A Deception Planning Framework for Cyber Defense

Jafar Haadi Jafarian
University of Colorado Denver
haadi.jafarian@ucdenver.edu

Amirreza Niakanlahiji
University of Illinois Springfield
aniak2@uis.edu

Abstract

The role and significance of deception systems such as honeypots for slowing down attacks and collecting their signatures are well-known. However, the focus has primarily been on developing individual deception systems, and very few works have focused on developing strategies for a synergistic and strategic combination of these systems to achieve more ambitious deception goals. The objective of this paper is to lay a scientific foundation for cyber deception planning, by (1) presenting a formal deception logic for modeling cyber deception, and (2) introducing a deception framework that augments this formal modeling with necessary quantitative reasoning tools to generate coordinated deception plans. To show expressiveness and evaluate effectiveness and overhead of the framework, we use it to model and solve two important deception planning problems: (1) strategic honeypot planning, and (2) deception planning against route identification. Through these case studies, we show that the generated deception plans are highly effective and outperform alternative random and unplanned deception strategies.

1. Introduction

While reactive detection-based defense technologies remain a core component of cyber defense, they are no longer adequate in addressing the ever-increasing threats of evolving cyber attacks [1]. In recent years, we have witnessed novel classes of advanced and persistent attacks using stealthy or zero-day techniques that are not fully detectable by these reactive defense technologies [2]. Examples are very stealthy indirect link flooding attacks [3, 4] or on-the-rise Advanced Persistent Threats (APT) [5]. In lack of effective reactive countermeasures, defensive deception, as a proactive defense paradigm, could play a significant role in resisting against such stealthy and undetectable threats [1].

While a myriad of deception systems have been proposed in the literature, very few works have focused on the idea of developing strategies, theories, and techniques for a strategic combination of various deception systems in a goal-oriented manner. Meanwhile, effective deception of skilled attackers requires, not just a bunch of deception systems, but a *strategic combination* of a group of coordinated systems in a manner that their coherent system-wide deceptive image manipulates an attackers' thinking and leads them to a false reconnaissance and thus false attack planning. For instance, consider the very stealthy and undetectable *indirect* link flooding attacks such as Crossfire [3] and Coremelt [4], where the attacker tries to find all active routes to a target area, in order to identify links that are most critical to the connectivity of that area and only floods those links. Assume an ISP intends to defeat such route identifications by deceiving the attacker about parts of the routes that are under its control. These fake routes are announced by identifying `traceroute` packets and giving spoofed replies to them. One approach to this aim is to present each querying source with an isolated and random fake route in the network. This will confuse attackers about the routing topology (link map [3]) and may achieve some arbitrary effectiveness. However, by strategically selecting a set of coordinated fake routes, we can mislead the attacker to a better goal with higher benefit: misidentifying a set of trivial links as critical (min-cut) and flooding those links instead. This will rescue benign user traffic from flooding and lead to a benefit that is much higher than the sum of benefit from those individual spoofed replies.

Solving deception problems of this kind requires a *deception planning framework* that provides necessary paradigms for defining various deception

URI: https://hdl.handle.net/10125/63973
978-0-9981331-3-3
(CC BY-NC-ND 4.0)

actions, along with their inter-dependencies, benefits, and costs. More importantly, such a framework must provide paradigms for modeling how such deceptive actions would as a whole manipulate cognitive thinking process of attackers who may have different goals and sophistication levels. It must also be able to reason and identify the most beneficial deception plan (set of actions) with the given budget. We show that this planning problem is a generalization of the 0-1 knapsack problem and thus it is NP-hard.

In this paper, we present a framework for modeling and identifying optimal deception plans for cyber deception problems. The framework introduces a *deception logic* for defining deception models, where each deception model addresses a specific cyber threat model. The deception logic is an abstraction over satisfiability modulo theories (SMT) [6].

Our framework provides necessary user interfaces for knowledge engineers to define their deception models. Once defined, the given deception model is then synthesized into an SMT instance, which is then solved by the underlying SMT solver, for which we use Microsoft Z3 Theorem Prover [6]. The Z3 SMT solver finds a satisfiable assignment to the given instance. These assignments define what set of deceptive actions are optimal and are referred to as *deception plan*. Each deceptive action is then actuated in the system by a set of deception techniques that are implemented in the system.

We use our framework to solve two deception planning problems as case studies. First, we solve the problem of *strategic honeypot planning*, where the problem is to determine the optimal number and types of honeypots (regarding fidelity level) for an enterprise, given its mission, budget and sophistication level of potential attackers; this deception plan targets remote network reconnaissance and infiltration attacks. Through comparison with alternative honeypot planning paradigms, we show that the generated deception plan is effective. Secondly, we model the aforementioned problem of deception planning against route identification attacks. Through simulation, we show that the generated fake routes incur maximum deception on attackers. For both models, we discuss and evaluate the cost of model generation and planning, and demonstrate that the framework is highly expressive and fairly scalable.

2. Related Work

In early 2000s, several seminal works [7, 8, 9, 10] proposed different type of honeypots. Since then, defensive deception has been considered as one of the pillars of cyber defense. In this section, we investigate existing works focused on devising a formal theory, logic, or framework for modeling and planning cyber deception.

