Trust-based collaborative defences in multi network alliances

Ralph Koning
System and Network Engineering Lab
University of Amsterdam
Amsterdam, The Netherlands
r.koning@uva.nl

Ameneh Deljoo
System and Network Engineering Lab
University of Amsterdam
Amsterdam, The Netherlands
a.deljoo@uva.nl

Lydia Meijer
TNO
Groningen, The Netherlands
l.l.meijer@tno.nl

Cees de Laat
System and Network Engineering Lab
University of Amsterdam
Amsterdam, The Netherlands
delaat@uva.nl

Paola Grosso
System and Network Engineering Lab
University of Amsterdam
Amsterdam, The Netherlands
p.grosso@uva.nl

***Abstract*—Collaborative defences against, for example, Distributed Denial of Service attacks require trust between the collaborators. Even in a trusted alliance of collaborators, some members are more trusted than others with the execution of a specific task. This paper shows that trust can be used as a criterion in collaborative defences against attacks on computer infrastructures. We evaluate the implementation of a trust based defence in an overlay network and compare its' performance to other collaborative defences in the same environment. Results show that the trust-based defence has comparable results to the other collaborative defences after the trust values converge to reflect the behaviour of the nodes.**

***Index Terms*—Security, Trust, Multi-Domain Defence, Collaborative Defence, Computer Networks, Risk**

I. INTRODUCTION

Distributed attacks on computer infrastructures, such as Distributed Denial of Service (DDoS) attacks, disrupt widely used internet services on a regular basis [1]. Since the victim is attacked by a multitude of sources, the attack is often hard to defend because the attack-power already accumulated before reaching resources under control of the victim. Collaboration with upstream providers and transit networks can increase defence changes because they usually have more capacity and resources available to deal with such attack-power. Collaborations can be organised using alliances, in which members agree to a common set of rules to establish a base of trust [2]. Yet, even with an alliance in place, based on the ability, the willingness to help, and experience, some members will be trusted more than others. This trust differs per member and is based on the members' experience and its collected evidence about the member in question.

In [3], Koning et al. show that the efficiency of collaborative defences, even when applying the same task, can differ based on the approach and order in which the collaborators are asked for help. Since asking help from more trusted members may yield better results in defending, we ask ourselves: *If we have an indicator of trust, can this be used in ally selection and how does trust affect the efficiency of collaborative defences?*

978-1-7281-3949-4/19/$31.00 ©2019 IEEE

The objective of this research is to implement and evaluate a collaborative defence strategy that uses the Social Computational Trust Model (SCTM) [4]. First, we describe the background and the context of this work in Sec. II. Then, we briefly introduce the SCTM model and the parts of the model that are used in this work in Sec. III. We continue by explaining the implementation of the trust based defence and its prerequisites in Sections IV and V. In Sec. VI we explain how we evaluate the trust based defence and in Sec. VII we compare the results of the trust-based defence approach to the results of the defence approaches defined in [3]. Finally, we discuss the outcomes of this work in Sec. VIII, compare our work to other research in Sec. IX and conclude in Sec. X.

II. MULTI-DOMAIN SARNET

SARNET is a framework for detection and autonomous mitigation of attacks on computer infrastructures. When multiple defences are available for the situation the SARNET picks the *defence* with the highest efficiency ranking and executes it. A defence consists of multiple *tasks*. These tasks can be performed by the domain itself, or be delegated to collaborators. Collaborative defences require each participating domain to run its own SARNET agent. The agents are responsible for coordinating activities between collaborators in the alliance. These activities include coordinated response to threats as well as sharing information such as threat intelligence or analytic data with agents of other domains.

In multi domain defences we distinguish four categories of requests.

- Simple informational requests: Requests that can be answered directly based on the knowledge of the requested member.
- Complex informational requests: 1) Request for information that is provided over time at the discretion of the

requested member i.e. subscribing to an information service 2) Information that is collected (from other sources or members) and transformed by the requested member.
- Simple actionable requests: The decisions and actions are driven by the requester. Simple actionable requests are are executed directly by the requested member according to the requester specification.
- Complex actionable requests: The decisions and actions are driven by the requested party. 1) Automatic mitigation services allow the requested member to apply certain countermeasures at their discretion at some point in the future. 2) Delegated actions allow the requested member to further handle the orchestration the multi-domain defence.

