Modelling Cyber Defenses using s(CASP)

Steve Moyle[1], Nicholas Allott[2] and John Manslow[2]

[1]Amplify Intelligence U.K., Oxford, United Kingdom
[2]Nquiring Minds, Southampton, United Kingdom

Abstract

Cyber attacking is easier than cyber defending – attackers only need to find one breach, while the defenders must successfully repel all attacks. This preliminary work uses s(CASP) as a framework for modelling networks of devices, and their associated insecurities and defenses. Cyber defenders often need to reason with missing, uncertain, and contradictory evidence. This position paper demonstrates that cyber defenders can amplify their capabilities by joining forces with eXplainable-AI (XAI) utilising interactive human machine collaboration mediated through s(CASP).

Keywords

Explainable Artificial Intelligence (XAI), Cyber defense, reasoning

1. Introduction

Defending assets from cyber attack remains challenging. There are many asymmetries that disadvantage the defender, including – rare symptoms of compromise are buried within burgeoning log files of (most likely) normal behaviour; – specification of the assets are rarely complete and accurate; – public knowledge of exploits are always out of date, with the continual emergence of new zero-day attacks; – systems may have a vulnerable component, but the deployment context and configuration may make it unexploitable. This leaves the defender having to reason in the complex world where the information they have access to cannot be relied upon. Ultimately, skilled cyber defenders admit that it is almost impossible for them to give any security guarantees [13].

Cyber defenders apply their skill and reasoning in much the same way that a detective solves a murder case. They constantly form hypotheses, collect more evidence, and focus their investigations. They will use *abductive* reasoning to form hypotheses and seek yet-to-be discovered evidence, whilst forming constraints based on the evidence that will rule out incompatible hypotheses and lines of enquiry. They may be able to conclude, based on reasonable probability, the motive, opportunity, and weapon used by a suspect. Once the case is solved, then it is possible to follow a *deductive* reasoning process that explains the case and its artefacts[1].

Logic programming has a long history of knowledge representation and reasoning. Despite its longevity and many successes standard Prolog, with its sound semantics and elegant

3rd Workshop on Goal-directed Execution of Answer Set Programs (GDE'23), July 10, 2023

steve@amplifyintelligence.co.uk (S. Moyle); nick@nquiringminds.com (N. Allott); john@nquiringminds.com (J. Manslow)

[1] Furthermore, with the solution of similar styles of cases it might be possible to apply *inductive* reasoning to learn general patterns.(e.g. [19]).

computation model (e.g [10]), does not directly support the forms of reasoning that humans often rely on. The more expressive[2] Answer Set Programming (e.g. [9]), allows a broader range of reasoning, and is not restricted to Logic Programs with single models. In this investigation we utilize the `s(CASP)` [11] [3] approach that combines the strengths of the previously mentioned approaches[3].

`s(CASP)` has been used in various applications. Varanasi et al. [23] use `s(CASP)` to model the safety-critical behaviour of a timed system. Hall et al. [8] have used `s(CASP)` to model systems that have been specified using the EARS specification framework [12]. Unlike those domains, the cyber security world contains inconsistent, incomplete, and missing specifications. Worse, where specifications exist, the implementations may not be faithful.

2. Background to the Cyber Domain

Real computer networks can be arbitrarily complex. For example, detailed analysis of a small organisation's network, where data was collected for half a year, showed that 181 devices were connected within its networks[16]. Some of the devices are for normal operational duties (for example laptops, network routers, door swipe access controllers), others are defensive in nature (for example virus and vulnerability scanners). All devices pose a risk of expanding the organisation's *attack surface*.

Cyber defenders are keen to incorporate both *controls* (for example network firewalls) and *monitoring* as part of their defensive toolkit. Monitoring can be *passive* like the continuously running network security monitoring system `Zeek` [24], or it can be *active*, like the vulnerability scanner `OpenVAS` [17] where probes for known vulnerabilities are commissioned.

In this initial work we start by modelling a simple network of devices as shown in the block diagram in Figure 1 . There are *networks* and there are *devices*. A *router* device can connect more than one network. In the simple network the router `openwrt01` connects the networks `192.168.15.` and `192.168.116.`; while the IoT door control device `frontdoorring01` and the vulnerability scanner `openvas01` are connected to the `192.168.116.` network.

