Using Attack Pattern for Cyber Attack Attribution

Florent Avellaneda
CRIM
Montréal, Canada
Florent.Avellaneda@crim.ca

El-Hackemi Alikacem
CRIM
Montréal, Canada
El-Hachemi.Alikacem@crim.ca

Femi Jaafar
CRIM
Montréal, Canada
Fehmi.Jaafar@crim.ca

***Abstract*—A cyber attack is a malicious and deliberate attempt by an individual or organization to breach the integrity, confidentiality, and/or availability of data or services of an information system of another individual or organization. Being able to attribute a cyber attack is a crucial question for security but this question is also known to be a difficult problem. The main reason why there is currently no solution that automatically identifies the initiator of an attack is that attackers usually use proxies, i.e. an intermediate node that relays a host over the network. In this paper, we propose to formalize the problem of identifying the initiator of a cyber attack. We show that if the attack scenario used by the attacker is known, then we are able to resolve the cyber attribution problem. Indeed, we propose a model to formalize these attack scenarios, that we call attack patterns, and give an efficient algorithm to search for attack pattern on a communication history. Finally, we experimentally show the relevance of our approach.**

***Index Terms*—Cyber Attribution, Cyber Attack, Attack Pattern**

I. Introduction

Cyber attribution has several definitions in the scientific literature. For example, Boebert [2] defined cyber-attribution as "the activity of analyzing malicious functionality and malicious packets, and using the results of the analysis to locate the node that initiated, or is controlling, the attack". Another more general definition considers that the identification of the location of an attacker or its intermediary is also a part of cyber attribution [15]. In this case, the location can be specified by the Internet Protocol (IP) address, account name, or geographical address [4]. In this paper, we consider the first definition of cyber attribution and seek to identify the machine that initiated a cyberattack against an information system.

Being able to attribute a cyberattack is crucial for several reasons. First, knowing the source of an attack generally allows us to deduce the intentions of the attacker. Thus, the target of the cyberattack can implement the necessary actions to face the attack adequately. The second reason is that cyber attribution may reduce the risks of cyber threats as potential attackers will care about the risk of being identified and punished. Thus, cyber attribution as a deterrence system is essential to scale up the state's credibility and security [6], [15]. Finally, cyber attribution will allow cyber analysts to prioritize incidents effectively and proactively based on adversary purposes and used technique [6].

Although the problem of cyber attribution is important, it is also known to be a challenging problem. One of the main challenges in cyber attribution is the reliability of the IP address of malicious packets related to a cyberattack [6]. Indeed, almost all cyber attacks carried out through proxies. Thus, we are interested in the problem of identifying a cyber attack that uses proxies.

In this paper, we assume that the entire history of the communications between machines, including the malicious messages of the attack, is available. In practice, it is sufficient that this set contains all the messages that were used in the attack. For technical feasibility, we consider only the sources/recipient in the communication history and the time they occurred. Recent work shows that this hypothesis is feasible in practice [11].

Considering that the attack has already happened, and the messages of the attack received by the target are identified within the communication history, our goal is to identify the initiator of this attack. We propose to use the concept of attack pattern to resolve the problem of cyber attribution. An attack pattern corresponds to the strategy used by the attacker to send messages (the number of proxies used, the order of messages, the timing between exchanges, etc). Attack pattern is realistic because the patterns can be determined from previous incidents and documented attacks and were widely used in previous works [8].

The paper is organized as follows. Section 2 reviews the related works. Section 3 and 4 presents the problem of cyber attribution. Section 5 explore the concept of finding an attack model in a history log and proof that this problem is NP-Complete. Section 6 proposes an algorithm to effectively solve this problem. Section 7 presents the experimental result to show the relevance of our approach. Finally, we conclude in section 8.

II. Related Works

Several approaches have been proposed and used to face cyber attribution challenges. We summarize such approaches with some examples of related work.

