SOCCRATES

SOC & CSIRT Response to Attacks & Threats based on attack defense graphs Evaluation Systems

D4.4 Tactical Threat Intelligence for Attack Defence Graphs

Deliverable type:	Report
Contributing work packages:	WP4 AI based Detection & Prediction
Due date of deliverable:	28/02/2022
Submission date:	28/02/2022
Dissemination level:	PU

Responsible organisation:	MNM
Editor:	Martin Eian
Revision:	1.0

Abstract	This deliverable describes the developed demonstrator to enable automatic generation of Adversary Emulation Plans (AEPs). Furthermore, we show how this is integrated in the SOCCRATES Threat Intelligence Platform (TIP), and how the AEPs can be used by the SOCCRATES Attack Defence Graph (ADG) Analyser.
Keywords:	Threat Intelligence, Adversary Emulation Plans, Attack Defence Graph

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 833481
Call H2020-SU-ICT-2018 • Innovation Action • Start date: September 1st, 2019

Management summary

This deliverable describes the work on tactical threat intelligence in the SOCCRATES project, where we developed methods to enable automatic generation of Adversary Emulation Plans (AEPs). We describe our approach to modelling adversary behaviour, our implementation of the models, and our results. Furthermore, we show how these results are integrated in the SOCCRATES Threat Intelligence Platform (TIP), and how the AEPs can be used by the SOCCRATES Attack Defence Graph (ADG) Analyser to simulate the most likely behaviour of a specific adversary.
The deliverable is a demonstrator, and the tools and documentation that have been developed are available under the ISC Open Source license[1] on Github[2][3].

[1] https://github.com/mnemonic-no/aep/blob/master/LICENSE
[2] https://github.com/mnemonic-no/aep
[3] https://github.com/mnemonic-no/aep-web

Contributor(s)	Martin Eian (mnemonic)

Reviewer(s)	Mathias Ekstedt (KTH)

Security Assessment	**See deliverables table for security assessment requirements**
Approval Date	16/02/2022
Remarks	I have found no issues with the document.

TABLE OF CONTENTS

1 Introduction – Rationale of this document

This section introduces the SOCCRATES project and defines the goals of this deliverable.

1.1 The SOCCRATES project

SOCCRATES (SOC & CSIRT Response to Attacks & Threats based on attack defence graphs Evaluation Systems) is an EU funded project under the Horizon2020 programme that has the following main challenge:

How can SOC and CSIRT operations effectively improve their capability in detecting and managing response to complex cyber-attacks and emerging threats, in complex and continuously evolving ICT infrastructures while there is a shortage of qualified cybersecurity talent?

The main objective of SOCCRATES is to develop and implement a security automation and decision support platform (‘**the SOCCRATES platform’)** that will significantly improve an organisation’s capability (usually implemented by a SOC and/or CSIRT) to quickly and effectively detect and respond to new cyber threats and ongoing attacks.