Both types of simple requests consist of actionable tasks or queries and the performance can easily be verified. For an actionable task one can measure if the desired effect took place and the informational query is shortly followed up with a response. Complex requests often consist of multiple subtasks, or queries to other members. These tasks usually take longer to complete or give multiple replies over an extended time period. The quality of the responses depends on the amount of effort the requested domain is willing to spend on the analysis, execution, or processing. Therefore, it is important that the remote domain acts in the interest of the requester.

We implement SARNET on top of VNET which provides an experimental platform for secure networking research [3, 5].

A. attack scenario

Using VNET we can construct a virtual infrastructure by supplying a topology, (Fig. 1), that uses virtual machines which represent the following domains:

- a *service domain* (V) contains a web service that resembles a market place where clients make purchases;
- a *transit domain* (51–58) forwards traffic, it provides basic blocking, redirection and rate limiting functions;
- a *client domain* (12–18) interacts with the service domain by making transactions with the service domain;
- a *NFV domain* (61) provides a set of network functions (using Network Function Virtualisation, NFV) that can be used for further analysis such as an Intrusion Detection System or a honeypot or can be used as a countermeasure such as a traffic scrubbing NFV.

The services within the domains are provided using containers connected using a software switch.

Traffic flows back and forth from the client domains via the transit domains between the service domain. Normally, the traffic consists of transactions (simulated purchases) sent to the service domain. When we start an attack we instruct one or more client domains to attack the victim which is the service domain. The attacks consist of UDP based (Distributed) Denial of Service attacks initiated from one or more client domains (12–18) that congests the link between the victim (V) and its upstream provider transit domain (51). The defence consists of filtering out this traffic pattern at the members of the alliance. The alliance consists of all the transit domains, the service domain and the NFV domain and excludes the client domains.

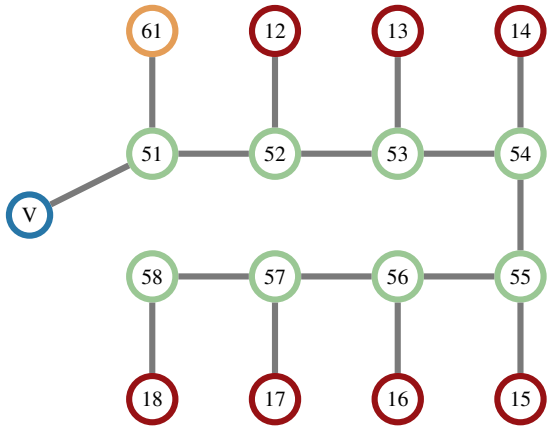

Figure 1: The network topology used in the experiments, V is the victim, 12–18 are attackers, 51–58 map to transit51-58 and are transit domains, 61 is nfv61 the NFV domain.

III. Trust in security alliances

Trust is considered as one of the success factors for alliances [6]. To form a security alliance, trust among the members needs to be organised, maintained and measured. Deljoo et al. [4] define trust in the alliance as follows: “a trustor expects a trustee to perform task t and the trustee will not exploit vulnerabilities of the trustor when the trustee is faced with the opportunity to do so." Therefore, every trustee:

- Has an ability to perform task t (competence),
- Fulfills the commitments towards the trustor (integrity), and
- Acts towards the trustor interest (benevolence).

In order to organise, maintain, and measure trust among the members, we propose to use the social computational trust model (SCTM) developed by [7]. The SCTM model evaluates the trust of a trustee based on the three distinctive factors, benevolence, competence and integrity; this work assumes that integrity is provided, automatically and fully, by being member of the alliance. Therefore, this work does not include the computation of integrity.

A. Notation

Let us denote alliances members as A where $l, r \in A$. The interaction history is stored in the evidence knowledge base E_{Kb_l} and is called evidence. We assign three different values to the evidence using function $val(E)$: 1, when *FD* (fulfilled), .5 when *FDD* (fulfilled with delay), and 0, when *V* (not fulfilled). The E_d function obtains direct evidence from the E_{Kb_l}. The $E_{in(t)}$ function extracts indirect evidence from all $E_{Kb_{r_{nbr}}}$ for task t where nbr are the neighbours of r. In this paper we initialise the knowledge base, for the topology in Fig. 1, using the values in Table III.

B. Benevolence

Benevolence is considered as one of the key trust components and the antecedent of trustworthiness (e.g. [8, 9]). The benevolence value is derived from the mutual interaction between a trustor and a trustee. According to [7] the benevolence of trustee r towards trustor l is computed by:

$$Ben(l,r) = \frac{1}{|N|} \sum_{l,r \in A} (val(E_d(l,r; E_{Kb_l}))), \quad (1)$$

where N is the number of entries in E_{Kb_l} with the defined value, in which l has interacted with r.