Several logic-based models of deception exist in the formal methods literature. Sakama *et al.* [11] introduce a modal logic for deception that can express belief, action, and intention and formulate eight different categories of deception. Authors in [12] introduce a propositional multi-modal logic to formulate various types of dishonest communications between agents. In [13], Rowe presents an automated deception planner that take a sequence of operating system commands as input and find viable deception plans that are consistent with constraints which are defined in second-order logic. Rosis *et al.* [14] propose a formal deception planning model that formalizes information impact on receiver's mind. These logical models cannot be directly used to address deception planning problem; however, they can be used to formally express both an act of deceiving and its effect on addressee's belief. Duan *et al.* present a multi-layered deception system called CONCEAL [15], which composes mutation (random changing of parameter values), anonymity, and diversity to defeat reconnaissance attacks on computer networks.

Another class of approaches focuses on modeling deceptive engagement between attacker and defender using game theory. Signaling game [16, 17, 18, 19] has been the primary modeling paradigm since it is inherently a natural fit for deception planning. Ettinger *et al.* [19] present game-theoretic modeling of bargaining tactic problem using belief manipulation. Carroll *et al.* [17] investigate the effects of deception on the interaction between an attacker and a defender in a computer network. The game-theoretic models, although insightful, only target specific deception problems and are not expressive or scalable in defining deception models in a more generic context.

Another class of models uses probabilistic approaches for deception planning. Rowe [20] presents a probabilistic model based on a decision tree to determine whether a single deception plan should be performed. In another work, Rowe [21]

introduces an obstructive counter-planning model that uses a probabilistic approach to determine the optimal set of deceptive mechanisms (called ploys) to deter the attack. Jajodia *et al.* [22] present probabilistic modeling of the problem of defeating network scanners by generating effective fake scan results to minimize the damage the attacker can produce.

While existing models provide useful insights into deception modeling and planning, none of them is generic and expressive enough to formulate a wider range of cyber deception planning against different threat models.

3. A Deception Modeling and Planning Framework

3.1. Deception Modeling Logic

A Deception model DM is formally defined as a tuple:

$$DM = \langle \underbrace{\langle \phi, T, \tau \rangle}_{facts}, \underbrace{\langle \boldsymbol{\beta} \rangle}_{beliefs}, \underbrace{\langle A, C, P, \boldsymbol{\alpha} \rangle}_{actions}, \underbrace{CR}_{caus.\ rules}, \underbrace{\langle \Theta, \chi \rangle}_{attackers}, \underbrace{\langle G, I \rangle}_{goal}, \underbrace{\langle B \rangle}_{budget} \rangle \quad (1)$$

In a deception model, only β (beliefs) and α (actions) are variables, and all other components are inputs to the model. Next, we formally define each of these components.

3.2. Facts

A deception problem is described in terms of *attributes*, where each attribute describes a factual item (*e.g.*, configuration parameter) of the system; *e.g.*, operating system of a host, criticality level of a server, route between two hosts, *etc.* The set of attributes is denoted as $\phi = \{\varphi_1, \ldots, \varphi_m\}$.

T_{φ_i} denotes the data type of φ_i; *i.e.*, the set of possible (alternate) values for φ_i, which could be either *Boolean*, *Ordered* (*e.g.*, low, medium, high), or *integral* (*e.g.*, from 1 to 100). The symbol T denotes the set of all attribute data types in the system.

The facts of the system are described in terms of attribute values. The real value of attribute φ_i is denoted as $\tau_{\varphi_i} \in T_{\varphi_i}$.

3.3. Attacker Types

The set of attacker types is denoted as $\Theta = \{\theta_1, \ldots, \theta_n\}$, where a probability distribution function χ determines the adversary type distribution over Θ; *i.e.*, $\chi(\theta_i)$ determines the likelihood that attacker is of type θ_i. Distribution over adversary types are given to the model as input.

3.4. Beliefs

An attacker's belief on an attribute denote her expected perception of that attribute. This belief could be different from the actual value of that attribute. We define variable $\boldsymbol{\beta}_{\varphi_i}$ to denote attacker's belief on attribute φ_i. In other words, $(\boldsymbol{\beta}_{\varphi_i} = v)$ means that adversary believes that the value of attribute φ_i is v. When the model has more than one attacker type, $\boldsymbol{\beta}_{\varphi_i}^{\theta_k}$ denotes belief of an attacker of type θ_k over φ_i.

The belief value v must be in $(T_{\varphi_i} \cup \psi)$, where symbol ψ is called the *null* or *undefined* value that denotes lack of knowledge. The attribute value (τ_{φ_i}) can never be ψ; *i.e.*, the fact can not be undefined.

3.5. Actions

An action denotes a deceptive representation of a fact in the system that directly manipulates the attacker's belief on one attribute. From manipulability standpoint, attributes are categorized into two categories: *actionable* and *derivative*. Actionable attributes are those for which attackers' belief can be manipulated directly via a deceptive action. Derivative attributes are those for which attackers' belief cannot directly be manipulated; rather, it can only be derived from her beliefs over other attributes.

The set of admissible actions on attribute φ_i is denoted as A_{φ_i}. The objective of the framework is to determine which action from A_{φ_i} must be selected such that the deception goal is achieved. The action on attribute φ_i is denoted as *action* variable $\boldsymbol{\alpha}_{\varphi_i} \in (A_{\varphi_i})$, which must be assigned by the framework (Z3 solver). The deception plan is essentially comprised of assigned action variables.

