**ENHANCED ARGUMENTATION BASED REASONER (E-ABR)**

**CHAPTER ONE**

* The main goal of this project is to construct an ABR to aid analysts in addressing the attribution problem and produce comprehensive explanations to support attribution.
* ***Objective 1***: present a list of entities (countries or hacker groups) that could be responsible for the cyber attack
* ***Objective 2***: provide a coherent explanations for the results of attribution
* ***Objective 3***: when there are more than one result for a given attack be able to rank the results by their relative strength
* ***Objective 4***: Highlight gaps in reasoning – highlight to the user any assumptions or potential gaps in the reasoning used during attribution
* ***Objective 5***: Show path for further investigation when there are insufficient evidence to make an attribution, suggests other information that can be given in order to successfully perform the attribution of cyber attack
* The reasoning rules in the reasoned are constructed by analyzing attributions of well known cases of cyber attacks and extract the reasoning process used by investigators and analysts who worked on those cases.

**CHAPTER TWO**

* A model proposed by Rid and Buchnan called the ***Q-Model*** divides the attribution process into 3 distinct layers: technical, operational and strategic
* It describes attribution as an incremental process each of the layers building on top of each other to form the final attribution.
* **Technical:** objective of this layer is to find out how the attackers comprised the system. E.g. of such evidences include: zero day vulnerabilities, signatures or unique traits in malware, trait like language indicators etc
* **Operational:** combines the information from various sources including derived information from technical layer, non-technical layer (social) evidences and geopolitical context.
* **Strategic:** uses information from the technical and operational layer in attempt to draw a conclusion

**GORGIAS**

* Gorgias is a general argumentation framework that combines ***preference*** and ***abduction***.
* The syntax for using Gorgias in Sicstus Prolog:
* ***Query: prove (Goals, Delta)***
* ***Rule: rule (Label, Head, Body)***
* ***Preference: prefer (Label1, Label2)***
* ***Conflict: conflict (Label1, Label2)***
* ***Abducible: abducible (abduciblePredicate(\_), [])***
* ***Fact: rule (Label1, Fact, [])***
* The Gorgias Syntax adds another layer on top of the basic Prolog construct syntax.
* All Gorgias constructs can be written in the following form:

***rule (Label, Head, Body)***

* where **Label** is a string, **Head** is a single literal and **Body** is a list of literals.
* Here we show the Prolog facts, rules and how queries translate into its equivalent form in Gorgias.



* If we use Prolog system predicate, might not work when placed inside the Gorgias rule/3 predicate.
* The following query fails when run with Gorgias-Visual on SWI-Prolog even though the literal is clearly true.

*prove( [ member ( [a, [a, b, c]])], D).*

*false*

* That is why in some of our rules we use Prolog definitions outside of the Gorgias rules.



* In general, when using Prolog system predicates as part of the body, we put them outside of Gorgias rules Prolog conditions and when using predicates defined using Gorgias rules, we put them outside of Gorgias rules using Prolog conditions and when using predicates defined using Gorgias rules, we put them inside the Gorgias rules.

**CHAPTER THREE**

* The user (analyst) interacts with the GUI which returns and displays the results to the user.
* ABR is designed to be used in an iterative process, the user can add more evidences, rules or preferences after evaluating the results produced by ABR.
* The users will input a set of evidences (technical and social) as well as rules and preferences through the GUI, which can trigger the integrated forensic tools to automatically extract some evidence.
* Standard execution returns the list of possible culprits while verbose execution run when standard execution does not return any results.
* Verbose execution return suggestions of what other evidences are required in order to get a result in standard execution.
* These evidences/rules/preferences are then used by the ABR reasoner, composed of the core rules and background knowledge, to execute the requested query; the result of which is returned to the GUI.
* The result can take two different forms, depending on which mode of execution (standard or verbose) the ABR is executed in.
* **Standard execution** returns the list of possible culprits; while **verbose execution**, run when standard execution does not return any result, returns suggestions of what other evidences are required in order to get a result in standard execution.
* The reasoner is the main component in ABR. It is constructed from two parts, (i) core Gorgias rules and (ii) background facts.
* The rules and background facts are used to analyse the given evidence and to produce a result (attribution of cyber attack) that is returned to the user.
* Core Gorgias Rules The core rules of the ABR are built from well-known attribution cases.
* We study a full range of cyber attack cases in detail, extracted the reasoning used by investigators and converted them into rules.
* The details of which rules are extracted from which cases can be found in Appendix B.
* The background knowledge contains information in two broad categories: general knowledge and domain specific knowledge
* General knowledge that models the common sense and knowledge that analysts use during investigation, e.g., which countries use English as their first language.
* Domain-specific knowledge that models past experiences and knowledge of analysts, e.g., what are the prominent Advanced Persistent Threat (APT) groups and what are their profiles.
* Details of the rules and background facts used in ABR can be found in Sections 4.3 and 4.4.
* Each derivation returned by ABR has a numerical score.
* The score can serve as a guideline for users to decide which derivation is stronger.