C. Competence

Competence refers to the ability of the trustee to perform the task. We assume that l and r have not collaborated before; therefore, the trustor needs to request the evidence form the trustees' direct neighbours (r_{nbr}). The competence of node r as the trustee is given by:

$$Comp(l,r,t) = \frac{1}{|r_{nbr}|} \sum_{l,r \in A} (val(E_{in(t)}(r_{nbr}, r; E_{Kb_{r_{nbr}}}))), \quad (2)$$

where $|r_{nbr}|$ is the number of neighbours who respond to the request of r. The benevolence and competence function return a value between [0,1].

D. Risk

Every collaboration comes with a risk that needs to be minimised. Das et al, [6] defined the relational and performance risks as two distinct risks for the alliance, and Deljoo et al. [4] present the risk evaluation framework for the security alliance. Relational risk concerns the behaviour of the alliance members, $1 - Ben(l,r)$, while the performance risk considers the competence of members to perform the given task, $1 - Comp(l,r,t)$. Performance risk also guarantees the service delivery through the collaboration. The risk is calculated using an updated version of the equation defined in [4], Section VI:

$$Ri(l,r,t) = \alpha(1 - Ben(l,r) + (1-\alpha)(1 - Comp(l,r,t)) \quad (3)$$

In Sec. II we mention that for complex messages it is important that the requested domain acts in the interest of the requester i.e. have a low relational risk. When the relational risk is considered more important α (a value between $[0,1]$) can be set to be higher than the default value of .5.

IV. Trust based defence prerequisites

To implement trust based defences we need to extend the existing SARNET agents. Since the trust is based on the time it takes for an ally to perform an action we need a message tracking mechanism. In Sec. IV-A we explain how we use this tracking mechanism to gather the evidence required for the trust computation. In Sec. IV-B we show how we compute the trust values that are used in the trust based defence algorithm in Sec. V.

A. Message tracker

Since the building of evidence is based on fulfilling requests within a time period, we need to implement a method of time tracking the messages. VNET uses an asynchronous programming model; sending and responding is handled using different code and the response handler has no information on what is being sent. For time tracking to work, we need to map the response back to the request. Therefore, we modified the original message format to include a message ID and a transaction ID which we also store upon sending. The recipient can now reply with the message ID included such that the sender can look up the corresponding request its *track database*. Table I shows the format of entries in the *track database*.

To each message sent we expect two replies: 1) an acknowledgement that the message is received, and 2) a message that the task is completed, each reply includes the message ID. When data is requested from the recipient, the recipient returns the data in a third message that includes the transaction ID.

When the acknowledgement or the task completed messages are received by the sender the sender will set *t_repl* in case of an acknowledgement, and *t_done* when the task is completed.

Periodically[1], we prune the *track database*. We convert the completed and expired transactions in *track database* to evidence and store them in the evidence knowledge base (evdb). When *t_done* is set for this request we store the difference between *t_req* and *t_done* in the evdb. When *t_done* is not set and the difference between *t_req* and the current time exceeds the timeout, the request did not complete in time and we store 0 in the evdb.

The *track database* only contains messages that are 'in flight', when no messages are exchanged and the timeout has expired the database should be empty.

msg_id	msg	type	local	remote	t_req	t_repl	t_done

Table I: Track database entry format: we store the id of the message, the complete message, the message type, local and remote agent identifiers, the time when the request is made request, and the times on which we receive the acknowledgement (t_repl), and task done notifications (t_done).

B. Trust computation

When computing trust, risk or any of their components we consult the evdb. The format is described in Table II. We store the task itself, the trustor (source), the trustee(destination), the time that has elapsed to complete the request, the request id and the time when the message is recorded. When the trust values are requested we use the logic in Algorithm 1 from [4] to convert the elapsed times to evidence.

Benevolence is based on all the fulfilled evidence in the evdb. To compute benevolence, we requires the name of the remote agent. When provided with the optional argument time

[1]The function should be called approximately every second but the timing can vary due to other periodic functions being executed in the same thread.

benevolence only obtains the evidence from the evdb that is produced after that time. When the evidence is obtained we calculate benevolence according to Eq. (1). If the benevolence computation fails the benevolence implementation returns 'None'

Competence depends on acquired data from all the neighbours of the target. Computing competence requires the task and the target and, like benevolence, the function returns 'None' when the value cannot be computed. To acquire the data from member domains, competence relies on the following (simple informational) requests:

- Trust Request: Requests information for some trust component. For *competence* the request must also contain the task in question.
- Trust Response: Responds to the trust request by providing the requested part of the evdb.

