, causal models

1. INTRODUCTION

Our current cyber defenses are insufficient to counter the sophistication level of modern attacks.[4,5] Signature-based detection is too brittle and results in too many false negatives.[6] Attempts have been made to move into machine learning and anomaly based approaches,[7] but anomaly based detection is often too myopic and results in too many false positives,[6] and classification-based machine learning can only find what it is told to look for. We must expand the scope of machine learning to address the breadth of possible attacks.

An expanded machine learning algorithm could learn a causal model to represent the entire cyber network and each host end node. Such a learning algorithm would run continuously on the system and monitor activity in real time. With a causal model of the system, the algorithm could anticipate novel attacks, take actions to thwart them, and predict the second-order effects of those actions. Designing such a learning algorithm is a complex task because computer systems generate a flood of information, and the learning algorithm would have to determine which streams of that flood are relevant in which situations. Additionally, the causal mechanisms of a system cannot be learned through observation alone.[8] This idea is highlighted by the old maxim that observing that prison inmates tend to have tattoos does not mean that tattoos cause crime. To learn a useful causal model out of the flood of network information, we present Cy-QLAP. Cy-QLAP (patent pending[9]) is an extension of the Qualitative Learner of Action and Perception, QLAP,[10,11] to protect cyber assets.

Send correspondence to Jonathan Mugan at jmugan@21ct.com. PA Approval Number: 88ABW-2013-1346. Approved Date: 2013-03-20 10:42:35.

www.manaraa.com

Cy-QLAP uses a developmental learning approach. Learning developmentally allows an algorithm to focus on subsets of the environment that are most helpful for learning given its current knowledge. Cy-QLAP also learns by actively exploring the environment. Much as randomized, controlled trials do in science, learning through direct actions allows Cy-QLAP to separate causality from correlation. Since Cy-QLAP is a learning algorithm, it can anticipate scenarios beyond those that were programmed in. Cy-QLAPs actions can naturally lead to subtle responses such as moving a file that is trying to be maliciously accessed or diverting requests to a honeypot. Expanding the scope of machine learning to learning system-wide causal models is an ambitious research effort, but cyber defense approaches to date have shown to be insufficient. The vision we are working toward is a parallel system that runs on top of our computing systems to keep them safe.

The state of the art for cyber security consists of firewalls, antivirus and anti-malware systems, intrusion detection systems (IDS), and intrusion prevention systems (IPS). Intrusion detection systems can be network based or host based. A host-based IDS watches activity on a host and warns an administrator if it suspects malicious activity. An IDS can identify malicious activity by using a set of signatures or by looking for anomalies in behavior. An IPS is just like an IDS, with the addition of taking actions to protect the system, such as blocking a system call that is suspected to be malicious. For example, Rootsense[6] is an intrusion prevention system that correlates events between processes, the file system, memory, and the network. When an application invokes a system call, Rootsense can determine if the process is malicious and terminate it or deny the system call.

Machine learning has been deployed because signature-based detection is static and therefore cannot detect new attacks, resulting in too many false negatives. There is classification-based machine learning that focuses on deciding if something is malicious or not. Siddiqui, Wang, and Lee[12] apply machine learning to disassembled programs to determine if a particular program is a worm. Machine learning can also include anomaly detection. Apap et al.[13] use machine learning (anomaly detection) to determine if a registry access is malicious or not. Existing IDS methods often lead to too many false positives and uncorrelated messages and alerts. The next step, therefore, is to employ machine learning using the alerts generated by an intrusion detection system. To this end, Treinen and Thurimella[14] learn association rules on top of alerts from IDS systems. Their goal was to reduce the burden of false positives from IDS systems. Zurutuza et al.[15] proposed a method for data mining on alerts detected from intrusion detection systems. The goal was to make alerts more meaningful by abstracting them into meta-alerts. They extract fields from alerts that they use for clustering, and they distill each cluster down to an association rule that is labeled as malicious or not.