Action Plausibility. One of the key elements of a successful deception is that the deception should conform to the expectations of the target (*i.e.*, the attacker). This feature of deception is called plausibility [23]. To achieve plausibility, the deception plan must present a coherent and consistent deceptive depiction of the system. This plausibility is modeled as a system-dependent set of constraints over possible actions that could *not* be assigned to attributes at the same time and defines semantic inter-dependency among actions. These

conflicting action pairs are called *implausibility pairs* and are given to the model as inputs:

$$P = \{(a \in A_{\varphi_i}, b \in A_{\varphi_j})\}$$

where $(a \in A_{\varphi_i}, b \in A_{\varphi_j})$ denotes an implausible pair of actions on attributes φ_i and φ_j.

In synthesis of the deception model to its corresponding SMT instance, the following constraint is generated and incorporated in the instance for each implausible pair:

$$(\boldsymbol{\alpha_{\varphi_i}} = a) \rightarrow (\boldsymbol{\alpha_{\varphi_j}} \neq b) \quad \text{[Added in Synthesis]} \qquad (2)$$

Example: Let srv_{25} and srv_{80} denote the type of FTP and Web servers running on a network host. The following domain constraint could be defined, because *vstfpd* is a Linux service, while *IIS* is a Windows service. So, *(IIS, vstfpd)* constitute an implausible pair.

$$(\boldsymbol{\alpha}_{srv_{25}} = \text{vstfpd}) \rightarrow (\boldsymbol{\alpha}_{srv_{80}} \neq \text{IIS})$$

Action Costs. Applying actions on attributes are costly, as it requires misrepresentation of attribute values. We assume that costs are all expressible as monetary values, and reflect the total price of building and actuating the deceptive action. The cost associated with applying action $a \in A_{\varphi_i}$ on attribute φ_i is denoted by a numerical value $c_{\varphi_i}(a)$ which is given as input to the model.

The total available budget, B, is also given to the model as input. In the synthesis process, the following constraint is incorporated in the deception model to ensure that the linear summation of action costs in a plan does not surpass the available budget, B:

$$\sum_{\varphi_i} \sum_{a_l \in A_{\varphi_i}} (\boldsymbol{\alpha}_{\varphi_i} = a_l) \cdot c_{\varphi_i}(a_l) \leq B \quad \text{[Added at Synthesis]} \qquad (3)$$

3.6. Causation Rules

We use the logical implication ($\rightarrow$) to denote how a combination of certain facts, actions, and/or belief induces a consequent belief in attacker's mind. Specifically, causation rules of an attribute determine how (cause) an attacker's belief regarding that attribute is manipulated (effect). A causality rule has the following general form:

$$(\boldsymbol{\Gamma}_1 \wedge \ldots \wedge \boldsymbol{\Gamma}_k) \vee \ldots \vee (\boldsymbol{\Gamma}_n \wedge \ldots \wedge \boldsymbol{\Gamma}_m) \rightarrow (\boldsymbol{\beta}^{\boldsymbol{\theta_i}}{}_{\varphi_k} = v) \quad (4)$$

where $\boldsymbol{\Gamma_i}$ is a constraint predicate (or its negation) as defined next. Note that antecedent is in disjunctive normal form (DNF). The consequent is a belief variable over attribute φ_k and $v \in T_{\varphi_k}$.

Constraint predicate is an atomic condition over facts, belief, or actions and has one of the following types:

Fact Constraint: constraint on attribute values, τ_{φ_i}. It is of the form $(\tau_{\varphi_i} = v)$ where $v \in T_{\varphi_i}$.

Action Constraint: constraint on the *action* type. It is of form $(\boldsymbol{\alpha}_{\varphi_i} = a)$ where $a \in A_{\varphi_i}$.

Belief Constraint: constraint on adversary's beliefs, which is of the form $(\boldsymbol{\beta}_{\varphi_i} = v)$ where $v \in (T_{\varphi_i} \cup \psi)$.

3.7. Deception Goal

Conceptually, the intention of deception is to derive an adversary toward certain false conclusions. In order to allow scientific reasoning for cyber deception, and make various deception models comparable, we need to quantify the benefit associated with each potential deception plan. The benefit of a deception plan depends on the individual benefit of certain *goal* beliefs, *i.e.*, beliefs that an attacker must be driven to or away from. The knowledge engineer determines the monetary (financial) impact of these goal beliefs, which could be either negative (for beliefs that are undesirable for defense) or positive (for beliefs that are contributing to defense).

The impact associated with a belief $(\boldsymbol{\beta}_{\varphi_i} = v')$ when its truthful value is v $(\tau_{\varphi_i} = v)$ is denoted as value $I_{\varphi_i}(v \rightarrow v')$. This impact value shows the significance of deceiving an attacker to believe that φ_i is v' when in fact its value is v.

Also, value G, which is given as input, defines the minimum acceptable benefit that a deception plan must provide. Given belief impacts and the goal values, during the synthesis process the following constraint is incorporated in the SMT instance:

$$\sum_{\theta_k \in \Theta} \sum_{\varphi_i} \sum_{v \in T_{\varphi_i}} \sum_{v' \in T_{\varphi_i} \cup \{\psi\}} (\chi(\theta_k) \cdot (\tau_{\varphi_i} = v) \cdot (\boldsymbol{\beta}^{\theta_k}_{\varphi_i} = v') \cdot I_{\varphi_i}(v \rightarrow v')) \geq G \quad \text{[added at synthesis]} \qquad (5)$$

3.8. Solving Deception Model

This deception modeling problem is reducible to 0-1 knapsack problem, where beliefs are items, impacts are their item values, costs are their weights, and

budget is the maximum weight capacity. The knapsack problem is known to be NP-hard. This is why we convert the problem to a satisfiability problem, using generalized Boolean/arithmetic format of satisfiability modulo theories (SMT) [6], in order to make it solvable in a scalable manner.