When the relevant evidence is acquired, we compute the competence according to Eq. (2). Acquiring competence is an expensive operation, the requester has to block until all the neighbours of the target reply with their evidence. A neighbour can take some time to reply because the request has to be sent out on the network, a relatively slow medium, and is subject to delays, and round trip times. Since competence can be called for multiple times per second and the reply is not instant, the long running requests will be sent out multiple times per second, consuming resources and congesting the network. To prevent congestion from happening, we cache the result from competence for 5 seconds. Every subsequent request is now answered from the cache until the cache is cleared in which case the request goes on the network to get updated values. Caching does introduce a trade-off between accurate data and performance but it is necessary to prevent excessive resource use. For the caching to be effective, we recommended to keep the time interval to clear the cache, which is in our case 5 seconds, larger than the response time (in our case < 1 second).

Risk is provided by taking the benevolence and accumulated competence values and computing them according to Eq. (3). If the value cannot be computed, the function returns 'None'. Since the tasks that are executed only consists of the simple request types (See Sec. II) we do not prioritise relational risk over performance risk. Therefore, we calculate risk using: $\alpha = .5$.

task	source	destination	elapsed	id	time

Table II: Evidence knowledge base entry format: A single entry stores the task (in our case message type), the source and destination agent, the elapsed time of the request (0 if the task never completed), the id of the message and the time that the evidence is gathered.

V. Risk based defence implementation

Algorithm 1 shows how we implemented the risk based approach. The algorithm has to distinguishable phases: 1) Lines 1–6. When the defence starts we first collect the necessary values, risk, benevolence, and random to rank the nodes. Once these values are collected, they will not be updated as long as the defence is active. For defending, we are interested in the nodes that have the lowest risk and, when the risks are equal, the node with the highest benevolence. Therefore, we sort the nodes firstly on their risk, and secondly on their benevolence. To break ties when two nodes have equal risk and benevolence, we assign a third ranking value to a random and unique number. In our experiments we seed the random number generator in order to achieve consistent results. 2) Lines 7–13. When the trust information is gathered and the ranking is complete, the actual defending starts. The top ranked domain is asked whether it detects the traffic and to deploy a defence when this is the case. To each queried member it will exchange the following messages:

Algorithm 1: Risk based algorithm

Input: $pattern$: attack pattern
$alliance$: alliance members
$time$: wait time

1 **for** $n \in alliance$ **do**
2 $Risk_n \leftarrow$ gather_risks(task, n);
3 $Benevolence_n \leftarrow$ gather_benevolence(n);
4 $Random_n \leftarrow$ random();
5 **end**
6 $ranked \leftarrow$ sort N on $Risk$, $1 - Benevolence$, and $Random$;
7 **for** $node \in ranked$ **do**
8 **if** *attack not resolved* **then**
9 *ask* $node$ for its $neighbours$ that produce $pattern$;
10 *deploy* countermeasure at $node$;
11 wait for $time$ seconds;
12 **end**
13 **end**

- **Ask**: Asks a member whether it has seen traffic according to a certain pattern and from which of its neighbours the traffic originates.
- **Match**: The response to the ask message containing the responding domain's neighbours on which the pattern is seen.
- **Deploy**: Deploys a filtering action on the specified traffic pattern towards one of a members neighbouring domains.

Ask is considered a *simple informational request* and *deploy* a *simple actionable request*

VI. Evaluation

First, we need to verify whether the trust collection and the algorithm behave as expected. To achieve this, we attack the victim using same attack multiple times and between each time we make sure that the network returns to a normal state. After receiving the attack and implementing defensive actions, the victim should have more evidence available and the trust numbers should allow the victim be able to make a better

decision. To validate, we first we set the initial evidence such that it mismatches the behaviour of the nodes; this mismatch should immediately result in unfulfilled tasks in the evidence database and in turn change the ranking for the next defence. In Table III we list how we initialise the network. We give a preference to transit51 by initialising with two extra fulfilled (FD) transactions. We demote the use of nfv61 by adding two extra violated (V) transactions.

node	FD	FDD	V
transit51	3	1	1
transit52	1	1	1
transit53	1	1	1
transit54	1	1	1
transit55	1	1	1
transit56	1	1	1
transit57	1	1	1
transit58	1	1	1
nfv61	1	1	3

Table III: Initial evidence in the evdb: amount of fulfilled (FD), fulfilled with delay (FDD) and violated (V) of the members participating in the alliance.