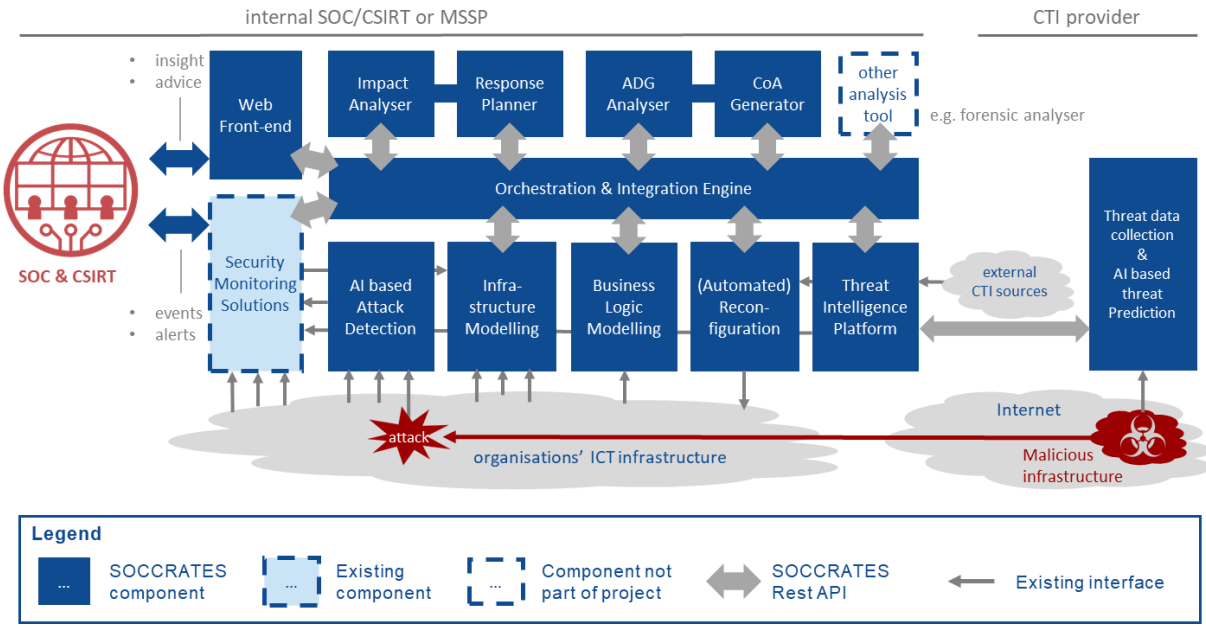

Figure 1.1 – The SOCCRATES platform

The SOCCRATES platform (see **Figure 1.1**) consists of an orchestrating function and a set of innovative components for automated infrastructure modelling, attack detection, cyber threat intelligence utilization, threat trend prediction, and automated analysis using attack defence graphs and business impact modelling to aid human analysis and decision making on response actions, and enable the execution of defensive actions at machine-speed.

SOCCRATES has the following concrete project objectives:

1. Deliver the SOCCRATES platform consisting of an orchestration function and a unique integration of innovative background solutions that seamlessly work together.
2. Show that the SOCCRATES platform can improve SOC operations by evaluating the SOCCRATES platform in two diverse real-life pilot environments.
3. Examine and illustrate the benefits of automation for selected SOC activities to help manage the cyber security skills gap in organizations.
4. Prepare for successful exploitation by the SOCCRATES partners of the individual innovated components and the integrated SOCCRATES platform in commercial products that are offered to the market and are available for the European (business) community.

Please visit www.soccrates.eu for more information on the SOCCRATES project.

1.2 This deliverable

This deliverable describes the work performed in Task 4.4: (Tactical) Thtreat Intelligence for Attack Defence Graphs. The deliverable is a demonstrator, and contains pointers to the tools and documentation that have been developed. The tools are available under the ISC Open Source license on Github.

1.3 Structure of this deliverable

Section 2 presents Adversary Emulation Plans (AEPs): the state of the art, standards, use cases, our problem statement and related work. Section 3 describes our approach to modelling technique dependencies, which forms the basis for the generation of AEPs. Section 4 describes our implementation of tools to generate AEPs, while Section 5 presents the results of our experiments. Section 6 provides suggestions for future work.

2 Adversary Emulation Plans

2.1 Adversary Emulation

Organisations defend their infrastructures against real adversaries. In order to realistically test their cyber defenses, they need to emulate how these adversaries behave: what they do during an intrusion and how the organisation is able to detect and respond to them. Adversary emulation can be automated, semi-automated or performed by human "red teams". In all cases, adversary emulation depends on knowledge about adversary behaviour: cyber threat intelligence (CTI). This CTI is collected from sources such as MITRE's Adversarial Tactics, Techniques and Common Knowledge (ATT&CK)[4].

In the SOCCRATES platform, the Attack Defence Graph (ADG) Analyser simulates adversary behaviour to guide an organisation's response to new CTI, infrastructure changes, discovered vulnerabilities, and intrusion attempts.