Cy-QLAP is a next step in this evolution of defenses. While most systems try to identify attacks in progress by watching for anomalies or matching signatures, Cy-QLAP seeks to stay a step ahead by proactively finding vulnerabilities before an attack occurs. Cy-QLAP actively fills in the gaps in its knowledge by asking what would happen if the adversary performed such an action, and then trying it out. Cy-QLAP can then actively protect the system. For example, if the system is currently in state x, and through causal model learning, Cy-QLAP knows that action a would lead to bad state y, Cy-QLAP can prevent action a from being performed. Because Cy-QLAP has learned this causal model, an attack such as this can be thwarted even if it was not previously conceived of by an analyst.

Many machine learning approaches treat the learning algorithm like a black box that receives input and produces output. Neural networks are an excellent example of this opacity in learning algorithms. As the intelligence and scope of our cyber defenses expands, our defensive systems will be more like partners and less like computer programs. As with all partnerships, the analyst must be able to communicate with and trust the defense system. If an analyst can't understand why decisions are made by the system, the system will simply be turned off.

Cy-QLAP is designed so that its data representations correspond to human intuition. Instead of creating one large and cumbersome model of the world, Cy-QLAP breaks its representation of the cyber system up into many small and easily understandable causal models. Each model is based on a contingency, such as if I close port 25, then email will not function. Humans think naturally in terms of contingencies, and it is even posited that we humans have an innate contingency detection module.[16]

Cy-QLAP begins its learning process with a set of primitive actions, and by using these actions, Cy-QLAP expands its knowledge by learning the effects of those actions. Piaget[3] described how children construct knowl-

www.manaraa.com

edge in stages and learned new concepts on top of those they already know. Focusing on what the learner needs to learn based on its current knowledge helps to narrow the learning problem. This method of learning can be extended with instructors, as discussed by the psychologist Lev Vygotsky.[17] Vygotsky proposed the concept of the *zone of proximal development*, which leads to the conclusion that a teacher can maximize a child's capabilities by helping the child with tasks at the edge of the child's current knowledge.

To employ Cy-QLAP, an analyst can specify a set of critical assets that should be protected. An example would be that all files of a particular type should be protected from exfiltration. The analyst can direct Cy-QLAP's learning by specifying an initial set of primitive actions and important events. If need be, the analyst can also intervene to help Cy-QLAP stay within its zone of proximal development.

In this paper, we present experimental results in a simple environment that demonstrate how Cy-QLAP can use knowledge learned through exploration to take actions to proactively thwart an attack. Cy-QLAP explored an environment consisting of an end node that it was protecting and a virtual machine that Cy-QLAP used to probe the end node. Through its exploration, Cy-QLAP learned the dynamics of the environment. Specifically, it learned that a sensitive file could be exfiltrated when a file share was opened. Cy-QLAP was then asked to use its acquired knowledge to protect the system. Cy-QLAP monitored the system, and when it found that a file share was open, it used its learned knowledge of the dynamics of the system to close it, thus keeping the files from being exfiltrated. Cy-QLAP did not have to be told what the dangers were or how to prevent them; Cy-QLAP learned both how the system could be compromised and how to defend the system. Because Cy-QLAP does not have to be given the dynamics of the system, Cy-QLAP is a domain general defensive system.

2. PROTECTING CYBER SYSTEMS THROUGH LEARNED CAUSAL MODELS

2.1 Learning Causal Models

Cy-QLAP autonomously learns causal models to predict and control a system. Each causal model has a learned contingency at its core. A contingency is a pair of events that occur together in time such that an antecedent event is followed by a consequent event. An example would be that flipping a light switch (the antecedent event) is soon followed by the light going on (the consequent event). An example that Cy-QLAP could learn on a computer system is that shutting down port 25 causes SMTP email to stop working correctly. Contingencies are a particularly useful method for learning models. They are easy to learn because they only require looking at pairs of events, and they are a natural representation for planning because they indicate how events lead to other events.