We set members to behave differently from the initialisation (see Table IV). Transit51 who should be initialised as more trustworthy is given a low success rate, which should result in to violations when being asked. We also vary the success rate of the others.

This means we expect two important things to happen: 1) In the first attempt to defend transit51 is considered very trustworthy and gets selected. During that attempt it will generate many violations (V). Therefore we expect that transit51 should move down to the bottom. 2) transit52 has a high success rate, resulting in fulfilled messages (FD) eventually, because it has the highest success rate it should end on top. The other nodes are expected to converge to the behaviour respective to their number.

node	success rate
transit51	0.0000001
transit52	1.0
transit53	0.8
transit54	0.6
transit55	0.4
transit56	0.2
transit57	0.1
transit58	0.0000001
nfv61	1.0

Table IV: Member behaviour for our experiments, lists the probability of successfully executing a task for all the members in the alliance.

We will now execute the attack 10 times and display how the ranking changes: Second, we need to see the effect of the ranking on the efficiency.

Finally, we compare the efficiency of this approach to the efficiency of three original approaches we evaluated before in [3]:

- Approach 1 - Counteract Everywhere; places a defence on every ally that sees attack traffic starting near the victim and working its way to the attackers.
- Approach 2 - Minimise Countermeasures; first discovers where the attackers are located by recursively asking the allies from where it originates and defend it close to the attackers at the border.
- Approach 3 - Minimise Propagation; similar to Approach 1, this approach places a defence starting at the ally closest to the victim. Only this time we wait for a time period to notice the effect. When still under attack it continues.

VII. RESULTS

The topology used for the experiments is listed in Fig. 1. We use the same four scenarios as in [3] to compare the results; each scenario portrays distinct attack conditions:

- *single attacker near*, with the attacker in position 12;
- *single attacker far*, with the attacker in position 18;
- *two attackers (1 far, 1 close), one far and one close*, respectively at 18 and 12;
- *all clients attacking*, all clients attack;

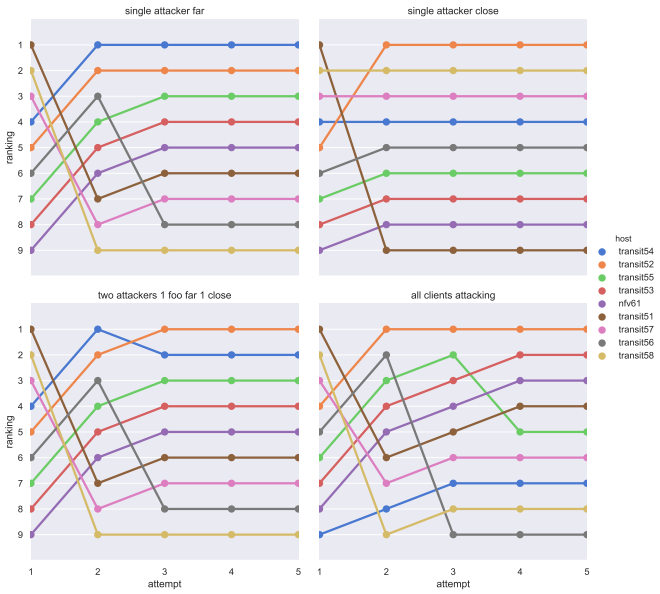

Figure 2: Changes of node ranking after defending x times, 1 is the highest rank and gets selected as member to defend.

Fig. 2 shows how the ranking of each node evolves over time while learning the member behaviour. The rankings are snapshots of a single run; they are not averaged. Each line represents a member; we take snapshots of the ranking at the moment the defence is created. As expected, the first attempt ranks the members according to the initialisation. For the subsequent attempts the ranking changes until it converged to the set member behaviour and stabilises as expected.

Fig. 3 shows the efficiency [10] over runs. Each data point is an average over three runs, the error bars contain the standard

deviation from the mean. There is not much variation between the runs, so the error bars in the plots are small and mostly unnoticeable. It is clearly visible that on the first run, when transit51 is ranked high (see Table III, efficiency is lower than in the consecutive runs. Also in all cases the efficiency seems to increase and stabilise over time. The variations after the stabilisation can be explained because we are working with probabilities driven by random numbers that change over time.