A. Query Hosts and "Hack Back"

Query hosts for internal state information requires that there be a pre-existing query function. If an attacker controls the host, this may alert the attacker and make the information much less reliable [7]. A "hack back" does this without the owner's permission, and clearly requires significant legal control. The inconvenient of this technique is that being able

to make a hack back is not an easy task and assumes that the attacker has a security flaw that we can exploit.

B. Transmit Separate Messages

When routers route a message, they can send a separate additional message to aid in attribution. The limitation of this approach is that if the separate messages are sent for all messages, this could easily overwhelm network resources. This idea was first proposed by Bellovin and other researchers as follows: every router should sample a packet with a small probability, copy its content onto a special ICMP packet, add information about the adjacent upstream and/or downstream routers, and send it towards the same destination as the original packet. The victim of an attack can then use these packets to reconstruct the paths back to the attackers [1].

C. Exploit Attacker Self-Identification

The information the attacker sends, intentionally or not, is used to identify him or her. In some cases, the defender can cause the attacker to send this data. When this technique works, it can directly reveal the attacker regardless of how well he or she is hidden usually, but many of these techniques depend on highly technical and specialized approaches (e.g., beacons, web bugs, cookies, and watermarking) that are easily foiled once an attacker knows about them. The defender can also specially mark data flowing back to the attacker, and then have intermediate systems detect these markings. This can trace the attacker through stepping stones, but requires detectors of these reverse flows and may be thwarted by encryption [15].

The defect of this technique is that an attacker can provide false information to pretend to be another person. Thus, from a theoretical point of view, this type of identification does not allow to be sure to correctly identify the initiator of an attack.

D. Reasoning-based Approach

A reasoning-based technique is used in artificial intelligence and knowledge-based systems to generate conclusions from available knowledge using logical techniques such as deduction and induction. These approaches model technical knowledge from attack activities, information about known attacks, as well as diverse knowledge such as motivations, political crises, and so on that can justify the attack; then, a reasoning mechanism allows to conclude at the possible origin of the attack [9], [10]. The problem with this approach is that an expert is required to build the knowledge base and correctly setting the probabilities. Unfortunately, this task is very complex.

III. Modeling the cyber attribution problem

The communication history between machines in a given network is modelled by a history log defined as follows.

Definition 1. *A* history log $H = (\mathcal{P}_H, \mathcal{M}_H, T)$ *is given by:*

- *A set of machines* $\mathcal{P}_H$ *in the network.*
- *A set of messages* $\mathcal{M}_H$*. Each element* $m \in \mathcal{M}_H$ *is a message. We denote by* $dom(m)$ *the machine that sends the message and by* $cod(m)$ *the machine that receives the message.*
- *A time function* $T : \mathcal{M}_H \to \mathbb{N}$*. If* $m \in \mathcal{M}_H$*, then we denote by* $T(m)$ *the time when the message exchange occurred.*

For efficiency, we consider time as a series of time steps which are a period of times represented by an integer. In our experiments, we use an interval of 10 ms as a time step. Also, for simplicity, we consider that a message exchange is instantaneous and assume that messages cannot be lost.

Let H be a history log and suppose there is a machine $P_A \in \mathcal{P}_H$ in the network that initiated an attack. Suppose we have a set of messages identified as malicious whose initiator is P_A. The question is identifying the machine P_A. This question is complicated when communications are not direct. Indeed, the machine P_A is able to send a message to a machine P_P asking it to send the message a to the target machine P_T. Even if the message has been labeled as malicious, the initiating machine of this message is then P_A; P_P is only an intermediate machine.

Note that we do not consider the content of the messages in our modeling, because it is too complicated to save such data in practice. In addition, data circulating on the network can be encrypted according to the communication protocols used.

We remark that finding the machine that initiates an attack m is an intractable problem. To illustrate this statement, let us consider the history log depicted graphically in Figure 1. There are two possibilities. The first possibility is that Carol asked Bob to send the message m to Alice. In this case, Carole is the initiator of the message m. The second possibility is that m' is not related to the m message and in this case, Bob is the initiator of the message.