After defining the deception model, the synthesis module incorporates implausibility constraints (Eq. 2), budget constraint (Eq. 3) and goal constraint (Eq. 5) in the model and creates a Z3 SMT instance. Then, using the underlying Z3 SMT solver, the framework solves the model and determines appropriate assignments to variables. The deception plan is a satisfiable solution to the SMT instance, determined by the SMT solver. The deception plan is comprised of values assigned to only action variables. If the plan is not satisfiable, the model constraints need to be relaxed to make it satisfiable. In the next sections, we use the framework to model and solve two deception planning problems.

4. Case Study A: Strategic Honeypot Planning

Honeynet mapping aims to identify honeypots located inside a production network from real production machines, in order to exclude them from the attack. So while low-interaction honeypots are potentially not detectable by naive attackers, their decoy nature is easily detectable even by remote network reconnaissance [24] due to their low fidelity. However, they are cheap in terms of deployment and maintenance [8, 25]. In contrast, high-interaction honeypots have a higher level of fidelity. They are usually implemented using software-based full virtualization, which makes them hard to detect by remote network reconnaissance; however, as mentioned above they are noticeably costlier than low-interaction honeypots [8, 26].

Now, assume we aim to protect an enterprise network from network reconnaissance attacks, ranging from automated worms and scanners to advanced reconnaissance. Given a limited budget, we aim to place a number of honeypots of either type (low-interaction or LI, high-interaction or HI) in our DMZ. The LIs are effective in slowing down attacks launched by *naive* attackers but are detectable by *elite* adversaries. However, their cost is very low. In contrast, HIs are not detectable by elite attackers, but their deployment is costly. The network DMZ has a limited number of unused public addresses that could be assigned to honeypots. Also, the budget is limited and the total cost should be within the bounds of the available budget. Attacker types distribution is determined based on enterprise mission and history of attacks. Given this expected attacker distribution as input, our problem is to *determine the number of LI and HI honeypots that could be placed in the DMZ to achieve a high deception benefit.*

4.1. Deception Model

Assume the DMZ includes n hosts and m public addresses. Therefore, the address space can have $(m-n)$ honeypots. Next, we define the deception model.

Attributes: attribute s_i denotes the status of IP address IP_i in the public (DMZ) address space: $\boldsymbol{\varphi} = \{s_1, \dots s_m\}$

Attribute Types: attribute type of an IP address denotes alternative configurations that an address could have. An IP address may be dark, assigned to a real host, or assigned to a decoy: $T_{s_i} = \{dark, real, decoy\}$

Facts: For a dark address IP_i, $(\tau_{s_i} = dark)$ and for a live address IP_j, $(\tau_{s_j} = real)$.

Adversary Types: attackers could be either naive or elite: $\Theta = \{naive, elite\}$.

$\chi(naive)$ and $\chi(elite)$ denote distribution of attackers, and in Section 4.2 we investigate effect of various distributions on produced deception plans.

Action Types: the action type denotes alternative actions that could be taken on an IP address. We may either place a LI or HI honeypot on an IP address or leave it as it is: $\boldsymbol{A_{s_i}} = \{\psi, LI, HI\}$

Causation Rules: causation rules define how the placement of honeypots affects attackers' beliefs. These rules are extracted by the knowledge engineer as part of defining the deception model.

Rule 1 (Real Host): all attackers (naive or elite) would eventually identify a real host as real host.

$$(\tau_{s_i} = real) \rightarrow (\beta_{s_i}^{naive} = real), (\beta_{s_i}^{elite} = real)$$

Rule 2 (Dark Address): all attackers would identify a dark address.

$$(\tau_{s_i} = dark) \wedge (\alpha_{s_i} = \psi) \rightarrow (\beta_{s_i}^{naive} = dark), (\beta_{s_i}^{elite} = dark)$$

Rule 3 (LI): LI honeypots are not detectable by naive attackers, but detectable by elite ones.

$$(\tau_{s_i} = dark) \wedge (\alpha_{s_i} = LI) \rightarrow (\beta_{s_i}^{naive} = real), (\beta_{s_i}^{elite} = decoy)$$

Rule 4 (HI): HI honeypots are not detectable by any attacker type.

$$(\tau_{s_i} = dark) \wedge (\alpha_{s_i} = high) \rightarrow (\beta_{s_i}^{naive} = real), (\beta_{s_i}^{elite} = real)$$

Actions Costs: the cost of creating LI honeypots are much lower than HI honeypots.

$$c_{s_i}(\psi) = 0\$, c_{s_i}(low) = 10\$, c_{s_i}(high) = 100\$$$

Belief Impacts: defines the impact of goal beliefs on deception benefit.
Impact 1 (an attacker correctly identifies a real host): $I_{s_i}(real \rightarrow real) = -50\$$
Impact 2 (an attacker misidentifies a honeypot as real host): $I_{s_i}(dark \rightarrow real) = 100\$$
Impact 3 (an attacker correctly identifies a honeypot): once a honeypot is detected, it becomes useless and merely a waste of resources: $I_{s_i}(dark \rightarrow decoy) = -20\$$
Impact 4 (an attacker correctly identifies a dark address): $I_{s_i}(dark \rightarrow dark) = 0\$$
Goal and Budget: in the next section, we evaluate the model with various goal and budget values.

