Chapter 6
CONCEAL: A Strategy Composition for Resilient Cyber Deception: Framework, Metrics, and Deployment

Qi Duan, Ehab Al-Shaer, and Mazharul Islam

Abstract Cyber deception is a key proactive cyber resilience technique to reverse the current asymmetry that favors adversaries in cyber warfare by creating a significant confusion in discovering and targeting cyber assets. One of the key objectives for cyber deception is to hide the true identity of the cyber assets in order to effectively deflect adversaries away from critical targets, and detect their activities early in the kill chain.

Although many cyber deception techniques were proposed including using honeypots to represent fake targets and mutating IP addresses to frequently change the ground truth of the network configuration (Jafarian et al., IEEE Transactions on Information Forensics and Security 10(12):2562–2577 (2015)), none of these deception techniques is resilient enough to provide high confidence of concealing the identity of the network assets, particularly against sophisticated attackers. In fact, in this chapter our analytical and experimental work showed that highly resilient cyber deception is unlikely attainable using a single technique, but it requires an optimal composition of various concealment techniques to maximize the deception utility. We, therefore, present a new cyber deception framework, called CONCEAL, which is a composition of mutation, anonymity, and diversity to maximize key deception objectives, namely concealability, detectability, and deterrence, while constraining the overall deployment cost. We formally define the CONCEAL metrics for concealability, detectability, and deterrence to measure the effectiveness of CONCEAL. Finally, we present the deployment of CONCEAL as a service to achieve manageability and cost-effectiveness by automatically generating the optimal deception proxy configuration based on existing host/network

The original version of this chapter was revised: Chapter authors have been added. The correction to this chapter is available at https://doi.org/10.1007/978-3-030-02110-8_12

Q. Duan (✉) · E. Al-Shaer · M. Islam
Software and Information Systems, University of North Carolina at Charlotte, Charlotte,
NC, USA
e-mail: qduan@uncc.edu

E. Al-Shaer et al. (eds.), *Autonomous Cyber Deception*,
https://doi.org/10.1007/978-3-030-02110-8_6

configuration, risk constraints of network services, and budget constraints. Our evaluation experiments measure both the deception effectiveness based on the above metrics and the scalability of the CONCEAL framework.

6.1 Introduction and Motivation

Cyber deception is an act of intentional misrepresentation of facts in order to induce an incorrect perception of reality in adversaries' mind. As a result, an adversary can be misled about the true configuration of the system such that their belief will be different from the reality. This false reality can assist in deflecting adversaries to a desired state of knowledge for invalidating their effort, slowing down their progress, and/or learning about their goals and techniques, even when adversary's techniques are unknown.

We believe that an effective cyber deception technique requires at least these five criteria: (1) It should provide a resilient *concealment* of the identity of cyber assets (at least the critical ones) including the real and fake (e.g., honeypots services), even if they are at some point discovered, (2) It should significantly increase the potential mistakes of the attackers (*detectability*), (3) It should significantly increase attackers' effort to achieve the target (*deterrence*); (4) It should provide automated configuration management that is highly transparent to users; (5) It should scalable to a large number of services and hosts, and (6) It can be tuned to provide cost-effective configurations based on mission risk and deception cost. Most of the existing deception techniques fail to satisfy these criteria, thereby they provide very limited deception effectiveness. In specific, honeypots configuration is static and can be easily detected and blacklisted by skilled attackers. In addition, while IP mutations are effective in slowing down reconnaissance attackers proactively, they cannot fully hide the identity of hosts and service against fingerprinting attackers. Moreover, deploying a large number of honeypots is expensive and unmanageable.

Since every deception technique has its own benefit and cost, and every asset may have its own risk based on exploitability and impact, it is important to find a composition of multiple deception techniques to achieve superlinear effectiveness than applying them individually, while satisfying the budget, risk, and operation constraints. In this chapter, we present a new deception framework called CONCEAL that composes m-mutation for address anonymization, k-anonymity for fingerprint anonymization, and l-diversity for configuration diversification in order to satisfy the above objectives. Here m-mutation means that for every 1/m seconds, the addresses of the hosts are mutated, k-anonymity means that for each network host a group of $k - 1$ hosts with identical fingerprints (called shadow services) are placed in the network, and l-diversity means that for every service type (e.g., web service), at least $l - 1$ fake (operationally unused) services of the same type but with different vendors, versions, or patch levels are placed in the network (e.g., Apache, IIS, etc. for web services).