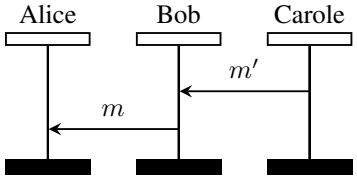

Fig. 1. Example of an MSC

IV. Attack Pattern

To make the attribution problem decidable from a history log, we are using the concept of attack pattern. An attack pattern corresponds to the strategy used by the attacker to send messages (the number of proxies used, the order of messages, the timing between exchanges, etc.). Several previous work [8] analyzed data and meta data in relation with cyber attacks to identify attack pattern. Concretely, if the attack is performed through software (each infected machine follows the instructions of the software), then an analysis of this software would extract the attack pattern.

In the rest of the paper, we are using the Messages Sequence Charts (MSC). The MSCs are a well-known formalism for describing interaction scenarios between remote sites. MSCs

have been the subject of much work in recent years. They benefit from a standard visual and textual representation (ITU-T recommendation Z.120) and are close to other formalisms such as UML sequence diagrams. Concretely, an MSC provides a graphical description of communications in a network.

An MSC can be seen as a directed acyclic graph where the nodes represent events (sending or receiving messages) and the arcs represent the order of relationships between the different events. Each event is also linked to a process. A process corresponds to a communicating entity (a machine, a thread, a user, etc.).

Considering that the message exchanges are instantaneous in our approach, we simplify the definition of MSC.

Definition 2. *An attack pattern $P = (\mathcal{P}_P, \mathcal{M}_P, T_{min}, T_{max})$, is defined by:*

- *A set of machines $\mathcal{P}_P \subseteq \mathcal{P}_H$.*
- *A set of messages $\mathcal{M}_P$.*
- *Complete functions $T_{min} : \mathcal{M}_P \times \mathcal{M}_P \to \mathbb{N}$ and $T_{max} : \mathcal{M}_P \times \mathcal{M}_P \to \mathbb{N}$ indicating the minimum and maximum elapsed time between any two exchanged messages, respectively.*

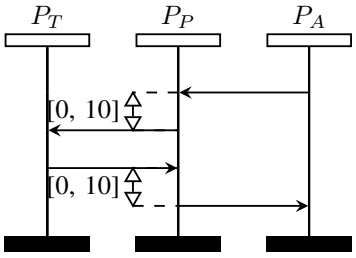

Fig. 2. Pattern of attack consisting in passing through one proxy.

All messages in an attack pattern are said to be malicious because they participate in an attack. In practice, malicious messages from the attack pattern intended for the target machine and returned by the target machine are recorded in the log history. A machine $\mathcal{P}_A \in \mathcal{P}_P$ is the initiator of the attack if the first message in the attack pattern was sent by $\mathcal{P}_A$. The goal is to identify which machine in the history log is $\mathcal{P}_A$.

A trivial example of an attack pattern is when an attacker targets directly the victim's machine without using a proxy; in this case, the attacker identification becomes trivial. Obviously, we are interested in more complex patterns than this one.

Example 3. *Suppose that an attacker goes through a single proxy machine to carry out an attack. The attack will correspond to sending a particular message in order to retrieve confidential information. This type of attack is common, and can be carried out using backdoors on software installed on the target machine, or by exploiting a buffer overflow bug. This attack pattern has been modeled in Figure 3. This pattern models the fact that an attacker $\mathcal{P}_A$ sends a message to a machine $\mathcal{P}_T$ via a proxy $\mathcal{P}_P$. Then, the $\mathcal{P}_T$ machine responds to $\mathcal{P}_A$ through the $\mathcal{P}_P$ proxy. We can notice that a time constraint has been put on the $\mathcal{P}_P$ process. Indeed, the standard proxy behavior is just to forward messages, so we could add the constraint that the proxy does not keep messages for more than 10 time steps, which correspond to 100 ms while the time step is set to 10 ms.*

V. Attack Pattern Identification in the History Log

In order to identify the initiator of an attack, we define a mapping between an attack pattern and a history log as follows.