Designing a secure device is fundamentally challenging. Certifying the level of security for a particular device is also challenging and costly (e.g. [5] [6]). A certified secure device at installation time allows us to *presume* that it is *good* at that time, but some time later a *vulnerability* in that version of the device is discovered. This does not automatically mean the device is *bad* – the vulnerability needs to be *exploitable*, and that a path-way to exploitation is present.

Scanning systems for probing devices for known vulnerabilities are sometimes used to catalogue the status of devices. There is clearly a time-lag between the discovery of exploits and the knowledge being operationably available to a vulnerability scanner (e.g. Zero-day attacks are by definition unknown to the broader defense community). Vulnerability scanning is not perfect, and like all decision systems, they make errors. A scan of a device may return *false*

[2] But introducing limitations of its own.

[3] `s(CASP)` also introduces its own limitations.

negatives[4] and *false positives*[5]

Installers of devices rarely know precisely how the device should behave. There is a growing appreciation for published device information from the device manufacturers themselves helping to improve security (e.g. [7], [14]). The *Distributed Device Descriptors (D3)* standard[1] allows both manufacturers and interested parties to specify what resources a device class requires, and the behaviour that it produces. This allows a run-time comparison of the behaviour of an instance of the device class and its actual, monitored behaviour. In the model of the network we introduce the notion of a device instance and its behaviour profile as published by some other party (e.g. a manufacturer) and the ability to observe the device instance and compare its behaviour to that specified in a profile. Deviations from specification can be caused by the device going *bad*, or simply that the specification has errors of omission or commission.

3. Modelling the Cyber Domain

In this section we provide a flavour of the modelling and reasoning it supports for the very simple network of devices shown in Figure 1.

The code listing in Figure 2 provides illustrative fragments of the `s(CASP)` program that encodes the network and the security characteristics of the devices. In line 1 a device is identified, and line 9 a scanner is defined, thus introducing some of the basic objects of the domain. Line 12 through 16 encode how scanners identify that devices are `good` or `bad`.

```
 1  device(frontdoorring01).
 2
 3  bad(Device) :- device(Device), not
        good(Device).
 4  good(Device) :- device(Device), not
        bad(Device).
 5
 6  scan_outcome(bad).
 7  scan_outcome(good).
 8
 9  scanner(openvas01).
10  device(Scanner) :- scanner(Scanner).
11
12  scan(Device, Scanner, bad) :-
13      device(Device), scanner(Scanner),
14      not scan(Device, Scanner, good).
15  scan(Device, Scanner, good) :- device(
        Device), scanner(Scanner),
16      not scan(Device, Scanner, bad).
```

Figure 2: Encoding of the simple network of devices. Some illustrative code fragments of the network, the devices, and the logic to support exploratory reasoning.

3.1. Interrogating the Model

With a plausible initial encoding of the domain we can interrogate the model by posing it queries. The code listing in Figure 3 provides illustrative security queries we can pose to the `s(CASP)`[6] program that encodes the network and the devices. Note that each query starts with `?` indicating to the `SWI-Prolog` top-level that we are posing the query to the `s(CASP)` system and not just the `Prolog`.

[4] The scanner gives the all clear when the device actually has a vulnerability.

[5] The scanner reports the device has a vulnerability when it actually does not.

[6] Software versions: `SWI-Prolog (threaded, 64 bits, version 9.1.10)`; and `scasp@0.10.0`.