4.2. Effectiveness Analysis

Fig. 1 and 2 denote the crafted deception plan for a DMZ with $m = 128$ addresses and $n = 12$ hosts for different budgets and different attacker type distributions. In our evaluation, instead of having a minimum benefit value (G), we aim to determine maximum attainable benefit value for every evaluation scenario. Note that when the budget is low (Fig. 1), the number of HI honeypots is very low, even when all attackers are expected to be elites ($\chi(elite) = 1$). This is because, for a low budget, the cost of a HI honeypot is hardly justifiable. However, as the budget increases (Fig. 2), HI honeypots become affordable, especially when the probability of facing elite attackers increases.

Also, note that even for high budgets and even when both attacker types have the same probability ($\chi(elite) = \chi(naive) = 0.5$), deception plan includes more LI honeypots than HI. This is because, in our deception model, we assume that the advantage (impact) of deceiving a naive attacker (100) far exceeds the disadvantage of failing to deceive an elite one (-20). However, as Fig. 1 and 2 indicate, when the probability of facing elite attackers becomes ≥ 0.8, LI honeypots are not justifiable anymore (more costly than beneficial) and there is a steep decrease in their number.

Fig. 4 and 5 compare the benefit (Eq. 5) of the optimal deception plan generated by the framework with the following alternative plans, for $\chi(elite) = \chi(naive) = 0.5$:

- All honeypots are LIs.
- All honeypots are HIs.
- Half honeypots are LIs, half HIs: in this strategy budget distribution conforms to attacker types; since both attacker types have the same probability of 0.5, half the budget is assigned to LIs and the other half to HIs.

Note that the deception benefit of the optimal plan is always higher or equal to other alternative strategies. When the budget is low (Fig. 4), the optimal strategy is very close to the all-LI strategy. This is also the case for when the address space size is too large. However, in Fig. 5, note that when the budget is high enough for the given address space, the benefit of the optimal plan is almost 50% higher than that of all-LIs.

4.3. Planning Cost

As discussed in 3, the deception planning problem is a generalization of 0-1 knapsack problem, where impacts are values and costs are weights, each item (belief) can be selected at most once, and our objective is to select beliefs that have maximum aggregate benefit for a given budget (knapsack size).

With respect to the honeypot planning problem, we have two parallel knapsacks (address space size and budget) and also negative values (impacts). While the dynamic programming approach to solving the original 0-1 knapsack problem is known to be pseudo-polynomial (polynomial in input size), developing heuristics for solving our generalized knapsack problem is not as straightforward.

Fig. 3 shows planning time for DMZs with various address space sizes (m) and differing number of hosts. Note that planning time increases exponentially with the address space size because the number of variables in the corresponding SMT instance is a function of m (and also n). Also, note that for higher budgets the planning time is lower because a less strict budget exerts less constraint on the solver.

5. Case Study B: Strategic Network Topology Obfuscation

Discovering active routes to a target area (*e.g.*, an enterprise network) is precursory reconnaissance

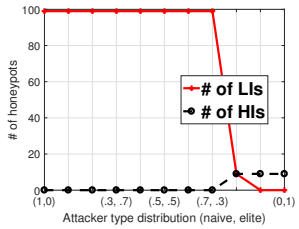

Figure 1. Dec. plan (Budget = 1,000$)

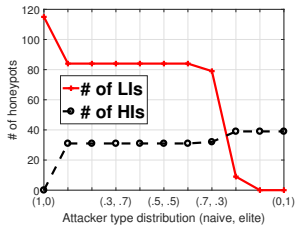

Figure 2. Dec. plan (budget = 4,000$)

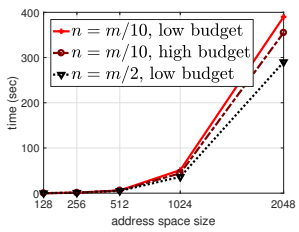

Figure 3. Solving time for various network sizes

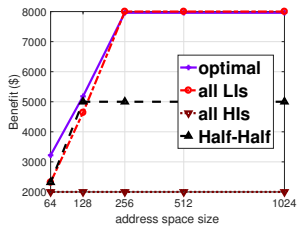

Figure 4. Comparing dec. benefit for various plans (budget = 1,000$)

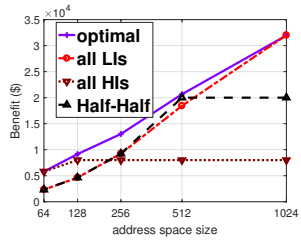

Figure 5. Comparing dec. benefit for various plans (budget = 4,000$)

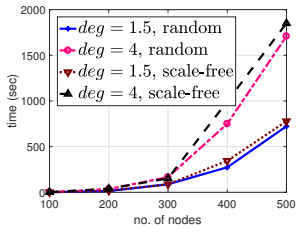

Figure 6. Model generation and solving time for various network sizes

step for a variety of network threat models such as link flooding DDoS attacks, and man-in-the-middle and eavesdropping attacks. The route discovery allows attackers to identify critical network links that are responsible for transiting majority of traffic to a target area and focus their attack on those links. For example, by identifying and only flooding the critical links in the *minimum cut* of the graph of routes between a group of bots and a target area, attackers can cut off *all* the traffic to that target area [3, 4]. Examples of such link flooding attacks are Crossfire [3] and Coremelt [4] which are so effective that they can cut off 33% of all connections to the West Coast of the US.