Definition 4. *A mapping between an attack pattern $P = (\mathcal{P}_P, \mathcal{M}_P, T_{min}, T_{max})$ and a history log $H = (\mathcal{P}_H, \mathcal{M}_H, T)$ is a function $f : \mathcal{M}_P \to \mathcal{M}_H$ such that for all $m_1, m_2 \in \mathcal{M}_P$:*

- $T_{min}(m_1, m_2) \leq T(f(m_1), f(m_2)) \leq T_{max}(m_1, m_2)$
- $X(m_1) = X(m_2) \Leftrightarrow X(f(m_1)) = X(f(m_2))$ *where the function X can be the function dom or cod.*

If there is only one mapping f between an attack pattern P and a history log H, then it is easy to deduce which machine in the log history initiated the attack, and guarantee that no other machine could have been the initiator.

The identification now consists in finding all the mappings between an attack pattern and a history log, which is close to the subgraph isomorphism problem known to be a NP-Complete [14]. Thus, our problem is also NP-Complete.

Theorem 5. *The problem of identifying a pattern in an Timed MSC is NP-Complete.*

Proof. The problem is NP because a resulting mapping can be checked in polynomial time.

To prove completeness, we can reduce the k-clique problem to the mapping problem. Let $G = (V, A)$ be a graph where V is a set of vertices and $A \subseteq V \times V$ is a set of edges. Assuming that V has an order, we build a history log $H = (V, A, T)$ such that T enumerates each message of A in lexicographical order, i.e., a message (a, b) is before (c, d) if $a < c$ or if $a = c$ and $b < d$. To search for a k-clique, we will build an attack pattern $(V', A', Tmin, Tmax, PA)$ with k processes such that (V', A') is a k-clique graph and for all $m_1, m_2 \in A'$ such that $m_1 < m_2$, we have $T_{min} = 1$ and $T_{max} = +\infty$. □

Although the subgraph isomorphism problem is NP-Complete, when the subgraph is fixed, the problem becomes polynomial. Ullmann described for the first time a procedure to solve this problem in polynomial time when the subgraph is fixed [12]. The algorithm was then improved several times [3], [5], [13].

We have been inspired by these approaches to elaborate our algorithm of a mapping between an attack pattern and a history log.

VI. Approach

In this section, we propose an efficient mapping algorithm between an attack pattern and a history log and a theoretical analysis of its complexity.

We paid a particular attention to the data structure used in order to have an efficient implementation. Thus, the history log H is an array where the index of each cell corresponds to the time step of a message. Therefore, a cell $H[i]$ contains the messages occurred at the time step i.

Algorithm 1 identifies all the mappings between an attack pattern P and a history log H.

If some messages are already identified as malicious, the algorithm can take a partial function that maps these messages as an input to increase its efficiency.

Algorithm 1 Function match

Input: A log history $H = (\mathcal{P}_H, \mathcal{M}_H, T)$, a pattern $P = (\mathcal{P}_P, \mathcal{M}_P, T_{min}, T_{max})$, a map $Pmap : \mathcal{P}_H \to \mathcal{P}_P$, and a partial mapping function $f : \mathcal{M}_P \to \mathcal{M}_H$.
Output: All possible mapping functions between P and H.

1: $p := null$
2: $t_{min} := -\infty$
3: $t_{max} := +\infty$
4: **for all** $m \in \mathcal{M}_P$ not mapped in f **do**
5: $tmp_{min} := -\infty$
6: $tmp_{max} := +\infty$
7: **for all** $m' \in \mathcal{M}_P$ already mapped in f **do**
8: $tmp_{min} := max(tmp_{min}, T(f(m')) + T_{min}(m', m))$
9: $tmp_{max} := min(tmp_{max}, T(f(m')) + T_{max}(m', m))$
10: **end for**
11: **if** $tmp_{max} - tmp_{min} \leq t_{max} - t_{min}$ **then**
12: $t_{min} := tmp_{min}$
13: $t_{max} := tmp_{max}$
14: $p = m$
15: **end if**
16: **end for**
17: **if** $p = null$ **then**
18: **return** $\{f\}$
19: **end if**
20: **for all** $t := t_{min}...t_{max}$ **do**
21: **for all** $msg \in H[t]$ **do**
22: **if** $Pmap[dom(msg)]$ **then**
23: **if** $Pmap[dom(msg)] \neq dom(p)$ **then**
24: **continue**
25: **end if**
26: **end if**
27: **if** $Pmap[cod(msg)]$ **then**
28: **if** $Pmap[cod(msg)] \neq cod(p)$ **then**
29: **continue**
30: **end if**
31: **end if**
32: $result := result \cup match(H, P, Pmap \cup \{(dom(p, dom(msg)), (cod(p, cod(msg))\}, f \cup \{(p, msg)\})$
33: **end for**
34: **end for**