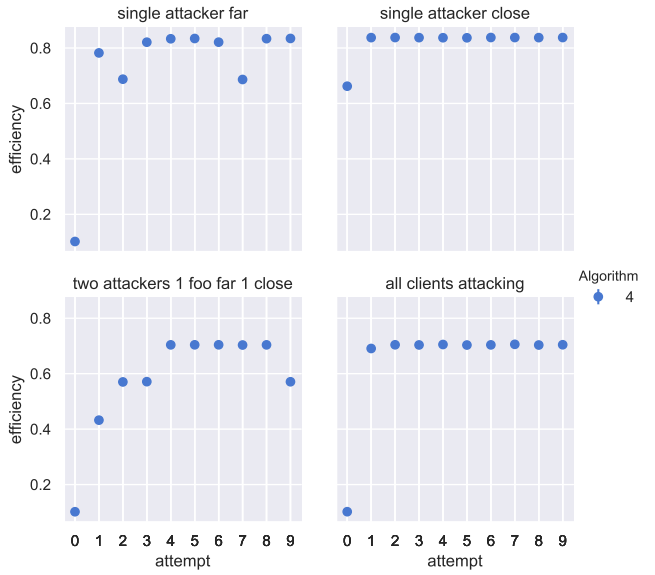

Figure 3: Changes in efficiency after defending x times

Fig. 4 shows the trust-based approach we discussed in this paper in comparison to the approaches we studied in [3]. On the x axis *attempt* indicates the defence attempt using the same evdb. The different colours or symbols indicate the different approaches. The trust based approach is indicated as *Approach 4*.

In Figs. 4 and 5 we compare the approaches under two different budget constraints: high (unlimited) budget, and low budget. With a budget of 900, the victim can ask all the members of the alliance to defend: 9 members each member charging 100 credits. With a budget of 300 (Fig. 5) the victim can, therefore, only ask 3 members each defence.

When the budget is large enough to ask all the members (Fig. 4), the ranking can converge well. For the first attempt trust based approach is less efficient than the other approaches, but from the second attempt the graphs show no considerable efficiency difference from the top approaches, in the single attacker far and the all clients attacking approach the trust based approach is more efficient than the others.

In the constrained budget approach (Fig. 5) we can see that the trust based approach (approach 4) is not performing optimally. Still, in the single attacker far case, it performs better than the other algorithms. For the other cases, we can see that approach 4 converges to higher efficiency yet slower than in (Fig. 4). The slower convergence can be explained by the fact that the algorithm can only ask the top 3 nodes in the list and thus accumulates less trust data.

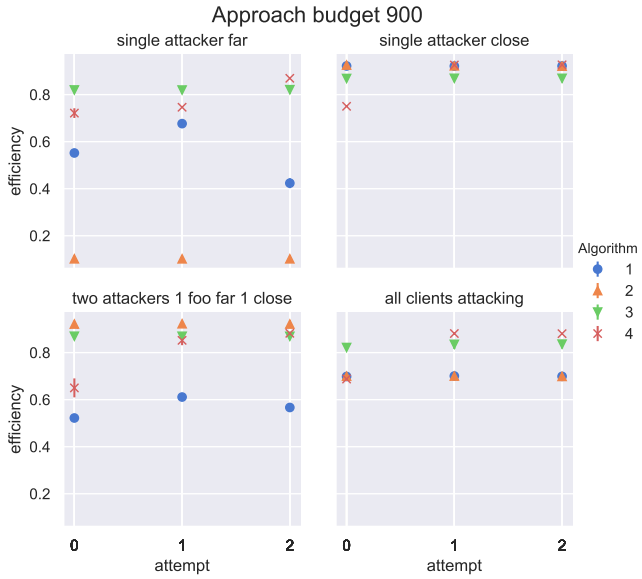

Figure 4: Comparison of trust based approach (4) to approaches from [3] with unlimited budget