To develop CONCEAL as a cloud service, the clients' requests will be directed to the CONCEAL gateway which will then translated and redirected based on the site configuration to the appropriate real or proxy (shadow or diversity service) hosts. To minimize the number of CONCEAL proxies, the gateway may redirect requests destined to many IP addresses to a single physical proxy machine. Therefore, we model the problem of finding satisfiable composition of mutation, anonymization, and diversity as a constraint satisfaction problem using satisfiability modulo theories (SMT) [8] to find the appropriate values of m, k, l, and the gateway configurations that minimize the cost of physical proxies for shadow and diverse services.

Our implementation and evaluation validate the effectiveness and scalability of the CONCEAL framework. The main tool we use for CONCEAL constraints formalization is SMT. SMT is a powerful tool to solve constraint satisfaction problems arise in many diverse areas including software and hardware verification, type inference, extended static checking, test-case generation, scheduling, planning, graph problems, etc. [8]. An SMT instance is a formula in first-order logic, where some function and predicate symbols have additional interpretations.

6.2 Threat Model and Objectives

In this section, we first present threat model addressed by our approach, as well as generic defense objective against this threat model.

6.2.1 Threat Model

Reconnaissance is the primary and initial step of any advanced and persistent intrusion on enterprise networks [9]. At the network and host levels, reconnaissance refers to the process of (1) discovering and enumerating network hosts, (2) scanning these hosts at network- and transport-level (detecting OS and open ports), or application-level (identifying services names), and (3) discovering exploitable vulnerabilities for each host [15].

If attackers know the name of a host, they can obtain its IP address from DNS and using the IP to attack that host. However, adversaries usually avoid this to remain stealthy, as their frequent request to DNS can be detected. In addition, only public servers can exist in the public DNS; however, internal services will still need to be discovered through local reconnaissance. Due to these reasons, we believe that sophisticated attackers will only rely on stealthy host/service scanning to discover resources and propagate their attacks. Examples of network reconnaissance tools to scanning and fingerprinting include `Nmap` and `Nessus` [15].

As attackers need to probe hosts to scan the address space and compromise services, benign users usually reach their remote services after querying the server's IP address from the corresponding authoritative DNS. Some attackers may also issue a reverse-DNS query to obtain the domain name from the host's address after discovering an active IP address. However, in both cases, this can be used for analyzing and detecting malicious behavior due to reconnaissance.

While random host IP mutation (RHM) [12] is effective for automated worms and naive adversaries, skilled human adversaries can still use any available information such as a host's fingerprint to re-identify that host even if its addresses have been mutated. This is because a host identity can still be recognized and traced by its fingerprint in RHM. This means a skilled attacker can fingerprint a host and use that fingerprint in future to distinguish that host. This is the major motivation to apply a more sophisticated deception approach in this chapter.

6.2.2 Defense Objectives

The defender has the following three ways to invalidate the attacker's knowledge gained from reconnaissance:

- **Address mutation** refers to the act of changing host addresses over time. With no *address mutation*, attacker can use a host IP address to identify it and avoid re-probing that host.
- **Fingerprint anonymization**: refers to the notion of hiding a host fingerprint in a pool of honeypots with identical fingerprints, thus preventing a skilled adversary to trace a host by its fingerprint. A skilled attacker can use the potentially unique fingerprint of a host to identify it later if host fingerprint anonymization is not applied.
- **Configuration diversification** refers to the act of diversifying fingerprints of network OS and services over time, thus making that any existing OSes or services will have a number of similar OSes and services in the same network.

6.3 Technical Approach

6.3.1 CONCEAL Framework Key Components

CONCEAL is a multi-strategy deception technique that combines three techniques, m-mutation, k-anonymization, and l-diversity.

Strategy 1: m-Mutation for address anonymization. For every $1/m$ seconds, CONCEAL mutates addresses of hosts. The goal of mutation is to anonymize host IP addresses over time. Note that mutating with rate m IP/Sec means an IP address is only active for $1/m$ seconds. In other words, IP addresses are anonymized every $1/m$ seconds. Mutation rate is a trade-off between benefit and cost (for address translation, updates, etc.). The limitation of mutation is that mutation is costly, and sometimes there is not enough available addresses. For a skilled attacker, host fingerprint may serve as a quasi-identifier, which means mutation may not be enough for deception.