The algorithm works in two phases. In the first phase (lines 4 to 16), the algorithm selects a message in the attack pattern that minimizes the number of potential mapping in the log history .

When a message in the attack pattern is found, the second phase of the algorithm identifies a mapping between the message of the attack pattern and a message of the log history (lines 20 to 34). The two phases are executed recursively until all the pattern messages are processed. Finally, the algorithm returns a list of the occurrences of attacks that map the attack pattern.

Complexity analysis. The complexity of the first phase is $O(|\mathcal{M}_P|^2)$. The complexity of the second phase without the recursive call is $|H[t_{min}]| + |H[t_{min} + 1]| + \ldots + |H[t_{max}]|$. The latter is $O(avg\_rate \times max\_duration)$ in the worst case, where avg_rate is the average message rate in the network and $max\_duration$ is the maximum duration of the attack pattern. Thus, each recursion in the algorithm has a complexity of $O(|\mathcal{M}_P|^2 + avg\_debit \times max\_duration)$. The number of recursions is in the $O(|\mathcal{M}_H|^{|\mathcal{M}_P|})$ in the worst case. We notice that the proposed algorithm is exponential to the size of the attack pattern. In practice, this worst-case analysis for the number of recursive calls is not very useful, because the majority of messages are not compatible, so such a combinatorial explosion will not appear. In practice, we do not face a combinatory explosion since phase 1 of the algorithm minimizes the subset of messages in which the mapping is performed. Consequently, the log history does not impact the complexity of the algorithm.

VII. Benchmark

In order to evaluate the effectiveness of our approach and the scalability of the algorithm, we conducted an experimentation of attack pattern matching. We considered several network configurations depending on the given input data, which are the following:

- n: Number of machines in the network
- avg_rate: Average rate of the number of messages sent per second per machine
- t_total: The total time of the history log
- P: the attack pattern

Based on the input data, we randomly generate a history log H, in which we incorporate an attack that matches pattern P.

Each configuration is executed 100 times, then we report the average execution time. The implementation of Algorithm 1 was carried out in C++ and we performed the benchmarks in a VirtualBox with 8 GB of RAM and i5-7500 processor.

The first attack pattern corresponds to an attack through two proxies (Figure 3), in which A is the attacker, T is the target, and P_1 and P_2 are the proxies.

We assume that the two messages entering and leaving the target machine T were detected as malicious so we already map these two messages. The obtained results are presented in Table I.

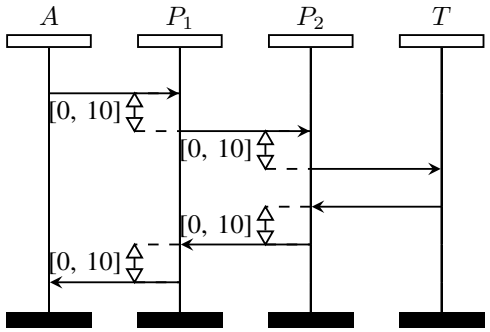

Fig. 3. Pattern of attack consisting in passing through one proxy.

In that table, the first column indicates the total number of messages in the history log. The columns n, avr_rate, t_total are the input data. The column $Times$ indicates the average of 100 execution times of the algorithm. The last column indicates the success rate of the attacker identification. Each row corresponds to the configuration of one experimentation.