Detecting such stealthy flooding is very hard, primarily because the flows are not destined to (and so are not visible at) the target area, and so the defense must be implemented by an external entity at the ISP level [3].

A straightforward approach for attackers to discover routes to a target area is to issue `traceroute` queries from geographically distributed bots [3] to this area. ISP can deceive potential attackers by *lying* about the routes to the defended area at border gateways by providing fake responses to these traceroute packets. In providing such fake replies, the fundamental deception planning question is what fake routes could be announced to deceive attackers maximally and achieve the best protection for the defended area. Our objective in this section is to model and solve this problem using our deception framework.

5.1. Deception Model

The potential adversarial traceroutes enter from a set of *entry* gateways, denoted as S and exits ISP network to the defended area from a set of *exit* gateways, denoted as T. We assume that the attacker's traceroutes are issued from a set of geographically distributed bots [3], and therefore all the routes from S to T are ultimately probed and identified by the attacker. The set of routes from S to T denote the routing topology of the defended area that is under the control of the ISP. We denote this real routing topology graph as G^r. This G^r routing graph is a sub-graph of the ISP physical network topology, denoted as G.

After discovering this routing graph G^r, the attacker would determine the set of critical links by identifying the minimum cut of this graph. Our goal is to present a fake routing graph G^f for a defended area (again a sub-graph of G) such that attacker misidentifies the critical links. By persuading DDoS attackers to flood non-critical links (that are not part of our routes to the defended area), we will be able to rescue a significant portion of the defended area's users from this flooding. Using the deception plan, we intend to identify the optimal G^f that could be presented to attackers to achieve minimum cut-off for clients of the defended area.

The route discovery deception planning problem

is formalized as follows:

Attributes: attribute $rt_{s,t}$ denotes the route from entry gateway $s \in S$ to exit gateway $t \in T$. Attribute cs denotes the set of critical links (on minimum cut) of the graph.

Attribute Types: for each $rt_{s,t}$, there are a number of candidate simple paths. While enumeration of all simple paths between two nodes in graph is generally not polynomial, we can either use an upper bound K for maximum path lengths or use heuristic algorithms such as [27] to discover the candidate routes between a s-t pair in polynomial time. Assume $r_1, \ldots, r_k$ denote the simple paths between s and t. We have: $T_{rt_{s,t}} = \{r_1, \ldots, r_k\}$.

Action Types: the attribute cs is a derivative (non-actionable) attribute. The action type for each route attribute $rt_{s,t}$ is the same as its attribute type: any $r_k \in T_{rt_{s,t}}$ could be represented as the route between s and t.

Adversary Types: we assume only one adversary type for this threat model.

Causation Rules: the causation rules determine attackers' belief on the critical set for every potential fake routing topology (G^f).

Rule 1: fake replies are not distinguishable from real replies. Therefore, if the route r_k is announced for an s-t pair, an attacker would believe that r_k is the route between s and t.

$$(\alpha_{rt_{s,t}} = r_k) \rightarrow (\boldsymbol{\beta}_{rt_{s,t}} = r_k)$$

Rule 2: if we announce fake routes for all s-t pairs, we mislead attackers to a different critical set. For every possible assignments to all $rt_{s,t}$, we need a causation rule of the following form:

$$\bigwedge_{s \in S, t \in T} (\boldsymbol{\beta}_{rt_{s,t}} = r_i) \rightarrow \boldsymbol{\beta}_{cs} = mincut_k, \forall r_i \in T_{rt_{s,t}} \quad (6)$$

where $mincut_k$ is the minimum cut for the graph generated by route assignments to all action variables $\alpha_{rt_{s,t}}$. Note that the left-hand side includes exactly one assignment for every and each s-t pair.

Action costs: while we can assume higher costs for longer routes, the increase in the number of fake responses is negligible and therefore we assume that lying on all routes has equal costs.

Belief Impacts: $CutRatio(set_k, G^r)$ is the value that denotes the ratio of s-t pairs that are disconnected when $mincut_k$ is cut off (deleted) from the real routing topology, G^r.

$$I_{cs}(\tau_{cs} \rightarrow mincut_k) = 1 - CutRatio(mincut_k, G^r)$$

Budget and Goal: to investigate the upper bound of benefit, we assume the budget is not limited. The goal is to achieve a deception benefit of $G = 1$, which means no link on min-cut of G^f is on an s-t route in G^r; *i.e.*, the cut-off is 0. If $G = 1$ is not satisfiable, we reduce it by a small δ and then solve the model again, until the model becomes satisfiable.

5.2. Effectiveness Analysis

To understand how effective this deception planning is, we investigate the optimal deception plan for a small example network in Fig. 7 and 8. Fig. 7 shows the real routing topology between S and T, and also its minimum cut. Fig. 8 shows the same network with the optimal fake routing topology and its minimum cut. Note that if the fake min-cut is deleted from the real routing topology, no route would be disconnected and the cut-off ratio would be 0; *i.e.*, the deception benefit would be 1.