Cy-QLAP learns a contingency between an antecedent event e_1 and a consequent event e_2. It does this using the *soon* window, which is a window of time. For the light switch example, flipping a light switch on (the antecedent event) causes a light to go on (the consequent event) in a small amount of time (the soon window). Mathematically, Cy-QLAP learns a contingency if

$$Pr(soon(e_2)|e_1) > Pr(soon(e_2)) + penalty \tag{1}$$

Contingencies form the basis for Dynamic Bayesian Networks (DBNs). DBNs are the mathematical representation of causal models used in Cy-QLAP because they represent changes over time. DBNs are a type of graphical model. Graphical models are used to represent multi-dimensional probability distributions, where variables are nodes and edges capture the conditional dependence between variables.

Due to the large number of variables that Cy-QLAP has to monitor on a computer system, estimating the joint probability distribution of all variables directly is an intractable task. However, since each variable is typically only dependent on a subset of the other variables, a DBN can allow for the compact representation of a probability distribution. For example, if the probability of a value of variable A conditioned on some other variables B and C is independent of the value of a variable D, then $P(A|B,C) = P(A|B,C,D)$. This means that D can be dropped, and Cy-QLAP can concisely represent some small piece of the world.

Figure 1 shows an example DBN. We see that the antecedent event A (e.g., the light switch is flipped) and the consequent event B (e.g., the light goes on) form the core of the DBN. Once the core of the DBN is created by a contingency, Cy-QLAP identifies the important variables $V_1, \ldots, V_n$, called context variables, from the set

www.manaraa.com

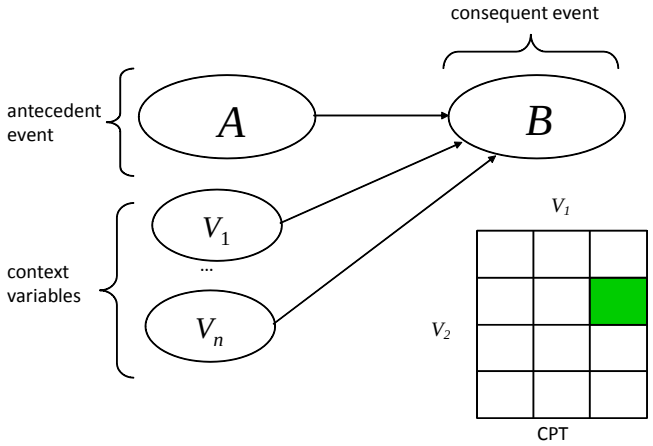

Figure 1. Dynamic Bayesian Network and Conditional Probability Table

of all variables. The conditional probability table (CPT) gives the probability of the antecedent event A bringing about the consequent event B for each possible value of the context variables $V_1, \ldots, V_n$.

As Cy-QLAP observes and actively explores the system, it learns the context variables for each DBN model through a process called marginal attribution.[18] Marginal attribution works by iteratively adding context variables as long as each new context variable makes the DBN marginally more deterministic. The level of determinism is measured by the highest probability of any value in the CPT, as long as that value is less than 0.75. After that value, the level of determinism is measured by the entropy of the entire CPT. This duel method of measuring determinism is used because we initially want to find some situation in which the contingency is reliably achieved, and once that is done, we seek a representation of the environment that is predictable in all situations.

2.2 Protecting the System Using Learned Causal Models

The result of the learning process is a set of DBNs, where each DBN contains a CPT that gives the probability of the antecedent event leading to the consequent event for each relevant state of the world as described by the context variables. Cy-QLAP learns many small DBNs to model the system, and these DBNs can be chained together to form plans. These plans allow Cy-QLAP to actively protect the system using the Threat Monitoring Module and the Threat Intervention Module. After the human analyst has specified a set of undesirable events that should be avoided, the Threat Monitoring Module can observe the system to see if it is possible to formulate a plan to bring about an undesirable event. If it finds such a plan, the Threat Intervention Module takes an action to break a link in that plan. To specify exactly how this is done, we first provide some definitions.