**CHAPTER FOUR**

**ABR REASONER**

* The Q-model consists of three main layers: technical, operational and strategic.
* To illustrate, we show below some examples of rules for each of the three layers:
* **Technical:**

***requireHighResource(Att) ← usesZeroDay(Att).***

* If the attack (Att) uses zero-day vulnerabilities (usesZeroDay(Att)), then we say that it requires a lot of resources (requireHighResource(Att)).
* **Operational:**

*hasCapability(X, Att) ← hasResources(X), requireHighResource(Att).*

* If the attack (Att) required a large amount of resources (requireHighResource(Att)), and an entity X has (large amounts of) resources (hasResources(X)), then X has the capability to carry out the attack (hasCapability(X, Att)).
* **Strategic**

***isCulprit(X, Att) ← hasCapability(X, Att), hasMotive(X, Att).***

* If X has both the motive (hasMotive(X, Att)) and the capability (hasCapability(X, Att)) to carry out the attack, then X is the culprit (isCulprit(X, Att)).
* The core rules are the Gorgias rules that performs the reasoning behind the attribution.
* These rules are crafted by studying various attribution cases and extracting the reasoning that investigators employed to make the attribution and translate them into argumentation rules.
* All three layers together, technical, operational and strategic, form the core rule in the reasoner.
* In this section, we give an overview of some of the strategic rules of the reasoner, then delve into one of the core rules of the reasoner to explain how the layers work together.
* We show below some of the rules from the strategic layer.
* These rules describe some circumstances where we can prove an entity to be a culprit (isCulprit(X, Att)) or not (¬isCulprit(X, Att)).

**str\_rule\_1:isCulprit(X, Att)** ← **existingGroupClaimedResponsibility(X, Att).**

**str\_rule\_2:isCulprit(X, Att)** **← hasMotive(X, Att), hasCapability(X, Att).**

**str\_rule\_3:isCulprit(X, A1) ← malwareUsedInAttack(M1, A1), similar(M1, M2), malwareLinkedTo(M2, X), notFromBlackMarket(M1),**

**notFromBlackMarket(M2).**

**str\_rule\_4:¬isCulprit(X, Att) ← ¬attackOrigin(X, Att).**

**str\_rule\_5 : ¬isCulprit(X, Att) ← ¬hasCapability(X, Att).**

**str\_rule\_6: ¬isCulprit(X, Att) ← target(X, Att).**

**Rule Walkthrough**

* To illustrate how the rules in the different layers are connected, we present, in detail, one rule from start to end, from the strategic layer, to operational layer, to technical layer.
* The rule we use in this example is the following rule from the strategic layer:

***str\_rule\_2 : isCulprit(C, Att) ← hasMotive(C, Att), hasCapability(C, Att).***

* Rule ***str\_rule\_2*** uses the predicates ***hasMotive/2*** and ***hasCapability/2***;
* ***hasMotive/2*** is a derived predicate proved in the operational layer and ***hasCapability/2*** is a derived predicate proved in the technical layer.

***hasMotive(X, Att)***

***op\_rule\_1 : hasMotive(C, Att) ← target(T, Att), industry(T), hasEconomicMotive(C, T), contextOfAttack(economic, Att), specificTarget(Att).***

This rule can be read as: if a country/group has an economic motive to attack the industry that was targeted in the attack, and the context of the attack was economic, and the attack had a specific target, then the country/group has motive to carry out the attack.

The predicates used in this rule are summarised as below:

* ***target(T, Att)*** is a ***base evidence***, it means ***T*** is the target of the attack Att.
* ***industry(T)*** is one of the background facts, it is true when T is an industry (see Table 4.1 for the full list of background facts).

Table 1: Background Rules

|  |  |  |
| --- | --- | --- |
| **S/N** | **Predicate Example** | **Explanation** |
| 1. | industry(infocomm) | Type predicate for industries |
| 2. | pollindustry(military) | Political Industries |
| 3. | norindustry(infocomm) | Non-political industry |
| 4. | country(united\_states) | Type predicate for countries |
| 5. | cybersuperpower(united\_states) | List of cyber super powers |
| 6. | gci\_tier(afghanistan, initiating) | Global Cybersecurity Index |
| 7. | gci\_tier(poland, maturing) | Global Cybersecurity Index |
| 8. | gci\_tier(russian\_federation, leading) | Global Cybersecurity Index |
| 9. | firstLanguage(english, united\_states) | First Language used in a country |
| 10. | goodRelation(united\_states, australia) | Good relation between countries |
| 11. | poorRelation(united\_states, north\_korea) | Poor relation between countries |
| 12. | prominentGroup(fancyBear) | Prominent hacker groups |
| 13. | groupOrigin(fancyBear, russian\_federation) | Country of origin of a group |
| 14. | pastTargets(fancyBear, [france, …, poland]) | Past targets of a hacker group |
| 15. | malwareLinked(trojanMiniduke, cozyBear) | Past attribution of malware |
| 16. | malwareUsedInAttack(flame, flameattack) | Past attribution of malware |
| 17. | ccServer(gowin7, flame) | C&C servers of malware |
| 18. | domainRegisteredDetails(gowin7, adolph\_dybevek, prinsen\_gate\_6) | Domain registration details of C & C servers. |