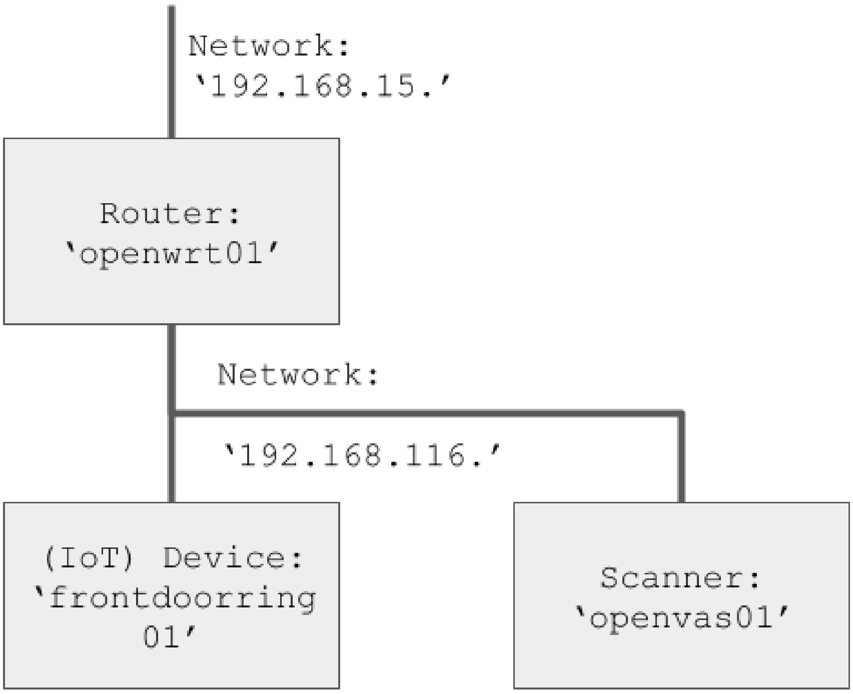

Figure 1: Simple Network Context Diagram. Two networks connected by a router device, with one of the networks containing an IoT door security lock system, as well as a network vulnerability scanner.

```
? router_good(Router).
? scan(openwrt01, Scanner, Scan).
? device_not_functional(Device).
? device_not_functional(
    frontdoorring01).
? router_not_functional(Router).
? network(N).
? network_not_secure(Network).
? mismatched_profile(Device, Profile).
? incorrect_type(Device).
? good(Device).
? bad(Device).
? scan(Device, Scanner, false_positive
    ).
? -scan(Device, Scanner,
    false_positive).
? -functional(Router).
? secure(Network).
? secure('192.168.116.').
? secure('192.168.15.').
? insecure(Network).
```

Figure 3: sCASP queries relating to cyber security. These are illustrative queries that can be posed to the sCASP system of the encoded simple network and its security.

Cyber defenders are interested to determine existing security weaknesses in the assets they are responsible for. For example they are constantly wondering "*Are there any bad devices on my network?*". They are curious to know "*Is the network secure?*", or better still "*What makes the network insecure?*". A sample list of `s(CASP)` queries that cyber defenders may be interested in is shown in Figure 3.

The answers being simply *yes* or *no* does not allow the defender to identify specific actions they can take. However, interacting with the `s(CASP)` model provides answers supported with the elements that must hold. These elements allow the defender to determine where (and, sometimes, how) to remediate so as to improve security. For example, the response that a particular network is insecure, the `s(CASP)` accompanying explanatory model identifies that for the network to be insecure, it can be caused by a particular device being *bad.* This allows the defender to actively improve the security of the device, so that the overall network is more secure. Figure 4 shows the `s(CASP)` query and corresponding `s(CASP)` model to determine which device is bad, and the supporting reasoning that makes it so.

```
3 ?- ? bad(Device).
Device = frontdoorring01,
% s(CASP) model
{ bad(frontdoorring01),
   :
  device(frontdoorring01),
  device(openvas01),
  device(openwrt01),
  not good(frontdoorring01),
   :
  router(openwrt01),
   :
  scanner(openvas01),
   :
  scan(frontdoorring01,openvas01,bad),
  -scan(frontdoorring01,openvas01,
      false_negative),
  scan(frontdoorring01,openvas01,
      false_positive),
  not scan(frontdoorring01,openvas01,
      good),
   :
  not scan(openvas01,openvas01,bad),
  scan(openvas01,openvas01,
      false_negative),
  scan(openvas01,openvas01,good),
   :
  not scan(openwrt01,openvas01,bad),
  scan(openwrt01,openvas01,
      false_negative),
  -scan(openwrt01,openvas01,
      false_positive),
  scan(openwrt01,openvas01,good)
}, ...

4 ?-
```

Figure 4: sCASP output for the query `?- ? bad(Device).` This is a sample of the results generated by the `SWI-Prolog` `s(CASP)` library. The query asks whether there could be a device on the network, and in what way would a device be determined as *bad.*

4. Discussion

The results presented are preliminary, but they do demonstrate the feasibility of the approach. Using Logic Programming syntax to describe complicated networks and security properties has a relatively low impedance to human cyber defenders. Human-computer comprehensibility has been a long standing requirement for those interested in strong AI (e.g. [15]), and is crucial for eXplanatory AI environments. Ultimately, we aim for systems that interact in a strong way with deep human experts so that both the humans and the machines can critique and improve each other's understanding (see [22]). The ability to identify actionable remedies is a particular strength of the approach offered by s(CASP).