To evaluate the deception benefit for large networks, we developed two programs. One, a random graph generator program that we used to generate networks with n nodes and average node degree d according to either the Erdos-Renyi (random graph) or Barabasi-Albert (scale-free) models. Second, an automated deception plan generator that we used to create deception plans for a given input graph.

Fig. 9 shows the deception benefit for random graphs with various sizes and connectivity. To calculate average deception benefit for each data point, we generated and investigated 100 different random graphs of the same size. In the figure, note that the deception benefit increases with network size (n), because the larger the network, the more the number and diversity of alternative routing topologies that could be offered to the attacker. For example, for $d = 1.5$, a random network with $n = 300$ has an average deception benefit of 0.8, while a network with the same average degree, but $n = 100$ achieves a deception benefit that is slightly higher than 0.5.

Also, note that the deception benefit increases for larger average node degrees (*i.e.*, higher number of links), again because the number of alternative routing topologies that could be built on top of the network graph increases, and so the deception space is larger. In the figure, note that for all network sizes, when $d = 4$, the deception benefit is 1.

Fig. 10 calculates deception benefit for scale-free (Power law) networks. Note that, like random

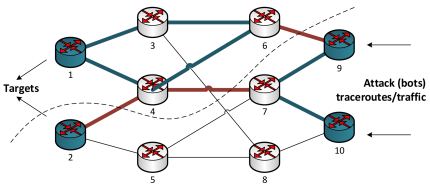

Figure 7. Real (observable) routing topology

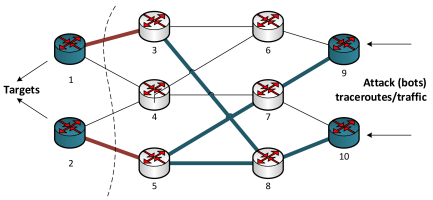

Figure 8. Fake (announced) routing topology

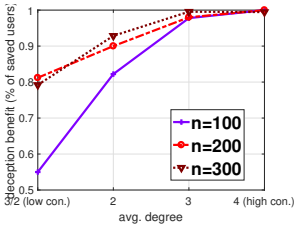

Figure 9. Deception Benefit for random networks with various sizes

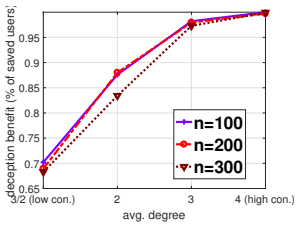

Figure 10. Deception Benefit for scale-free networks with various sizes

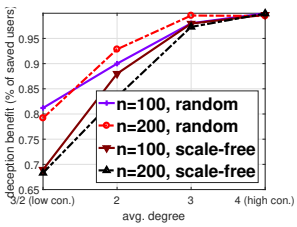

Figure 11. Comparing deception benefit for random vs. scale-free networks

networks, the deception benefit increases with the average node degree. However, contrary to random networks, the deception benefit does not increase with network size (n) noticeably. This is due to the existence of hub nodes in the scale-free network that increases the probability of overlaps between alternative routing topologies of the network.

Fig. 11 compares deception benefit for random vs. scale-free networks of the same size and connectivity. As expected, for the same network size and connectivity, a random graph has higher deception benefit than a scale-free one, because, in a scale-free graph, alternative routing topologies have a higher probability of overlap.

5.3. Planning Cost

The planning time for a given network of size n and node degree d includes (I) time for generating the deception model, and (II) time for solving the generated model by the framework. To generate the deception plan for a given network with given entry and exit sets, we need to enumerate the simple paths between all entry-exit node pairs. Since the number of simple paths could be exponential, we only find simple paths with length $\leq k{\cdot}D$ where D is the diameter (the longest shortest path between any two nodes) of the network graph. Each alternative routing topology corresponds to a causation rule of form 6. An important step in the planning process is generating min-cuts and calculating their deception benefit, which is done by the deception model generator. So, while solving the model (finding a satisfiable assignment) for the solver is straightforward, the number of candidate routing topologies could be large, making the problem computationally hard to solve.

Fig. 6 shows the time for generation and solving of the deception model for various network sizes. Note that the planning time increases exponentially with network size, because of the exponential increase in the number of simple paths. This effect is even higher for larger average node degrees. For example for $n = 500$, planning time for a network with $d = 4$ is more than twice higher as compared to that of a network with $d = 3/2$. Also, note that planning time for random networks is slightly lower than that of scale-free networks. Finally, since the deception space increases exponentially with the network size, solving the problem for very large network sizes ($n = 15,000$) is computationally challenging, thus limiting the scalability of the model.

6. Conclusion

In this paper, we present a framework for deception modeling and planning against cyber threats. The deception models are defined based on a deception logic which is an abstraction over satisfiability modulo theories (SMT). We used the framework to model and solve two deception problems. First, we solved the strategic honeypot planning problem. We showed that the resulting deception plan outperforms alternative plans that could be envisioned with the same budget.

In the second problem, we showed that by representing carefully planned fake replies to `traceroute` queries, we can mislead attackers to attack links that are not critical for transmitting flows of an enterprise. We showed that with enough size and link diversity in the underlying physical network, this scheme could save up to 100% of network traffic from such attacks.