* ***hasEconomicMotive(C, T)*** is also a base evidence.
* When ***hasEconomicMotive(countryX, industryY )*** is true, it means countryX will benefit economically from attacking industryY.
* For example, if countryX has identified industryY as a strategic emerging industry in official public documents such as white papers or other government reviews, we say that ***hasEconomicMotive(countryX, industryY )*** is true.
* ***specificTarget/1*** An attack is assumed to be targeted unless more than one country is targeted (***targetCountry/2***) , or there are technical evidence that show that the malware was constructed specifically for attacking that particular organization or country (***specificConfigInMalware(M)***).
* ***contextOfAttack/2*** can be proven if the target is a ‘normal’ industry.
* A ‘normal’ industry, as opposed to a ‘political’ industry, are industries that are not closely related to a country’s national interests.

***Rule 2*** Another operational rule used to derive ***hasMotive/2*** is given below:

***op\_rule\_2 : hasMotive(C, Att) ← targetCountry(T, Att), attackPeriod(Att, Date1), hasPoliticalMotive(C, T, Date2), dateApplicable(Date1, Date2), contextOfAttack(political, Att), specif icTarget(Att).***

* This rule can be interpreted as: if a country has a political motive (we will explain later what this means) to attack the target country, and the order of events fits into the general timeline (more on ***dateApplicable/2*** later), and the attack was a targeted attack, then the country has motive to perform the attack.
* The predicates used in this rule can be summarised below:
* ***targetCountry(T, Att)*** is a base evidence, denoting that the country T is the country of the target of the attack Att.
* ***attackPeriod(X, Date)*** is also a base evidence, it denotes the date of the attack.
* Date is a list, in the format [YYYY, MM].
* ***hasPoliticalMotive(C,T,Date2)*** is a derived predicate.
* We have one rule that derives ***hasPoliticalMotive(C, T, Date2)***.
* If the target country has imposed sanctions on country C, then as a form of retaliation, country C has political motive to attack the target country T.
* ***dateApplicable(Date1, Date2)*** is an auxiliary rule used to ensure that the triggering event (in this case, the imposing of sanctions) takes place before the attack, and occurred shortly before the attack.
* Alternatively, if the event is long-term and ongoing, we use the constant ongoing in place of the date of the event in the [YYYY, MM] format, ***dateApplicable(\_, ongoing)*** is always true.
* ***contextOfAttack(political, Att)*** can be proved if either the target is a country, or the target is a ‘political’ industry (industries that are closely related to a country’s national interests, such as the military or energy sector).

We show below the operational rules used to derive the above mentioned predicates.

***op\_rule\_3*** : ***hasPoliticalMotive(C, T, Date) ← imposedSanctions(T, C, Date)***.

***op\_rule\_4 : contextOf Attack(political, Att) ←target(T, Att), country(T)***.

***op\_rule\_5 : contextOf Attack(political, Att) ←target(T, Att), industry(Ind, T ), politicalIndustry(Ind).***

**Rule 3** We move on to the next operational rule to derive ***hasMotive/2***.

***op\_rule\_6:hasMotive(X, Att) ←target(T, Att), attackPeriod(Att, Date1), news(Event, T, Date2), dateApplicable(Date1, Date2), causeOfConflict(X, T, Event), specificTarget(Att).***

The general idea of this rule is that if

* an incident occurred in the target country and was publicized, and
* that incident is the cause of international conflict or tension with another country shortly before the attack, then the other country has motive to attack the target country.

To better explain this rule, we use a real-world example. The Sony Pictures hack in 2014 was attributed to North Korea, that allegedly attacked Sony Pictures in retaliation for the upcoming North Korean-based comedy “The Interview”. The relevant evidences are as follows: ***target(sony, sonyhack).***

***attackP eriod(sonyhack, [2014, 11]).***

***news(theInterview, sony, [2013, 10]).***

***causeOfConflict(north korea, sony, theInterview).***

**hasCapability(X, Att)**

* Next, we briefly cover the explanations for ***hasCapability/2***, which is defined in both the operational and technical layer.
* We show below one of the rules in the operational layer that defines ***hasCapability(X, Att)***.

***op\_rule\_7:hasCapability(X, Att) ← requireHighResource(Att), hasResources(X).***

* The predicate ***requireHighResource(Att)*** can be derived from the technical layer.