There are many directions for expanding this work. The incorporation of pre-existing cyber security domain knowledge will make it possible to expand the encoding. Sikos [21] provides semantic web style ontologies for the cyber security domain.

The current simple encoding does not take into account the temporal nature of the domain. Much cyber information is recorded in log files with time stamped entries. The Logic Programming community has long been interested with temporal problems and reasoning about actions and change. One well studied formulation is the Event Calculus (e.g. see [20]). The Event Calculus has been used with ASP which requires the discretisation of the time dimension. Ray [18] provides a discrete-time version of the Event Calculus resolving many issues in previous formulations, but remains not fully compatible with real world scenarios requiring continuous time. Some authors ([23] and [4]) report that the Event Calculus is amenable to solution using s(CASP). However, the reported results only address specific use cases. In particular the work presented in [23] was not reproducible – although the paper showed a general purpose formulation of the Event Calculus, the actual results were achieved using a customized formulation for the specific case study only, which worked around known limitations in s(CASP).

The logical language of s(CASP) makes it easy for a cyber defender to interact and reason about security and take action. But cyber defenders are a scarce global resource[7]. Future work would consider putting the reasoning into a smart gateway [2] and have pro-active improvements being done by the gateway itself.

The domain of cyber security has been studied to demonstrate co-inductive Machine Learning and reasoning [19] where both human and machine work together to come to an understanding of a cyber incident. Although promising, it fails to achieve full two-way explainablility as defined by Srinivasan et.al. [22]. Future work may seek to combine approaches from both the goal-directed execution of Answer Set Programming (s(CASP)) and Inductive Logic Programming (ALEPH and ACUITY).

5. Conclusion

This preliminary work demonstrates the feasibility of the s(CASP) approach as an enabler of human-computer reasoning in the field of cyber security. Providing actionable information about the state of cyber defenses underlines the value of eXplanatory AI to the improvement of security in our networks of devices.

6. Acknowledgements

This work is supported by the Innovate-UK *105592: CyberStone: Collaborative Secure IOT Gateway (ManySecured)* grant.

[7] Some sources claim there is a 3,000,000 shortfall of cyber defenders globally.

References

[1] Nicholas Allott et al. 2021. Distributed Device Descriptors (D3), https://specs.manysecured.net/d3/ accessed 2022-10-19.

[2] Nicholas Allott et al. 2021. ManySecured Technical Documents – Smart Secure Collaborative Gateways, https://specs.manysecured.net/ accessed 2022-10-19.

[3] Joaquín Arias, Manuel Carro, Elmer Salazar, Kyle Marple, and Gopal Gupta. 2018. Constraint Answer Set Programming without Grounding. *Theory and Practice of Logic Programming* 18 (3-4). Cambridge University Press: 337–354.

[4] Joaquín Arias, Manuel Carro, Zhuo Chen, and Gopal Gupta. 2022. Modeling and Reasoning in Event Calculus using Goal-Directed Constraint Answer Set Programming. *Theory and Practice of Logic Programming*. 2022. Vol. 22, no. 1p. 51–80. DOI 10.1017/S1471068421000156.

[5] Common Criteria. The Common Criteria for Information Technology Security Evaluation. *International Standard (ISO/IEC 15408) for computer security certification.* Version 3.1 revision 5.[1]

[6] DoDD 5200.28-STD. 2002. United States Government Department of Defense Standard, DoDD 5200.28-STD. *Trusted Computer System Evaluation Criteria.*

[7] Parisa Grayeli and Blaine Mulugeta. 2020. Mitigating Network-Based Attacks Using MUD – Improving Security of Small-Business and Home IoT Devices. Practical Use of the MUD Specification, RSA Conference https://published-prd.lanyonevents.com/published/rsaus20/sessionsFiles/17591/2020_USA20_SBX1-W1_01_Practical-Use-of-the-MUD-Specification-to-Support-Access-Control-in-the-IoT.pdf accessed 2022-10-19.