- A DBN *succeeds* if its antecedent event leads to the consequent event in the environment.
- A DBN is *sufficiently reliable* if the probability of success is greater than 0.75 in some state.
- A DBN is *satisfied* if the probability of success of the DBN is greater than 0.75 in the current state.
- If a DBN has a context, and the context is satisfied, the *satisfied context value* is the variable and value that satisfies it. For example, if a DBN is satisfied when variable $v = True$, and $v = True$ in the current state, then the satisfied context value is $v = True$.
- A goal is *achieved* if it is made to be true. For example, if the goal is $u = False$, and $u = False$ in the current state, then that goal is achieved.

The Threat Monitoring Module watches the system to see if it is possible to bring about an undesirable event. It does this by continually calling Subgoal Planner (Algorithm 1) and passing it each undesirable event as a goal.

www.manaraa.com

If the subgoal planner returns an action, that means that the undesirable event can be achieved. In this case, the subgoal planner also returns the satisfied context value that makes this possible. Cy-QLAP is then able to make sure this plan cannot be achieved by an adversary by calling the Threat Intervention Module.

Algorithm 1 Subgoal Planner

Require: *goal* {returns *subgoal* or (*action* and satisfied context value) or FAIL}

```
1: d = SelectDBNForGoal(state, goal)
2: if no DBN found then
3:    return FAIL
4: end if
5: if context of DBN d is satisfied then
6:    if antecedent of d is a defined action then
7:       return antecedent event of d and satisfied context value
8:    else
9:       let goal = antecedent event d
10:      goto step 1
11:   end if
12: else
13:   subgoal = SelectContextSubgoal(d, state)
14:   return subgoal
15: end if
```

Algorithm 1 uses two functions functions. *SelectDBNForGoal*: find the sufficiently reliable DBN that is most reliable in the current state, and *SelectContextSubgoal*: find the context value where the DBN is most reliable and return it as a subgoal.

The Threat Intervention Module changes the environment so that the threat found by the monitoring module is negated and cannot be exploited by an adversary. It does this by unsetting the satisfied context value that makes possible the plan to bring about the undesirable event. Specifically, it calls Achieve Event (Algorithm 2, shown below) with the negation of the satisfied context value as the goal.

Algorithm 2 Achieve Event

Require: *goal* {returns FAIL or SUCCESS}

```
1: push goal onto stack
2: while stack is not empty do
3:    retval = Subgoal Planner(top-of-stack)
4:    if retval is FAIL then
5:       return FAIL
6:    else if retval is a subgoal then
7:       push subgoal onto stack
8:    else
9:       perform action
10:      if top-of-stack now achieved then
11:         pop stack
12:         if stack is empty then
13:            return SUCCESS
14:         end if
15:      else
16:         return FAIL
17:      end if
18:   end if
19: end while
```