2.2 MITRE ATT&CK

MITRE ATT&CK is a de facto industry standard for describing adversary behaviour. ATT&CK is also a knowledge base that contains relationships betweeen techniques and adversary groups, software, mitigations, and tactics. ATT&CK uses the object-relationship model described in the ATT&CK Design and Philosophy paper[5], shown in Figure 2.

The ATT&CK object-relationship model does not describe *dependencies between techniques*. ATT&CK provides no guidance on whether or not a specific technique depends on the results from another technique or a set of techniques.

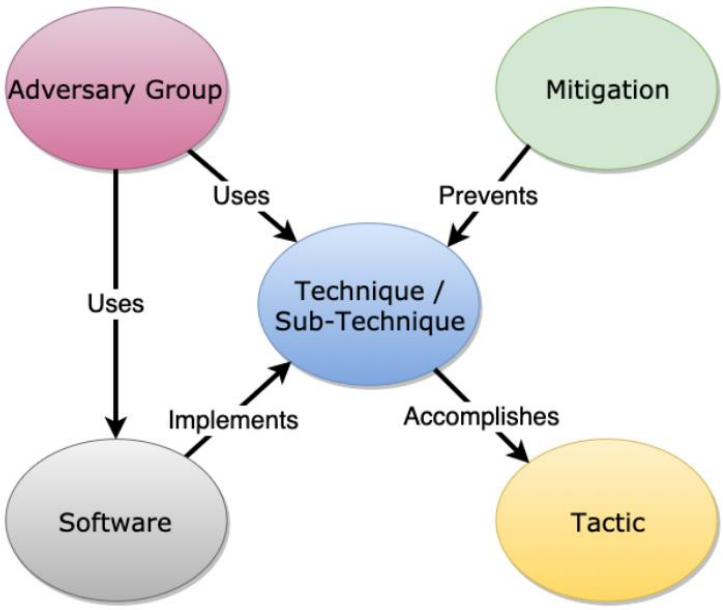

Figure 2: The ATT&CK Object-Relationship Model

[4] https://attack.mitre.org/

[5] https://attack.mitre.org/docs/ATTACK_Design_and_Philosophy_March_2020.pdf

2.3 Adversary Emulation Plans

Adversary emulation plans (AEPs) are used as a basis for adversary emulation, in order to ensure that the emulation is consistent and reproducible. AEPs are by the CTI community generally considered to be based on MITRE ATT&CK, and MITRE Engenuity's Centre for Threat-Informed Defense has published several examples of AEPs[6]. An AEP consists of three main parts:

1. An intelligence overview (human readable summary)
2. The operational flow
3. A command-by-command detailed plan of how to carry out an attack

The operational flow shows a high level description of the sequence of events during an intrusion, while the detailed plan shows exactly what the adversary does at each step. AEPs are currently created manually, and the creation of such a plan requires an excessive amount of time, expertise and effort.

2.4 Problem Statement

Large scale creation of AEPs is infeasible when done manually. We set out to automate this process, with the goal to provide input to the ADG Analyser so that it could model the most likely behaviour of a specific adversary. In order to achieve this, we needed more details than what is provided by the operational flow in current AEPs. We needed to be able to describe a set of attack stages with the specific techniques executed at each stage, in a machine readable format. The key to achieve this is to model *dependencies* between techniques. Techniques do not live in isolation. They require something in order to be used, and they provide something when used. If these dependencies can be described in a machine readable format, then that will make it possible to automatically generate such a set of attack stages.

Note that the AEP describes *possible* adversary behaviour, not one specific attack chain. It describes which techniques that could be executed at each stage of the attack, i.e. all the possible behaviours of a specific adversary given our current knowledge. Furthermore, the AEP does not consider the infrastructure under attack, it is purely a description of the adversary. Conversely, the ADG Analyser simulates specific attacks on a specific infrastructure.