[8] Brendan Hall, Kinjal Basu, Sarat Chandra Varanasi, Fang Li, Devesh Bhatt, Jan Fiedor, Kevin Driscoll, Joaquín Arias, Elmer Salazar, and Gopal Gupta. 2021. Knowledge-Assisted Reasoning of Model-Augmented System Requirements with Event Calculus and Goal-Directed Answer Set Programming, in Hossein Hojjat and Bishoksan Kafle (Eds.): *8th Workshop on Horn Clauses for Verification and Synthesis (HCVS 2021)* EPTCS 344, 2021, pp. 79–90, doi:10.4204/EPTCS.344.6

[9] Vladimir Lifschitz. What Is Answer Set Programming? https://www.cs.utexas.edu/users/vl/papers/wiasp.pdf accessed June 2023.

[10] John W. Lloyd. 1993. *Foundations of Logic Programming*. Springer-Verlag.

[11] Kyle Marple and Gopal Gupta. 2014. Dynamic Consistency Checking in Goal-Directed Answer Set Programming. *Theory and Practice of Logic Programming*, 14(4-5):415–427.

[12] Alistair Mavin, Philip Wilkinson, Adrian Harwood, and Mark Novak. 2009. EARS (Easy Approach to Requirements Syntax) in RE09, IEEE, August 2009 https://www.researchgate.net/publication/224079416_Easy_approach_to_requirements_syntax_EARS/link/568ce3bf08aeb488ea311990/download

[13] Haroon Meer. 2015. What Got Us Here Wont Get Us There. Key-note presentation Black-Hat Europe 2015. https://www.blackhat.com/eu-15/briefings.html#what-got-us-here-wont-get-us-there

[14] Michael Fagan, Katerina Megas, Paul Watrobski, Jeffery Marron, and Barbara Cuthill. 2022. Profile of the IoT Core Baseline for Consumer IoT Products, NIST Interagency/Internal Report (NISTIR), National Institute of Standards and Technology, Gaithersburg, MD, [online],

https://doi.org/10.6028/NIST.IR.8425, https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=935382 (Accessed February 28, 2023)

[15] Donald Michie. 1988. Machine Learning in the Next Five Years. In D. Sleeman and J. Richmond editors, *Third European Workshop Session on Learning (EWSL '88)*, p. 107 - 122. University of Strathclyde, Pitman, London.

[16] Steve Moyle. 2022. Simple Good-Turing Frequency Analysis applied to Network Monitoring for volatility indication. Internal Project Report – *105592: CyberStone: Collaborative Secure IOT Gateway (ManySecured)*.

[17] OpenVAS - Open Vulnerability Assessment Scanner. https://openvas.org/, June 2023.

[18] Oliver Ray. 2021. Learning and Revising Dynamic Temporal Theories in the Full Discrete Event Calculus. In the proceedings of the 30th International Conference on Inductive Logic Programming (ILP), LNAI 13191:219-233.

[19] Oliver Ray and Steve Moyle. 2021. Towards expert-guided elucidation of cyber attacks through interactive inductive logic programming *Proceedings of the 13th International Conference on Knowledge and Systems Engineering (KSE)*, IEEE Press.

[20] Murray Shanahan. 1997. *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*. MIT Press.

[21] Leslie F. Sikos, Markus Stumptner, Wolfgang Mayer, Catherine Howard, Shaun Voigt, and Dean Philp. 2018. Representing network knowledge using provenance-aware formalisms for cyber-situational awareness. In Procedia Computer Science, 126, 29-38. https://www.sciencedirect.com/science/article/pii/S1877050918311803.

[22] Ashwin Srinivasan, Michael Bain, and Enrico Coiera. 2022. One-way Explainability Isn't The Message. arXiv https://arxiv.org/abs/2205.08954, May 2022.

[23] Sarat Chandra Varanasi, Brendan Hall, Joaquín Arias, Elmer Salazar, Fang Li, Kinjal Basu, Kevin Driscoll, and Gopal Gupta. 2021. *Modelling and Verification of Timed Systems with the Event Calculus and s (CASP)*. ICLP Workshops. https://ceur-ws.org/Vol-2970/gdepaper2.pdf

[24] The Zeek Network Security Monitor. https://zeek.org/, June 2021.