www.manaraa.com

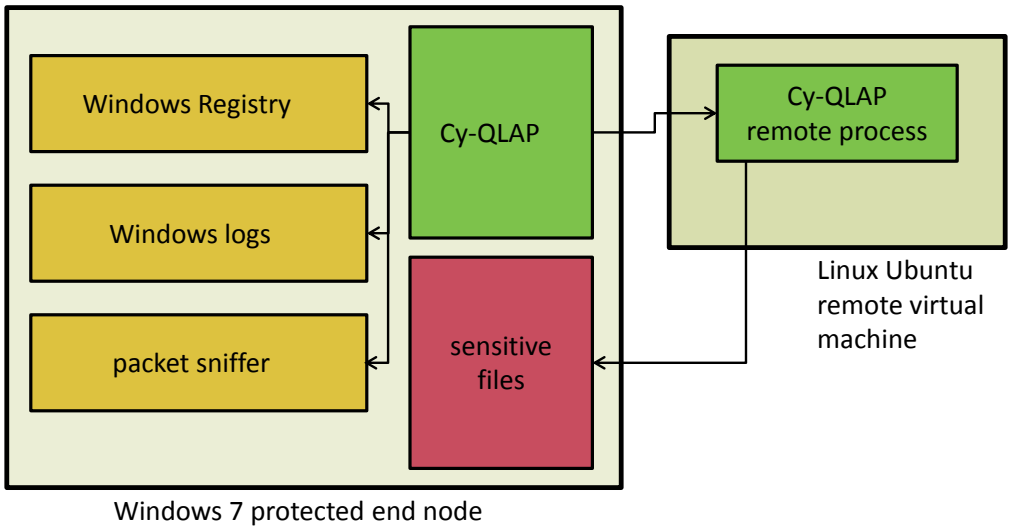

Figure 2. The Experiment Environment

3. EXPERIMENTAL EVALUATION

Cy-QLAP is designed to use knowledge learned through exploration to take actions to proactively thwart attacks. We performed the experiment described here to evaluate this ability.

3.1 Experiment Environment

We evaluated Cy-QLAP in the simple environment shown in Figure 2. The environment consists of a Windows 7 end node that we wish to protect and a Linux Ubuntu virtual machine that is used to learn the effects of external actions. Cy-QLAP runs on the end node and has a remote process on the virtual machine. Cy-QLAP takes exploratory actions both locally on the end node and remotely from the virtual machine. These actions relate to a set of sensitive files that we wish to protect on the end node. To learn about the effects of both its internal and external actions, Cy-QLAP reads the Windows Registry and the Windows logs, and it runs a packet sniffer.

State variables determine the state space of the system. Those variables are:

- $file\_open(x)$: Bool is $True$ when file x is open. Cy-QLAP determines the value of this variable by looking at the system log.
- $registry\_created(x)$: Bool is $True$ if a registry entry exists indicating a file share exists on x. Cy-QLAP determines this by looking at the Windows Registry.
- $exfiltration\_obs(x)$: Bool is $True$ if exfiltration of file x is observed. Cy-QLAP determines this by sniffing packets.

Actions are state variables that the agent can set directly. The actions are:

- $open\_file(x)$: Open file named x on the protected end node from the remote virtual machine using the Samba file sharing protocol.
- $create\_share(x)$: Open a file share on file x on the protected end node.
- $destroy\_share(x)$: Close a file share on file x on the protected end node.
- $copy\_file\_remote(x)$: Copy file x off the protected end node to the remote virtual machine using the Samba file sharing protocol.

www.manaraa.com

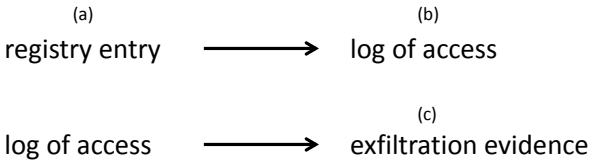

Figure 3. Exfiltration Scenario Represented in Cy-QLAP

The environment contains two files. At each timestep, Cy-QLAP will sense the environment and take a random action. At time t, an action will have value $True$ if it was taken and value $False$ otherwise. Each timestep will result in a state-action vector of the form:
$[file\_open\_1, file\_open\_2, registry\_created\_1, registry\_created\_2, exfiltration\_obs\_1, exfiltration\_obs\_2,$
$open\_file\_1, open\_file\_2, close\_file\_1, close\_file\_2, create\_share\_1, create\_share\_2, destroy\_share\_1,$
$destroy\_share\_2, copy\_file\_remote\_1, copy\_file\_remote\_2]$.

3.2 Evaluation Scenario

Figure 3 shows a representation of the file exfiltration scenario. The opening up of a Windows file share creates a registry entry (Figure 3(a)). Cy-QLAP can then notice that this allows the file to be accessed by looking at the Windows log (Figure 3(b)). Cy-QLAP can then notice that this log indicates that exfiltration evidence can be found through packet capture (Figure 3(c)).