Number of messages	n	avg_rate (msg/sec)	t_total (sec)	$Times$ (ms)	Successful Identification
6M	10K	10	60	13	99%
12M	10K	10	120	12	99%
18M	10K	10	180	12	100%
24M	10K	10	240	12	99%
6M	10K	10	60	13	99%
12M	10K	20	60	84	98%
18M	10K	30	60	260	89%
24M	10K	40	60	590	83%
6M	10K	10	60	13	99%
12M	20K	10	60	26	100%
18M	30K	10	60	41	99%
24M	40K	10	60	54	99%
600M	100K	100	60	93,000	91%

TABLE I
BENCHMARK FOR THE ATTACK PATTERN IN FIGURE 3.

We have several observations. The first one is that the execution time does not depend on either the length of the log history or the number of messages, but rather on the average rate of message exchange. The second observation is that the execution time grows linearly according to the number of machines in the network.

Considering the average rate, the execution time follows a polynomial function of order two. Finally, as a last observation, for a network of 100,000 machines, sending an average of 100 messages, the execution of the algorithm is about one minute.

We also notice that when the average rate increases while the attack pattern is simple, the success of identification decreases. However, the algorithm returns a subset of machines that might be the attack initiators.

We performed a second benchmark considering a more complex attack pattern (Figure 4). In this pattern, the attacker starts by sending a malicious message to cause damage. This first attack is carried out through two proxies, P_1 and P_2. Then, the attacker uses the newly created vulnerability to obtain critical data via the proxies P_1 and P_3.

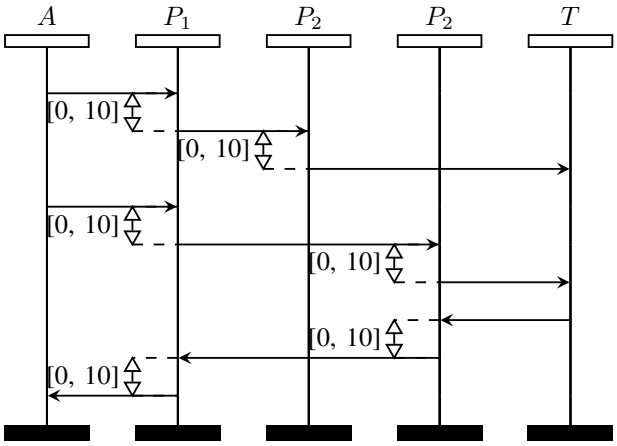

Fig. 4. Pattern of attack consisting in passing through one proxy.

The results obtained are presented in Table II. We notice that the resolution time is almost identical to the results in Table I. However, the successful identification rate is much better. Indeed, the more complex and precise the attack pattern is, the fewer chances there are to wrongly identify the pattern.

Number of messages	n	avg_rate (msg/sec)	t_total (sec)	Times (ms)	Successful Identification
6M	10K	10	60	12	100%
12M	10K	10	120	12	100%
18M	10K	10	180	12	100%
24M	10K	10	240	12	100%
6M	10K	10	60	12	100%
12M	10K	20	60	84	100%
18M	10K	30	60	290	100%
24M	10K	40	60	640	100%
6M	10K	10	60	12	100%
12M	20K	10	60	26	100%
18M	30K	10	60	42	100%
24M	40K	10	60	56	100%
600M	100K	100	60	97,000	100%

TABLE II
BENCHMARK FOR THE ATTACK PATTERN IN FIGURE 4.

VIII. CONCLUSION

When a cyber-attack occurs in a network, the problem of automatically assigning this attack without knowledge of the used attack pattern is undecidable even if we assume that the overall network traffic is recorded.

In this paper, we presented a specification of attack pattern, which is a formal representation of a scenario used by the attacker to execute an attack. We demonstrated that by knowing the attack pattern used by the attacker, it becomes possible to identify the initiator of an attack. We also showed that the automated detection of attack patterns is realistic using machine learning techniques and artificial intelligence.

In order to show the feasibility of this approach, we implemented our algorithm and performed benchmarks on two attack patterns. The results showed that our approach allowed us to identify the initiator of an attack with a high success rate.