References

[1] N. Virvilis, B. Vanautgaerden, and O. S. Serrano, "Changing the game: The art of deceiving sophisticated attackers," in *Cyber Conflict (CyCon 2014), 2014 6th International Conference On*, pp. 87–97, IEEE, 2014.

[2] S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, *Moving target defense: creating asymmetric uncertainty for cyber threats*, vol. 54. Springer Science & Business Media, 2011.

[3] M. S. Kang, S. B. Lee, and V. D. Gligor, "The crossfire attack," in *Security and Privacy (SP), 2013 IEEE Symposium on*, pp. 127–141, IEEE, 2013.

[4] A. Studer and A. Perrig, "The coremelt attack," in *European Symposium on Research in Computer Security*, pp. 37–52, Springer, 2009.

[5] L. Martin, "Cyber kill chain®," *URL: http://cyber. lockheedmartin. com/hubfs/Gaining_the_Advantage_Cyber_Kill _Chain. pdf*, 2014.

[6] N. Bjørner and L. de Moura, "Z3 : Applications, enablers, challenges and directions," in *Sixth International Workshop on Constraints in Formal Verification*, 2009.

[7] L. Spitzner, "Honeypots: Catching the insider threat," in *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pp. 170–179, IEEE, 2003.

[8] N. Provos *et al.*, "A virtual honeypot framework.," in *USENIX Security Symposium*, vol. 173, 2004.

[9] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling, "The nepenthes platform: An efficient approach to collect malware," in *Recent Advances in Intrusion Detection*, pp. 165–184, Springer, 2006.

[10] T. Sochor and M. Zuzcak, "Study of internet threats and attack methods using honeypots and honeynets," in *Computer Networks*, pp. 118–127, Springer, 2014.

[11] C. Sakama and M. Caminada, "The many faces of deception," *Proceedings of the Thirty Years of Nonmonotonic Reasoning*, 2010.

[12] C. Sakama, M. Caminada, and A. Herzig, "A formal account of dishonesty," *Logic Journal of IGPL*, vol. 23, no. 2, pp. 259–294, 2015.

[13] N. C. Rowe, "Finding logically consistent resource-deception plans for defense in cyberspace," in *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, vol. 1, pp. 563–568, IEEE, 2007.

[14] F. De Rosis, V. Carofiglio, G. Grassano, and C. Castelfranchi, "Can computers deliberately deceive?," *Computational Intelligence*, vol. 19, no. 3, pp. 235–263, 2003.

[15] Q. Duan, E. Al-Shaer, M. Islam, and H. Jafarian, "Conceal: A strategy composition for resilient cyber deception-framework, metrics and deployment," in *2018 IEEE Conference on Communications and Network Security (CNS)*, pp. 1–9, IEEE, 2018.

[16] N. Garg and D. Grosu, "Deception in honeynets: A game-theoretic analysis," in *Information Assurance and Security Workshop, 2007. IAW'07. IEEE SMC*, pp. 107–113, IEEE, 2007.

[17] T. E. Carroll and D. Grosu, "A game theoretic investigation of deception in network security," *Security and Communication Networks*, vol. 4, no. 10, pp. 1162–1172, 2011.

[18] J. Zhuang, V. M. Bier, and O. Alagoz, "Modeling secrecy and deception in a multiple-period attacker–defender signaling game," *European Journal of Operational Research*, vol. 203, no. 2, pp. 409–418, 2010.

[19] D. Ettinger and P. Jehiel, "A theory of deception," *American Economic Journal: Microeconomics*, vol. 2, no. 1, pp. 1–20, 2010.

[20] N. Rowe, "Planning cost-effective deceptive resource denial in defense to cyber-attacks," in *Proceedings of the 2nd International Conference on Information Warfare & Security*, p. 177, 2007.

[21] N. C. Rowe, "Counterplanning deceptions to foil cyber-attack plans," in *Information Assurance Workshop, 2003. IEEE Systems, Man and Cybernetics Society*, pp. 203–210, IEEE, 2003.

[22] S. Jajodia, N. Park, F. Pierazzi, A. Pugliese, E. Serra, G. I. Simari, and V. Subrahmanian, "A probabilistic logic of cyber deception," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2532–2544, 2017.

[23] N. C. Rowe, "Designing good deceptions in defense of information systems," in *Computer Security Applications Conference, 2004. 20th Annual*, pp. 418–427, IEEE, 2004.

[24] S. Mukkamala, K. Yendrapalli, R. Basnet, M. Shankarapani, and A. Sung, "Detection of virtual environments and low interaction honeypots," in *Information Assurance and Security Workshop, 2007. IAW'07. IEEE SMC*, pp. 92–98, IEEE, 2007.

[25] X. Fu, W. Yu, D. Cheng, X. Tan, K. Streff, and S. Graham, "On recognizing virtual honeypots and countermeasures," in *Dependable, Autonomic and Secure Computing, 2nd IEEE International Symposium on*, pp. 211–218, 2006.

[26] S. Antonatos, K. Anagnostakis, and E. Markatos, "Honey@ home: a new approach to large-scale threat monitoring," in *Proceedings of the 2007 ACM workshop on recurring malcode*, pp. 38–45, ACM, 2007.

[27] F. Rubin, "Enumerating all simple paths in a graph," *IEEE Transactions on Circuits and Systems*, vol. 25, no. 8, pp. 641–642, 1978.