We show two such rules that derives ***requireHighResource(Att)***:

***tech\_rule\_1: requireHighResource(Att) ← target(T, Att), highSecurity(T).***

***tech\_rule\_2 : requireHighResource(Att) ← highVolumeAttack(Att),***

***longDurationAttack(Att).***

* The predicates ***highSecurity(T)***, ***highVolumeAttack(Att)*** and ***longDurationAttack(Att)*** are all base evidences.
* They have the following definitions:
* ***highSecurity(T)*** (Organisation or company) T has high-security.
* ***highVolumeAttack(Att)*** The attack had a high volume.
* ***longDurationAttack(Att)*** The attack was performed over a long duration (few months or even years).

**Background Knowledge**

* The other part of the reasoner comprises of the background knowledge or background facts.
* Not all evidences or facts are given by the user, the background knowledge consists of other non-case-specific information.
* There are two main types of facts compiled in the background knowledge - (i) general knowledge, and (ii) domain-specific knowledge.
* We show in Table 4.1 the full list of predicates in the background file used by ABR.

**General knowledge**

* The general knowledge consists of information about the characteristics of countries, such as the first language used in the country and a measure of how advanced the country’s cyber security scene is; international relations between nations; classification of the types of industry.
* These facts can be used together with evidences given by the user to make the attribution.
* Below we illustrate how the predicates are used in the rules in the various layers.

**Language indicators**

* Language indicators in malware can provide useful clues regarding the possible origin of attacks [10].
* There are two types of language artifacts: (i) default system language settings and (ii) names used in code.
* In the technical layer, the following rules makes use of these language artifacts to deduce

the origin of the attack:

***lang1 : attackPossibleOrigin(X, Att) ← sysLanguage(L, Att), firstLanguage(L, X).***

***lang2 : attackPossibleOrigin(X, Att) ← languageInCode(L, Att), firstLanguage(L, X).***

* If it is found that the general configuration language settings of the attacker’s machine was language L, ***(sysLanguage(L, Att))***, and the first language of country X is L, ***(firstLanguage(L, X))***, then the attack might have originated from country X.
* Otherwise, if names of variables or folders in the code are words from a certain language or reference a specific cultural reference from a specific language ***(languageInCode(L, Att))***, then similarly, a country that uses L as its first language is a possible candidate for attack origin.
* **Capability of nation**
* The capability of a nation limits the level of attacks it can possibly sponsor or carry out.
* We classify the amount of resources owned by a country into three different groups.
* A country can (i) have large amount of resources (hasResources/1) (ii) not have large amount of resources (¬(hasResources/1)) (iii) have no resources (hasNoResources/1).
* To estimate a country’s cyber capabilities, we use the Global Cybersecurity Index (GCI) Groups 2017.
* The GCI is an integrated index that assesses countries based on their commitment to five core pillars: legal, technical, organisational, capacity-building and cooperation [39].
* There are three GCI groups: leading, maturing and initiating.
* We represent a country’s GCI group using the predicate gci\_tier/2.
* In the operational layer, we have rules for whether a country is capable of carrying out an attack (hasCapability/2):

***hasResources1 : hasResources(X) ← gci\_tier(X, leading).***

***hasResources2 : hasResources(X) ← cybersuperpower(X).***

***hasResources3 : hasNoResources(X) ← gci\_tier(X, initiating).***

* We classify a country as hasResources/1 if it is either in the ‘leading’ GCI group or is recognised as one of the cyber superpowers [40].
* Countries in the ‘initiating’ GCI group are classified as hasNoResources/1.
* International relations Good international relations between two countries can indicate that a state-sponsored attack is unlikely to happen.
* While it is difficult to accurately portray relations between all countries, we used some simple rules and statistics [41, 42, 43] to compile a list of facts regarding whether two countries have a good relationship (goodRelation/2).
* Information about relations between countries are incorporated in the operational layer to narrow down the countries that might have a motive to carry out the attack.

***geopolitics1 : ¬hasMotive(C, Att) ← targetCountry(T, Att), country(T), country(C), goodRelation(C, T ).***

* It should be recognized that this is an overly simplistic approach for classifying international relations.
* Public opinions on international relations could be different on the actual situation between two countries, or outdated, or it might not even be possible to simply classify a relation as simply ‘good’ or not.
* Therefore, even though we are able to prove that a country likely does not have motive to carry out an attack based on the estimated relation status, we have added preferences to prefer other rules when there is a conflict between the other rules and the rules that uses the background information on geopolitical situation.