2.5 Related Work

Previous work on this topic has been centered around MITRE CALDERA[7], a tool for automated adversary emulation. We reviewed CALDERA in detail, but we found two main issues:

1. CALDERA only covers a small subset of ATT&CK
2. The dependency model in CALDERA is tool specific

The latter point in particular prevented us from using CALDERA as a basis. CALDERA defines requirements for attack steps, but such a requirement could be that the tool "wifi.sh" from the CALDERA toolkit is present on the target system. CALDERA does not define dependencies in a way that can be used outside of CALDERA.

Andy Applebaum from MITRE held a presentation titled "Finding Dependencies Between Adversary Techniques" at the Annual FIRST Conference in 2019[8]. Our work in the SOCCRATES project was inspired by the semantic analysis in this presentation, but Applebaum used CALDERA as a basis, with

[6] https://github.com/center-for-threat-informed-defense/adversary_emulation_library

[7] https://github.com/mitre/caldera

[8] https://www.first.org/resources/papers/conf2019/1100-Applebaum.pdf

the same issues as described in the previous paragraph. Applebaum highlighted that modelling technique dependencies has a high barrier to entry, and that it is hard to maintain and extend such a model.

3 Modelling Technique Dependencies

3.1 Promise Theory

Our approach to modelling technique dependencies is inspired by promise theory, first proposed by Mark Burgess in 2004 [1]. Promise theory was originally developed as a model for policy based management of distributed systems, where consistency and consensus arise from voluntary cooperation. Autonomous agents give "promises", or statements or intended behaviour, to each other. An example promise from [1] is "will send database data in less than 5 ms", where a database server makes the promise to a web server. The autonomous agents and promises form a graph with autonomous agents as nodes and promises as directed edges.

Adapting this theory to our work, we define techniques as autonomous agents. Techniques require certain promises in order to be executed, and these promises can be provided by other techniques. The main benefit of this approach is that techniques can be assessed independently of each other: an analyst can review a technique and assign promises to it without having to review other, related techniques. The analyst only considers which of the promises that are required by the technique, and which promises that the technique provides. Thus, when a new technique is added to ATT&CK, it does not matter whether there are 10 or 1000 related techniques: the time to assess it stays the same. This addresses Applebaum's concern that such models are hard to maintain and extend.

3.2 Requires and Provides

In our model, a technique gives two types of promises: "requires" and "provides". The "requires" promises correspond to acceptance promises from [1]: the technique will use these promises if they are provided by other techniques. The "provides" promises correspond to conditional promises from [1]: they will be upheld if and only if all of the technique's "requires" promises are present (AND logic). Finally, we define a special type of "provides" promises: end conditions. An end condition represents the attacker's objective, i.e. what they are trying to achieve. Examples of end conditions include exfiltration, data destruction, and denial of service.

3.3 A Vocabulary of Promises

In order to operationalise our model, we needed a vocabulary of promises. The vocabulary had to match the abstraction level of techniques from ATT&CK, i.e. the promises could not be too detailed (low level), and not too general. Furthermore, we needed a vocabulary where the promises could be used as both "requires" and "provides". We investigated existing vocabularies from the Structured Threat Information eXpression (STIX)[9] and Malware Attribute Enumeration and Characterization (MAEC)[10] standards, but we found no suitable vocabularies. We thus developed a new vocabulary and made it available on Github[11]. Promises with the prefix "objective_" are end conditions.

[9] https://oasis-open.github.io/cti-documentation/

[10] https://maecproject.github.io/

[11] https://github.com/mnemonic-no/aep-data/blob/master/data/promise_descriptions.csv

3.4 Mapping Technique Dependencies

The most time consuming part of our work was to review every technique and sub-technique in ATT&CK and assign promises to them. A team of experienced threat analysts iteratively developed the vocabulary and assigned the promises to techniques. The result of this work is a JavaScript Object Notation (JSON) file of techniques with corresponding "requires" and "provides" promises, which is published on Github[12]. A graphical representation of two example techniques and their promises is shown in Figure 3, where the oval shapes represent techniques and the rectangles represent promises. Incoming arrows represent "requires" promises, and outgoing arrows represent "provides" promises.