The analysis of the complexity of our approach has shown us two interesting things. First, the time used by our algorithm does not depend on the duration of the communication recordings. Secondly, the resolution time seems to be polynomial in practice. For example, the proposed algorithm was able to assign an attack in a network of a hundred thousand machines where each machine sends 100 message/sec in less than a hundred seconds.

Three possible evolutions of our approach can be considered as future work:

- Since some routers may not be cooperative and may not disclose or record communications between machines, one possible evolution is to add the ability to process missing messages by proposing the best partial mapping.
- Be able to handle more complex attack patterns using HMSC (Highlevel Message Sequence Charts). Indeed, this formalism is used to describe a set of interaction scenarios between remote sites. In addition, the use of blocks, such as alternative and parallel blocks, could be used to support the modelling of complex attack patterns.
- The attack scenarios may not be deterministic. The use of probability in modelling could be used to manage this case of non-determinism.

ACKNOWLEDGMENT

This work has been partly funded by Defence Canada grant under the program Innovation for Defence Excellence and Security (IDEaS), and by the Ministère de l'Économie et de l'Innovation - Québec.

REFERENCES

[1] Steven Michael Bellovin, Marcus Leech, and Tom Taylor. Icmp traceback messages. 2003.

[2] W Earl Boebert. A survey of challenges in attribution. In *Proceedings of a workshop on Deterring CyberAttacks*, pages 41–54, 2010.

[3] Vincenzo Bonnici, Rosalba Giugno, Alfredo Pulvirenti, Dennis Shasha, and Alfredo Ferro. A subgraph isomorphism algorithm and its application to biochemical data. *BMC bioinformatics*, 14(7):S13, 2013.

[4] JL Camp. Digital identity. *IEEE Technology and society Magazine*, 23(3):34–41, 2004.

[5] Luigi P Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. A (sub) graph isomorphism algorithm for matching large graphs. *IEEE transactions on pattern analysis and machine intelligence*, 26(10):1367–1372, 2004.

[6] Jeffrey Hunker, Bob Hutchinson, and Jonathan Margulies. Role and challenges for sufficient cyber-attack attribution. *Institute for Information Infrastructure Protection*, pages 5–10, 2008.

[7] Vikas Jayaswal, William Yurcik, and David Doss. Internet hack back: Counter attacks as self-defense or vigilantism? In *IEEE 2002 International Symposium on Technology and Society (ISTAS'02). Social Implications of Information and Communication Technology. Proceedings (Cat. No. 02CH37293)*, pages 380–386. IEEE, 2002.

[8] Jamal Raiyn et al. A survey of cyber attack detection strategies. *International Journal of Security and Its Applications*, 8(1):247–256, 2014.

[9] Paulo Shakarian, Gerardo I Simari, Geoffrey Moores, and Simon Parsons. Cyber attribution: An argumentation-based approach. In *Cyber Warfare*, pages 151–171. Springer, 2015.

[10] Paulo Shakarian, Gerardo I Simari, Geoffrey Moores, Simon Parsons, and Marcelo A Falappa. An argumentation-based framework to address the attribution problem in cyber-warfare. *arXiv preprint arXiv:1404.6699*, 2014.

[11] Alex C Snoeren, Craig Partridge, Luis A Sanchez, Christine E Jones, Fabrice Tchakountio, Stephen T Kent, and W Timothy Strayer. Hash-based ip traceback. In *ACM SIGCOMM Computer Communication Review*, volume 31, pages 3–14. ACM, 2001.

[12] Julian R Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM (JACM)*, 23(1):31–42, 1976.

[13] Julian R Ullmann. Bit-vector algorithms for binary constraint satisfaction and subgraph isomorphism. *Journal of Experimental Algorithmics (JEA)*, 15:1–6, 2010.

[14] Ingo Wegener. *Complexity theory: exploring the limits of efficient algorithms*. Springer Science & Business Media, 2005.

[15] David A Wheeler and Gregory N Larsen. Techniques for cyber attack attribution. Technical report, INSTITUTE FOR DEFENSE ANALYSES ALEXANDRIA VA, 2003.