3.2.1 Figure 3(a): New Registry Entry

System configuration information is stored in the Windows Registry. The registry consists of a set of keys and corresponding values. The values themselves can be sets of keys and values, so the registry is hierarchical, like a file system. There are hundreds of thousands of registry keys and values. Figure 3(a) corresponds to the registry entry created when a file share is opened. We created a sensitive document in `C:\secret_folder`. By looking at the registry using the Regedit utility, we see that there is a share on the folder for the key `HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\services\LanmanServer\Shares`.

3.2.2 Figure 3(b): New Log Record

Windows creates log entries when a user logs in, installs an application, or connects the computer to a wireless network, among many other events. The Windows log can be viewed by going to the run or search box and typing eventvwr.msc. Some logs, such as logs on file sharing, need to be turned on explicitly. We turned on the file sharing audit log. Figure 10(b) corresponds to the log entry created when a file share is opened.

3.2.3 Figure 3(b): Evidence of File Exfiltration

One approach to identify the exfiltration is to use a packet sniffer and to then search for the file within the packets. Cy QLAP can perform packet inspection to monitor what enters and leaves the machine. Cy-QLAP can look for specific pieces of information and create events of the form $\langle$found information, time$\rangle$.

3.3 Experimental Procedure

The experimental procedure consisted of two phases. During the first phase, Cy-QLAP explored the environment to learn a set of causal models that could be used to represent the dynamics of the environment. During the second phase, Cy-QLAP used those learned causal models to monitor and actively protect the system.

To learn contingencies, Cy-QLAP explored the environment by taking 1,000 actions and observing their effects. This process was not optimized for speed and lasted about four hours. Most of the time was spent waiting for the packets to come across or for the system log to be updated. The processing time needed for Cy-QLAP learning was negligible. To learn DBNs, Cy-QLAP began with the learned contingencies and took an additional 1,000 actions and observed their effects. This process also lasted about four hours.

www.manaraa.com

3.4 Results of Learning Causal Models

Cy-QLAP was able to learn the dynamics of the environment. DBNs learned by Cy-QLAP are shown in `typewriter` font. The notation `X-->x` means that the event of variable X going to value x occurred. In this case, 1 is $True$ and 0 is $False$. The notation `A=>B` means that the contingency consists of the antecedent event A leading to the consequent event B. The top line following the contingency contains: `succ`, which means the number of times the contingency was successful; `fail`, the number of times the contingency failed; and `rel`, the reliability of the contingency. The second line contains the conditional probability table of the context, if one was learned. The first value is the variable in the context. The final value is the conditional probability table for the contingency given each value of the context variable `[False, True]`.

Cy-QLAP learned that creating a file share on a file results in a new registry entry indicating the share.

```
create_share_1-->1 => registry_created_1-->1 succ 52 fail 0 rel 1.00
create_share_2-->1 => registry_created_2-->1 succ 51 fail 0 rel 1.00
```

By trying to open files from the remote process, Cy-QLAP learned that once there is that registry entry, it can open a file.

```
open_file_1-->1 => file_open_1-->1 succ 38 fail 47 rel 0.45
     registry_created_1, [0.00 1.00 ]
open_file_2-->1 => file_open_2-->1 succ 35 fail 39 rel 0.47
    registry_created_2, [0.00 1.00 ]
```

Cy-QLAP learned that if a file share is created, it can send it off the computer.

```
copy_file_remote_1-->1 => exfiltration_obs_1-->1 succ 28 fail 55 rel 0.34
     registry_created_1, [0.00 0.88 ]
copy_file_remote_2-->1 => exfiltration_obs_2-->1 succ 41 fail 49 rel 0.46
    registry_created_2, [0.00 0.95 ]
```