**Domain-Specific Knowledge**

* Domain-specific knowledge consists of information about (i) prominent groups and (ii) past attacks.
* These facts are primarily used in the strategic and technical layer.
* Prominent APT groups Advanced Persistent Threat (APT) groups are well-organised hacker groups that usually pursue their objectives over months or years [44].
* Due to their scale, many prominent APT groups are receive instructions and backing from nation states.
* We have extracted information on prominent APT groups from FireEye’s report on APT groups [44] and Martin’s article [45].
* Each prominent APT group has (where available) the following facts:

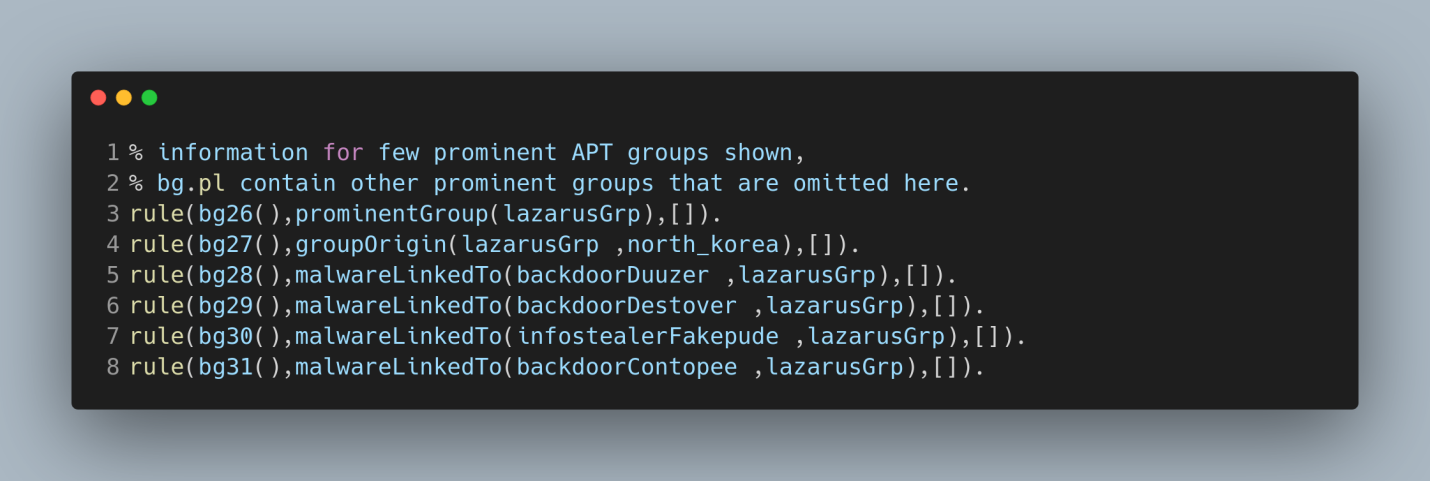
• Name or ID of the group;

• Country of origin of group;

• Countries/organisations targeted by the group in the past;

• Malware linked to the group (malware suspected to be made by group).

* We show below some example evidences regarding prominent APT groups that can be found in the background knowledge.
* These information are used in both the strategic and operational layer.



* Since prominent APT groups are APT groups that are active and have past records of conducting long attacks on other organizations, they, by definition, have large amounts of resources, and are thus capable of carrying out any attack (denoted by the rule below):

***str\_rule\_1 : hasCapability(X, Att) ← prominentGroup(X).***

* We can also link the APT group to its country of origin.
* If we are able to attribute the attack to an APT group and the country of origin of the APT group has motive to carry out the attack, we attribute the attack to the country (denoted

by the rule below):

***str\_rule\_2 : isCulprit(C, Att) ← prominentGroup(Group), groupOrigin(Group, C), country(C),***

***isCulprit(Group, Att), hasMotive(C, Att).***

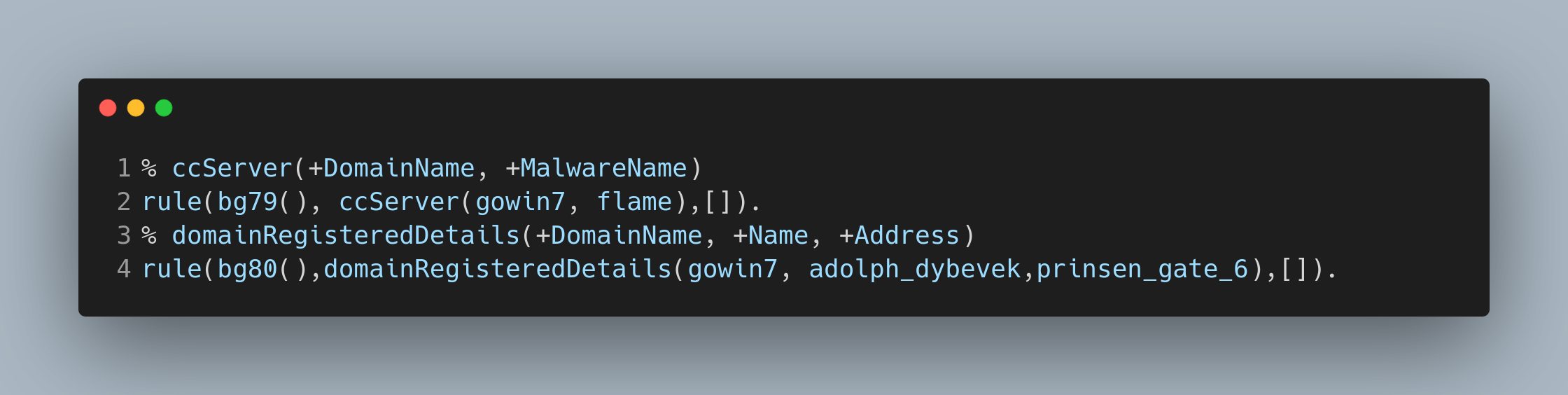
* Similarity to an APT-linked malware might also indicate that the culprit might be the same APT group (denoted by the rule below):