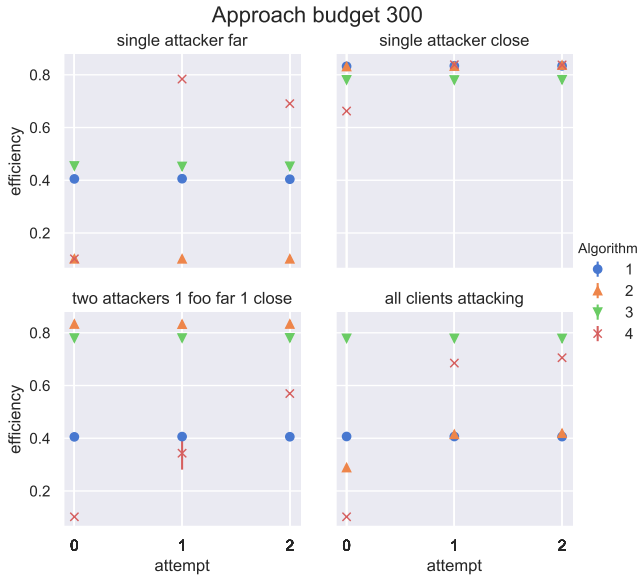

Figure 5: Comparison of trust based approach (4) to approaches from [3] with very constrained budget

VIII. Discussion

When the defence budget is constrained, we noticed some side-effects using the trust based approach. Due to the constrained budget, we can only ask a limited amount of members during each defence attempt. Because we rank according to trust, these are only the highest ranked members. When a lower ranked member changes behaviour and suddenly

becomes benevolent and competent, it will never move up the ranking because of the lack of interaction and evidence. Only when highly ranked members move down the ranking, evidence of some of the lower ranked members are collected; this could cause a low ranked and (suddenly) trustworthy member to move up. This side effect can be seen in Sec. VII the constrained budget approach (Fig. 5) where Approach 4 converges slowly and never reaches the efficiencies according to (Fig. 4).

A prerequisite for the trust-based defence, the message tracking implementation, adds significant overhead to the system. While originally receiving zero replies, with the message tracking implementation enabled, a single message can now receive up to three replies: the acknowledgement, the task done message, and the actual reply. When under heavy attack, these messages add to the load of the system, probably worsening the congestion problem. Removing the acknowledgement messages, which are currently not used by the trust based algorithm, and merging the task done message into the actual reply could help to reduce this overhead.

Our trust based approach currently asks the most trusted members in the alliance for help, regardless of whether it is transporting attack traffic. This can be improved using heuristics; we can establish an attack tree first by asking neighbours from which of their neighbours the attack originates, recursively. With the members that forward attack traffic identified, we can rank those on trust like we do now for the whole alliance. With the initial (time) penalty of learning the attack tree, we can be more targeted at asking members that can actually help in resolving the attack.

A trust based approach can also introduce new attack vectors. Since the victim always turns first to their most trusted allies, the amount of communication between them may be higher than members who are trusted less. Even when this communication is encrypted, by observing network communication patterns, a smart attacker may be able to identify the victims most trusted allies. She can use this to gain an advantage by somehow neutralising the victims allies first. If agents communicate over public infrastructure this information leakage is a potential risk. Establishing and using an encrypted p2p overlay network for agent communication can obfuscate these communication patterns since communication can go via any of the members.

IX. Related Work

Shabut et. al. describe a recommendation based trust model with an effective defence scheme for mobile ad hoc networks [11]. Their focus is mainly on preventing bad or malicious recommendations from nodes by providing defences against attacks on the trust model. The solution they propose relies on three centralised components, on which they perform statistical analysis and a clustering technique to detect deviations in trust and to prevent malicious recommendations. Our paper focuses on how trust can be used to improve defences on the network infrastructure. We recognise there are also attacks possible on the trust infrastructure, however this is beyond the scope of this paper.

Chen et. al [12] discuss a collaborative multi domain detection system for DDoS attacks. The authors developed a secure infrastructure protocol(SIP) to establish mutual trust or consensus. In SIP they adopt the adaptive trust negotiation and access control framework [13] which uses the PeerTrust [14] trust management system to establish trust. The trust used in these systems established by matching credentials of the requesting party to policies of the responding party. Our work uses a different form of establishing trust, the trust in this paper is based the behaviour of the participating node and their attitude towards benevolence, competence and, integrity.

X. Conclusion and Future Work

Using trust as a criterion for partner selection in multi-domain collaborative defences seems promising. Maintaining a short list of members who are responsive, capable and willing to help can be a beneficial when defending collaboratively. The results show indeed that the trust based algorithm increases efficiency in consecutive defences as more evidence is gathered from the members in the alliance. In cases where the budget is limited, and the collected evidence comes only from part of the alliance, the efficiency converges slowly and remains lower than when using the original approaches. At full budget, the trust-based approach converges faster to comparable, and in some case even higher, efficiencies than the original approaches. In general, we can conclude that there are benefits when considering the members' trust in selecting candidates for task placement in multi-domain collaborative defences.