Cy-QLAP also learned that if it closes a file share, it can change the registry entry so that the file can no longer be opened.

```
destroy_share_1-->1 => registry_created_1-->0 succ 54 fail 0 rel 1.00
destroy_share_2-->1 => registry_created_2-->0 succ 60 fail 0 rel 1.00
```

We see from the conditional probability table of the following two DBNs that Cy-QLAP learned that if the registry value is not there, then the file will not be able to be opened.

```
open_file_1-->1 => file_open_1-->1 succ 38 fail 47 rel 0.45
    registry_created_1, [0.00 1.00 ]
open_file_2-->1 => file_open_2-->1 succ 35 fail 39 rel 0.47
    registry_created_2, [0.00 1.00 ]
```

3.5 Results of Protecting the System Using the Learned Causal Models

We specify that the system should prevent file exfiltration by preventing $file\_exfiltration\_1 = True$ or $file\_exfiltration\_2 = True$. When passing the Threat Monitoring Module various states where file exfiltration was not possible (file sharing was turned on), the monitoring module returned that all was ok. However, when we passed it the state of the world where $registry\_created\_1 = True$. It indicated that it was possible to exfiltrate the file, and it stated that the world needed to be changed so that $registry\_created\_1 = False$.

The goal of $registry\_created\_1 = False$ was passed to the Threat Intervention Module. The module returned that the required action was *destroy_share*. This process worked in the same way when a file share was set up for file 2 and it was in danger.

www.manaraa.com

What is noteworthy about this scenario is that the human did not have to specify what the threat was or how to prevent it. The human only needed to specify what should be prevented, and Cy-QLAP figured out how the undesirable event could be brought about and how to take action to prevent it.

4. DISCUSSION

Cy-QLAP autonomously learned to predict and defend the system. Cy-QLAP is a domain-general protection system as shown in the Generalized Cy-QLAP Algorithm.

Algorithm 3 Generalized Cy-QLAP Algorithm

1: human SME specifies a set of states and actions
2: human SME specifies a set of undesirable events that should be avoided
3: Cy-QLAP actively explores to learn the dynamics of the environment
4: **loop**
5: Cy-QLAP monitors to see if it is possible to formulate a plan to bring about an undesirable event
6: If such a plan is found, Cy-QLAP takes an action to break a link in that plan
7: **end loop**

Using this generalized algorithm, Cy-QLAP is able to work to defend an arbitrary system. Cy-QLAP represents a step toward increased automation and intelligence of defense systems. This increased intelligence and autonomy raises at least two issues. The first issue is that of control. The flip side of an autonomous system is that the human operators have less control over its actions. Cy-QLAP actively defends a system by taking protective actions. Cy-QLAP chooses which actions are most important in a given situation, and this creates the possibility that Cy-QLAP could choose an action that would be undesirable from the perspective of the human operator. This problem is inherent in any system with autonomy, but, as described in the introduction in Section 1, the inner workings of Cy-QLAP are human understandable. This enables the analyst and the algorithm to work as a team, and can allow our defenses to have the best of both machine intelligence and human intelligence.

A second issue that stems from intelligence and autonomy is the level of abstraction at which the algorithm reasons about the world. Cy-QLAP learns a causal model to describe a system, and it can use that causal model to simulate an adversary to thwart and attack, but Cy-QLAP must be given the set of possible events over which it should reason. Learning abstractions suitable for reasoning and further learning is our future research direction.

5. CONCLUSION

We know of no other cyber defense system that learns to actively protect through autonomous exploration. As such, Cy-QLAP represents an extension both in intelligence and autonomy. Instead of using supplied rules or looking for anomalies, Cy-QLAP learns a general model of the system. This model allows Cy-QLAP to predict what can happen and prevent it so that in a sense it learns an attack tree.

When employing Cy-QLAP, the human operator does not have to specify how the system should be protected; the operator need only specify what should be protected. This increased autonomy coupled with the human understandability of Cy-QLAP allows Cy-QLAP and the human operator to work as partners. Any autonomous tool that is inscrutable to the analyst risks being turned off, but when Cy-QLAP takes an action, it can explain why it made that decision, which engenders trust in the algorithm. This cooperative relationship between the algorithm and the human expert can allow our defense systems to benefit from the best of both human and machine intelligence.