***str\_rule\_3 : isCulprit(X, A1) ← malwareU sedInAttack(M 1, A1), similar(M1, M2), malwareLinkedTo(M 2, X), notF romBlackMarket(M1), notF romBlackMarket(M2).***

* In the operational layer, APT groups that have attacked the same targets before has motive to attack them again.
* While this alone is definitely insufficient to make the attribution, combined with other evidences, it can steer us towards the culprit of an attack (denoted by the rule below):

***op\_rule : hasMotive(Group, Att) ← target(T, Att), prominentGroup(Group), pastTargets(Group, T s), member(T, T s).***

* Past attacks Information like the command and communication (C&C) server (ccServer/2) its domain registration details (domainRegisteredDetails/3) are included in the background file.
* We show an example of such details that can be found in **bg.pl**:



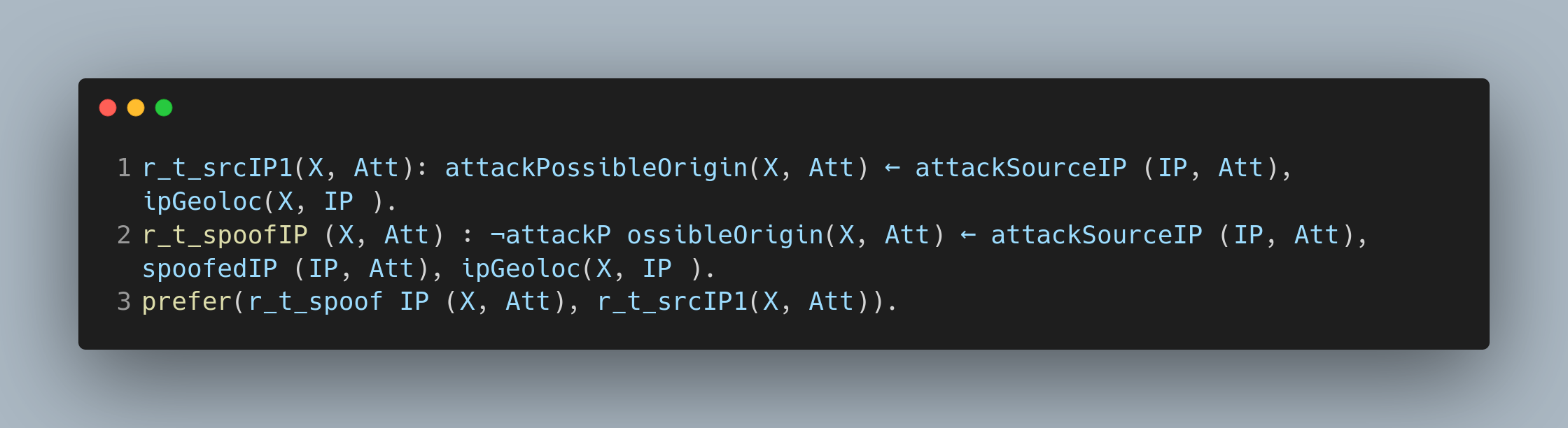
* These information can help us to spot similarities between past attacks and the current attack in the technical layer.
* We use the predicate ***similar(M1, M2)*** to denote that two malware are similar to each other.
* This similarity can stem from both malwares using the same C&C server, or the C&C server registered under the same name or address.

**Use of preference-based argumentation**

* In the next two sections we cover how preferences and abducible predicates were used in ABR.
* ABR uses preference-based argumentation, where preferences are used to specify relative strength between conflicting rules.
* To recap, conflicting rules are rules in which their heads are complementary to each other.
* So conflicting rules are any two rules with the form: ***rule1 : L ← . . . and rule2 : ¬L ← . . ..***
* In this section, we highlight the different ways in which preferences are used in the reasoner and justify why preferences are not used in some situations.