We propose to increase efficiency further by: first, establishing an attack tree and, consecutively, applying the trust based response on the nodes in the tree. More research is needed to see whether the impact of establishing the attack tree outweighs the benefits when asking for help. This paper assumes that integrity is considered a condition for joining the alliance, all members of the alliance are considered to have the same integrity. In the future we intend to include integrity in the risk calculation. We also pointed out that unintended information leakage can occur when depending on the most trusted allies. Researching if and under what conditions trust based communication patterns leak trust information and how to mitigate this leakage will be necessary continuation of this work.

Acknowledgements

SARNET is funded by the Dutch Science Foundation NWO (grant no: CYBSEC.14.003 / 618.001.016) and the National project COMMIT (WP20.11). We would like to thank Ben de Graaff, for his contributions to this project. We would also like to thank our other research partners TNO, KLM and Ciena.

References

[1] O. Kupreev, E. Badovskaya, and A. Gutnikov. Ddos attacks in q1 2019. [Online]. Available: https://securelist.com/ddos-report-q1-2019/

[2] A. Deljoo, L. Gommans, C. de Laat, T. van Engers *et al.*, "The service provider group framework," *Looking Beyond the Internet: Workshop on Software-defined Infrastructure and Soft ware-defined Exchanges*, 2016.

[3] R. Koning, G. Polevoy, L. Meijer, C. de Laat, and P. Grosso, "Approaches for collaborative security defences in multi network environments," in *2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*. IEEE, 2019.

[4] A. Deljoo, T. van Engers, R. Koning, L. Gommans, and d. Cees, "Towards trustworthy information sharing by creating cyber security alliances," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 8 2018, pp. 1506–1510.

[5] R. Koning, B. de Graaff, G. Polevoy, R. Meijer, C. de Laat, and P. Grosso, "Measuring the efficiency of sdn mitigations against attacks on computer infrastructures," *Innovating the Network for Data Intensive Science (INDIS) workshop at Super Computing 2017, Denver (CO)*, 2017.

[6] T. K. Das and B.-S. Teng, "Trust, control, and risk in strategic alliances: An integrated framework," *Organization studies*, vol. 22, no. 2, pp. 251–283, 2001.

[7] A. Deljoo, T. van Engers, L. Gommans, and C. de Laat, "Social Computational Trust Model (SCTM): A Framework to Facilitate the Selection of Partners," in *2018 IEEE/ACM Innovating the Network for Data-Intensive Science (INDIS)*, 2018.

[8] D. Z. Levin, R. Cross, L. C. Abrams, and E. L. Lesser, "Trust and knowledge sharing: A critical combination," *IBM Institute for Knowledge-Based Organizations*, vol. 19, 2002.

[9] T. R. Koscik and D. Tranel, "The human amygdala is necessary for developing and expressing normal interpersonal trust," *Neuropsychologia*, vol. 49, no. 4, pp. 602–611, 2011.

[10] G. Polevoy, "Defence efficiency," *arXiv preprint arXiv:1904.07141*, 2019.

[11] A. M. Shabut, K. P. Dahal, S. K. Bista, and I. U. Awan, "Recommendation based trust model with an effective defence scheme for manets," *IEEE Transactions on mobile computing*, vol. 14, no. 10, pp. 2101–2115, 2014.

[12] Y. Chen, K. Hwang, and W. Ku., "Collaborative detection of ddos attacks over multiple network domains," *IEEE Transactions on Parallel & Distributed Systems*, vol. 18, pp. 1649–1662, 06 2007. [Online]. Available: doi.ieeecomputersociety.org/10.1109/TPDS.2007.1111

[13] T. Ryutov, Li Zhou, C. Neuman, N. Foukia, T. Leithead, and K. E. Seamons, "Adaptive trust negotiation and access control for grids," in *The 6th IEEE/ACM International Workshop on Grid Computing, 2005.*, Nov 2005, pp. 8 pp.–.

[14] W. Nejdl, D. Olmedilla, and M. Winslett, "Peertrust: Automated trust negotiation for peers on the semantic web," in *Secure Data Management*, W. Jonker and M. Petković, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 118–132.