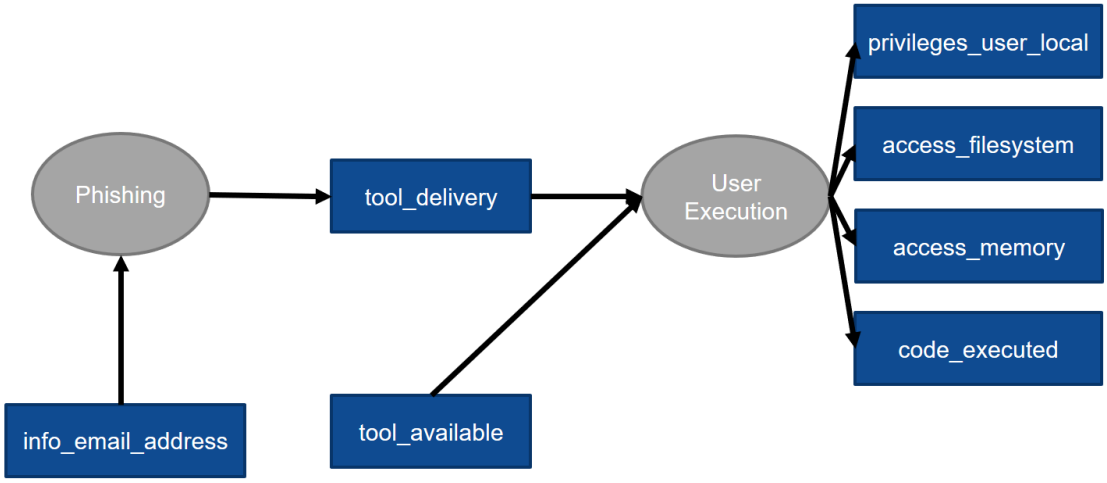

Figure 3: The techniques "Phishing" and "User Execution" with related promises

4 Implementation

4.1 The Adversary Emulation Planner

Having developed a vocabulary of promises and mapped them to techniques, we proceeded to create a command line tool to generate attack stages from a list of techniques: the Adversary Emulation Planner. The list of techniques could be techniques associated with a specific threat actor, campaign or incident. The tool is written in the Python programming language and the tool with documentation is published under the ISC Open Source license on Github[13]. The tool works as follows:

Inputs

1. Techniques with associated "requires" and "provides" promises[12]
2. A list of techniques, either an ATT&CK Navigator Layer or a JSON formatted list
3. An end condition (objective), provided as a command-line parameter

Initialisation

1. Initialise an empty queue
2. Set the attack stage to 1

[12] https://github.com/mnemonic-no/aep-data/blob/master/data/technique_promises.json
[13] https://github.com/mnemonic-no/aep

Execution

The tool will loop through the following until no more techniques can be executed:

1. Execute all techniques where:
    a. The technique is not tagged with an attack stage AND
    b. All "requires" promises for the technique are present in the queue
2. Add the "provides" promises from the executed techniques to the queue
3. Tag the executed techniques with the current attack stage
4. Increment the attack stage by 1

Output

Finally, the tool iterates through each attack stage and prints the techniques executed during that stage. It also supports printing the new promises provided at each stage, as well as the tactics implemented by the techniques executed at each stage. If the end condition specified during intitialisation is present in the queue, then the tool will output that the attack was successful, otherwise it will output that the attack failed. Figure 4 shows an example tool output for the SolarWinds incident[14] with "objective_exfiltration" as the end condition. The list of techniques observed during this incident was provided as input to the tool as an ATT&CK Navigator Layer[15].