ACKNOWLEDGMENTS

The author would like to thank the Air Force Research Laboratory (AFRL) for their generous support. In particular, the author would like to thank David Kapp for his guidance and support. The author also wishes to thank Fred Chang, Thayne Coffman, Laura Hitt, and Matt McClain for their insights and suggestions.

www.manaraa.com

REFERENCES

[1] Mandler, J., [*The Foundations of Mind, Origins of Conceptual Thought*], Oxford University Press, New York, New York, USA (2004).

[2] Gopnik, A., [*The Philosophical Baby: What Children's Minds Tell Us about Truth, Love, and the Meaning of Life*], Farrar Straus & Giroux (2009).

[3] Piaget, J., [*The origins of intelligence in children*], Norton, New York (1952).

[4] Simonite, T., "The antivirus era is over." http://www.technologyreview.com/news/428166/the-antivirus-era-is-over/ (June 2012). Accessed on June 13, 2012.

[5] Morel, B., [*Intrusion detection systems*], ch. Anomaly based intrusion detection and artificial intelligence, no. 2, InTech (2011).

[6] Koller, R., Rangaswami, R., Marrero, J., Hernandez, I., Smith, G., Barsilai, M., Necula, S., Sadjadi, S., Li, T., and Merrill, K., "Anatomy of a real-time intrusion prevention system," in [*International conference on autonomic computing (ICAC'08)*], 151–160, IEEE (2008).

[7] Chan, P., Mahoney, M., and Arshad, M., "A machine learning approach to anomaly detection," tech. rep., Department of Computer Sciences, Florida Institute of Technology (2003).

[8] Pearl, J., [*Causality: models, reasoning, and inference*], Cambridge University Press (2000).

[9] Mugan, J. and McClain, M., "A method of applying developmental learning to cyber security," (2012). Provisional patent application number 61/720,969.

[10] Mugan, J., *Autonomous Qualitative Learning of Distinctions and Actions in a Developing Agent*, PhD thesis, University of Texas at Austin (2010).

[11] Mugan, J. and Kuipers, B., "Autonomous learning of high-level states and actions in continuous environments," *IEEE Trans. Autonomous Mental Development* **4**(1), 70–86 (2012).

[12] Siddiqui, M., Wang, M., and Lee, J., "Detecting internet worms using data mining techniques," *Journal of Systemics, Cybernetics and Informatics* **6**(6), 48–53 (2009).

[13] Apap, F., Honig, A., Hershkop, S., Eskin, E., and Stolfo, S., "Detecting malicious software by monitoring anomalous windows registry accesses," in [*Proceedings of the 5th International Conference on Recent Advances in Intrusion Detection*], 36–53, Springer-Verlag (2002).

[14] Treinen, J. and Thurimella, R., "A framework for the application of association rule mining in large intrusion detection infrastructures," in [*Recent Advances in Intrusion Detection*], 1–18, Springer (2006).

[15] Zurutuza, U., Uribeetxeberria, R., Azketa, E., Gil, G., Lizarraga, J., and Fernández, M., "Combined data mining approach for intrusion detection," in [*International Conference on Security and Cryptography, Barcelona, Spain*], (2008).

[16] Gergely, G. and Watson, J., [*Early social cognition: Understanding others in the first months of life*], ch. Early socio-emotional development: Contingency perception and the social-biofeedback model, 101136, Lawrence Erlbaum (1999).

[17] Vygotsky, L., [*Mind in society: The development of higher psychological processes*], Harvard University Press (1978).

[18] Drescher, G. L., [*Made-Up Minds: A Constructivist Approach to Artificial Intelligence*] (1991).

المنارة للاستشارات

www.manaraa.com