**Conflicting rules with preferences**

* Some conflicting rules have preferences.
* When the bodies of both rules are satisfied, only one can be proven by Gorgias.
* There are two different ways in which we use preferences.
* **Negation as failure:** Firstly, we use this to model negation-as-failure (NAF).
* In negation-of-failure (NAF), we say that ¬p can be derived when we fail to derive p.
* Since Gorgias facts does not have NAF by default, preferences are used to model NAF in the reasoner.
* **Contextual knowledge:** Secondly, we add preferences between conflicting rules that, using contextual knowledge, we determine that there is more reason to follow one rule and disregard the other.
* An example of such rules are the rules involving spoofed IP addresses.



* Normally, if we geolocated the source IP of the attack to a country, we derive that the attack originated from that country ***(attackPossibleOrigin(X, Att)).***
* However, if we were also able to derive that the IP was spoofed, then we should instead arrive at that conclusion ***¬attackP ossibleOrigin(X, Att).***

**Conflicting rules without preferences**

* We also left some conflicting rules without preferences.
* Such are for the situations where it might not be clear-cut whether to disregard one of the rules, and we leave the decision to the user.
* Since we run the negative derivation neg(isCulprit(X, Att)) for every positive derivation isCulprit(X, Att) (refer to Section 6.4 for more details),
* If a derived evidence could be disproved (or the negation of the derived evidence could be proved), then it will show up under the negative derivations.
* For example, given the following two conflicting technical rules:

***tech\_rule1(X, Att) : attackOrigin(X, Att) ← attackP ossibleOrigin(X, Att).***

***tech\_rule2(X, Y, Att) : ¬attackOrigin(X, Att) ← attackP ossibleOrigin(X, Att), attackP ossibleOrigin(Y, Att), country(X), country(Y ), X\ = Y.***

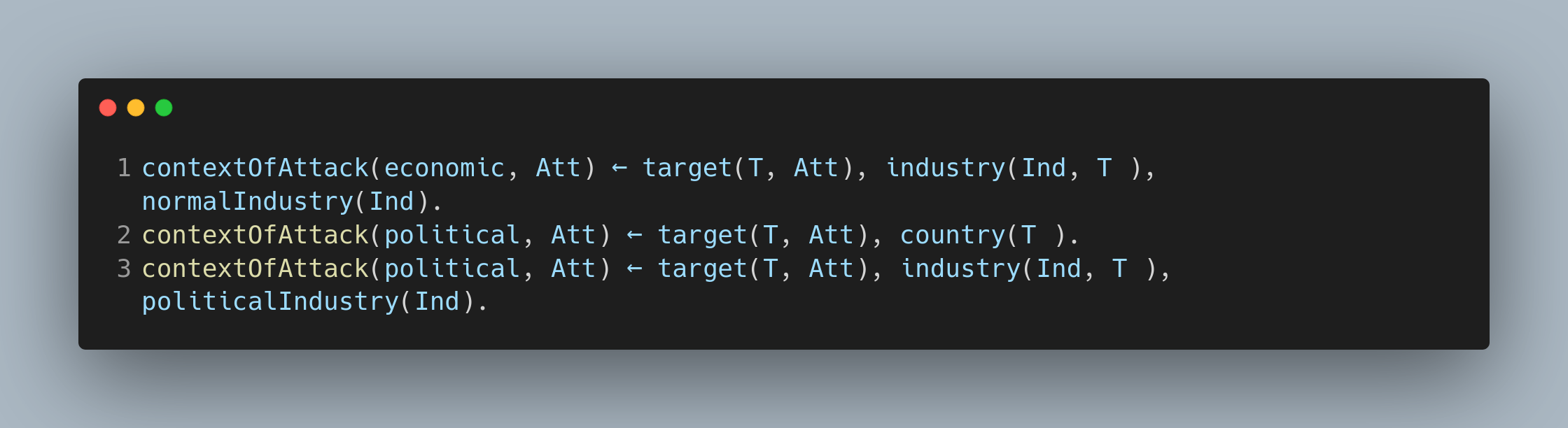
* We are able to prove attackOrigin(country1, attack) and attackOrigin(country2, attack) by using the first rule, and we are also able to prove ***¬attackOrigin(country1, Att)*** and ***¬attackOrigin(country2, Att)*** by using the second rule.
* This is the desired result since we want to leave the user with some flexibility in which rules they think is stronger in the specific use case.