The tool also provides additional functionality to help manage promises and mappings to techniques:

- Summary of promise usage
- Summary of a single technique
- Summary of a technique list/bundle
- Search for promises

[14] https://www.sans.org/blog/what-you-need-to-know-about-the-solarwinds-supply-chain-attack/

[15] https://raw.githubusercontent.com/center-for-threat-informed-defense/public-resources/master/solorigate/UNC2452.json

stage	techniques	new promises @end-of-stage	tactics
Attack stage 1	Develop Capabilities (Resource Development) Develop Capabilities:Malware (Resource Development) Domain Trust Discovery (Discovery) Obtain Capabilities (Resource Development) Obtain Capabilities:Code Signing Certificates (Resource Development) Supply Chain Compromise (Initial Access) Supply Chain Compromise:Compromise Software Supply Chain (Initial Access)	exploit_available info_domain_trust infrastructure_certificate privileges_user_local tool_available tool_delivery	Discovery Initial Access Resource Development
Attack stage 2	Command and Scripting Interpreter (Execution) Command and Scripting Interpreter:PowerShell (Execution) Command and Scripting Interpreter:Windows Command Shell (Execution) Scheduled Task/Job (Execution,Persistence,Privilege Escalation)	defense_evasion execute_code file_transfer persistence	Execution Persistence Privilege Escalation
Attack stage 3	Application Layer Protocol (Command and Control) Application Layer Protocol:Web Protocols (Command and Control)	access_filesystem access_host access_network command_and_control	Command and Control
Attack stage 4	Account Discovery: Local Account (Discovery) Dynamic Resolution (Command and Control) [*] Dynamic Resolution:Domain Generation Algorithms (Command and Control) [*] Event Triggered Execution (Persistence,Privilege Escalation) [*] Ingress Tool Transfer (Command and Control) [*] Permission Groups Discovery (Discovery) Permission Groups Discovery:Domain Groups (Discovery) Process Discovery (Discovery) Unsecured Credentials (Credential Access) Unsecured Credentials:Private Keys (Credential Access)	credentials_user_domain credentials_user_local credentials_user_thirdparty info_groupname info_process_info info_target_employee info_username	Command and Control Credential Access Discovery Persistence Privilege Escalation
Attack stage 5	Account Manipulation:Additional Cloud Credentials (Persistence) [*] Cloud Service Discovery (Discovery) Email Collection:Remote Email Collection (Collection)	info_cloud_services privileges_user_domain target_information	Collection Discovery Persistence
Attack stage 6	Archive Collected Data (Collection) Archive Collected Data:Archive via Utility (Collection) Data Staged (Collection) Data Staged:Remote Data Staging (Collection) Exfiltration Over Web Service (Exfiltration)	objective_exfiltration staged_data	Collection Exfiltration

```
[*] Technique does not provide any new promises
The following objectives were reached: ['objective_exfiltration']
SUCCESS: Attack chain exited with end condition 'objective_exfiltration'
```

Figure 4: Tool output for the SolarWinds incident

4.2 TIP Integration

The next step in our work was integration with the SOCCRATES Threat Intelligence Platform (TIP) component. To achieve this, we created a web version of the Adversary Emulation Planner tool, available under the ISC Open Source license on Github[16]. We then extended the functionality of the TIP GUI, such that an analyst can select a threat actor, campaign or incident in the GUI, and then click a button to generate an AEP. The web version of the tool also supports JSON output that can be used by the SOCCRATES ADG Analyser.

4.3 ADG Integration

One of the scenarios in SOCCRATES Use Case 2 (Reception of new Cyber Threat Intelligence) is triggered when the TIP receives information that a threat actor as been observed to use a technique that was previously not associated with that threat actor. The TIP will send the name of the threat actor to the SOCCRATES Orchestration and Integration Engine (OIE), which will forward it to the ADG Analyser. The ADG Analyser will first access the TIP API to receive the list of techniques associated with the threat actor. Next, it will submit the list of techniques to the Adversary Emulation Planner web interface to receive the set of attack stages with the associated techniques in JSON format.

[16] https://github.com/mnemonic-no/aep-web

5 Results

5.1 Experimental Results

We have conducted extensive experiments with the tool, covering both adversary groups from ATT&CK and techniques observed in campaigns and incidents. During the experiments, we observed significant intelligence gaps. None of the adversary groups in ATT&CK had a set of associated techniques that would be sufficient to carry out an intrusion. We found that our tool helped us to quickly identify gaps and generate candidates for missing techniques. We could then examine the original reports that the information in ATT&CK is based on, and identify the missing techniques. We submitted our findings to MITRE for inclusion in ATT&CK.