**Use of abducibles**

* After declaring a predicate as abducible, it can be used like normal predicates in the Gorgias rules.
* Abducible predicates are predicates that can be assumed to be true.
* There are two abducible predicates used in ABR, specificTarget/1 and contextOf Attack/2.
* These two abducibles are used in different ways.
* For specificTarget/1, we have rules to disprove the abducible predicate:

***¬specif icT arget(Att) ← targetCountry(T 1, Att), targetCountry(T 2, Att), T1 \ = T2.***

* These rules can also be thought of as the integrity constraints.
* This models how some facts are assumed to be true unless proven otherwise.
* On the other hand, for *contextOfAttack/2*, we have rules to prove the abducible predicates:

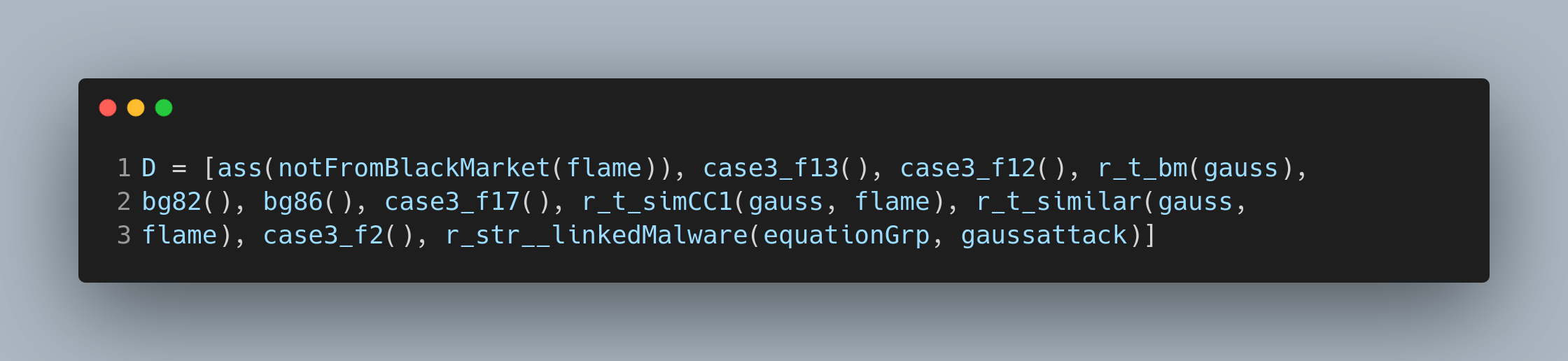


* The abducible is then used as part of other rules.
* In this case, the attribution will go through even without proving ***contextOfAttack/2***.
* It will be flagged up by the ABR as abduced predicates, prompting the user to look for more evidence in order to prove the contextOf Attack/2.
* This models the situation where there are incomplete information to fully make an attribution, but the analyst chooses to make some assumptions in order to put together other pieces of evidences.
* Later on, the analyst might go back and try to find more evidence to prove that the assumptions made during investigation were indeed true.

**CHAPTER FIVE**

**OTHER KEY COMPONENTS**

* When using ABR, if we have large amounts of evidence, our tool will likely produce several attributions, or different results for who was behind the attack.
* To reach a meaningful conclusion, it is essential that we construct a technique to rank or score different derivations.
* The above examples gives insight into how we intuitively decide if a rule is stronger.
* Generally, we say that a rule is stronger if its supporting evidences are more numerous and/or stronger.
* One simple way of scoring the derivations is to count the number of evidences used. Since there are two types of evidences: (i) case-specific evidence input by the user and (ii) background evidence, one possible refinement is to score case-specific evidence and background evidence differently.
* Case-specific evidence should weigh more than background evidence, since attributions based on more case-specific evidences should be stronger than attributions based mostly on background evidence.
* This alleviates the problem of always attributing attacks to enemy nations, regardless of case evidences, since the user will be able to see that the score of the derivation is low.
* When executing ***prove([isCulprit(X,A)], D)***, the result of ***D*** is the derivation, which is a list of all the rule names of rules used to prove the predicate.
* This include both rules and facts.
* Facts are a special case of rules, they are rules with no body predicates.
* We have chosen to assign each case-specific evidence a score of 3, and each background evidence a score of 1.
* Since users input evidences in Prolog syntax (Head :- Body1, Body2, ...) and all Gorgias rules are created automatically, we have full control over the rule names.
* The mapping of the ABR rulenames to their rule types and the files where they are located in is shown in Table 5.1.
* Case-specific evidences have rule names starting with ‘case\_’ and background evidences have rule names starting with ‘bg\_’
* To demonstrate the scoring system at work, we show the below example.



* The above derivation yields a score of 14, since there are 4 case-specific evidences {case3\_f13(), case3\_f12(), case3\_f17(), case3\_f2()} and 2 background evidences {bg82(), bg86()) used (4 ∗ 3 + 2 = 14}.
* The scores of all solutions are displayed to the user in the GUI, the user can decide if the score is to be taken into consideration when comparing different solutions for the attack.

**Forensic Tool Integration**

* ABR integrates with 4 different tools: Snort, Tor Exit Exporter, ip2nation IP geolocation and virustotal IP domain resolution
* The tool integrations are implemented in Java.
* The results are written into file as Gorgias facts and passed into the reasoner during the execution.