5.2 Contributions to MITRE ATT&CK

Our first contributions were included in ATT&CK version 10, published in October 2021, and due to this the SOCCRATES project is now listed as a contributor to ATT&CK[17]. We expect to be able to submit further contributions to ATT&CK with the help of the Adversary Emulation Planner.

6 Future Work

6.1 Vocabulary and Mappings

Our vocabulary of promises is a starting point that has worked well in practice. As we gain more experience with the tool the vocabulary should be refined and extended. The same applies to the mapping of promises to techniques and sub-techniques.

6.2 Standardisation

Long term, we hope to standardise the vocabulary and mapping of promises to techniques and sub-techniques, with the objective to maintain this work beyond the lifetime of the SOCCRATES project. Furthermore, we hope that MITRE will consider the integration of this information in ATT&CK, such that defining dependencies for new techniques is part of the standard process for adding new techniques to ATT&CK. Doing so would enable more automation in adversary emulation

[17] https://attack.mitre.org/techniques/T1588/002/

7 References

[1] Burgess M. (2005) An Approach to Understanding Policy Based on Autonomy and Voluntary Cooperation. In: Schönwälder J., Serrat J. (eds) Ambient Networks. DSOM 2005. Lecture Notes in Computer Science, vol 3775. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11568285_9

8 Abbreviations

This glossary serves as inventory of abbreviations used in the SOCCRATES documents.

Acronym	Description
ACT	semi-Automated Cyber Threat intelligence
ADG	Attack Defence Graph
AEF	Argus Event Format
AEP	Adversary Emulation Plan
AI	Artificial Intelligence
AIT	AIT Austrian Institute of technology
API	Application Programming Interface
APT	Advance Persistent Threat
ATOS	ATOS Spain
ATT&CK	Adversarial Tactics, Techniques and Common Knowledge
AV	AntiVirus
BPMN	Business Process Model and Notation
CC	Command and Control
CERT	Computer Emergency Response Team
CMDB	Configuration Management Database
CSIRT	Computer Security Incident Response Team
CoA	Course of Action
CTI	Cyber Threat Intelligence
DC	DataCentre
DGA	Domain Generated Algorithm
DNS	Domain Name System
EDR	Endpoint Detection and Response
ELK	Elasticsearch/Logstash/Kibana
FRS	Foreseeti
FSC	F-secure
ICT	Information and Communication Technology
IDS	Intrusion Detection System
IMC	Infrastructure Modelling Component
IMT	Institut Mines Télécom – Télécom SudParis
INTF	Interface
IoC	Indicators of Compromise
IP	Internet Protocol
IPS	Intrusion Prevention System
IRM	Incident Response and Management
ISC	Internet Systems Consortium
ITIL	Information Technology Infrastructure Library
JSON	JavaScript Object Notation
GUI	Graphical User Interface
KTH	Kungliga Tekniska högskolan - Royal Institute of Technology
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
M_n	Infrastructure Model (at time *n*)

MAEC	Malware Attribute Enumeration and Characterization
MNM	mnemonic
MSSP	Managed Security Service Provider
MTTD	Mean Time To Detection
NOC	Network Operations Centre
OT	Operational Technology
OS	Operating System
RORI	Return on Response Investment
SDN	Software Defined Network
SHS	Shadowserver
SIEM	Security information and event management
SOAR	Security Orchestration, Automation and Response
SOC	Security Operation Centre
SOCCRATES	SOC & CSIRT Response to Attacks & Threats based on attack defence graph Evaluation Systems
SSL	Secure Sockets Layer
STIX	Structured Threat Information eXpression
TAP	Test Access Point
TIP	Threat Intelligence platform
TLS	Transport Layer Security
TNO	Nederlandse Organisatie voor toegepast natuurwetenschappelijk onderzoek
TTC	Time To Compromise
UC	Use Case
VLAN	Virtual LAN
VM	Virtual Machine
VTF	Vattenfall