Strategy 2: k-Anonymity for fingerprint anonymization. For each network host, CONCEAL places a group of $k-1$ hosts with identical fingerprints; the $k-1$ addresses are redirected to one physical honeypot representing a group of shadow hosts.

In data privacy, quasi-identifiers are anonymized by the concept of k-anonymity.

In our domain, k-anonymity is achieved by shadow hosts. That is, for each real host, $k-1$ honeypots are included in address space with same fingerprints. A shadow host exhibits an identical fingerprint of a real host.

By satisfying k-anonymity, we anonymize identity of an individual host. However the adversary can still figure out what platforms and services are running in the real network, and the number of real hosts and unique fingerprints in the network. Thus, attackers need to make exploits for these services only.

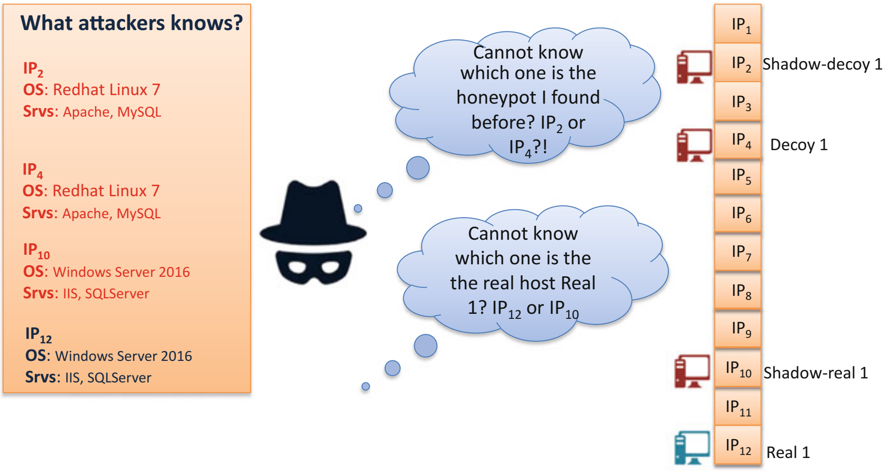


Fig. 6.1 An example of 2-anonymity

Figure 6.1 shows an example of 2-anonymity. In the system there is one real host and one proxy (decoy) host. For both of them there is one shadow host which has a different IP but identical configuration. The attacker cannot differentiate the shadow hosts from real or proxy host.

Strategy 3: l-Diversity for configuration anonymization. For every service type, CONCEAL places in the network at least $l-1$ fake services of the same type but with different vendors, versions, or patch levels. Here service types may include OS, HTTP Server, FTP Server, etc. Services of type HTTP may include IIS 8.0, IIS 7.0, Apache, etc.

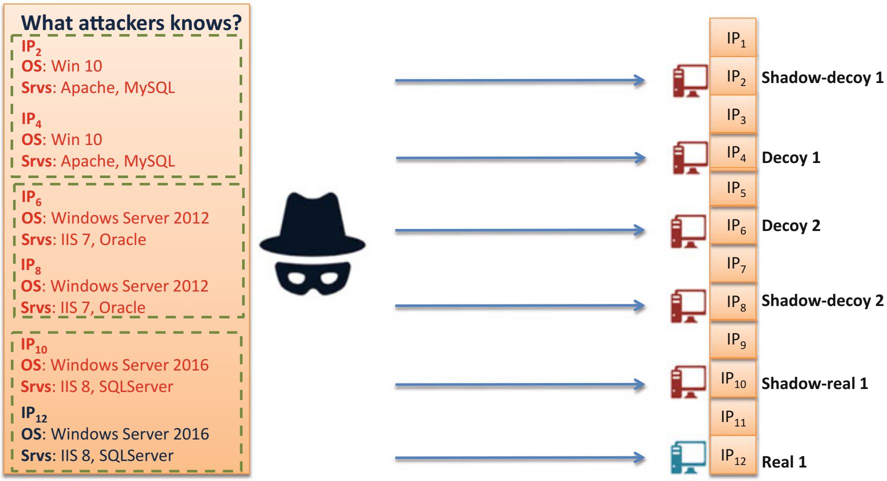


Fig. 6.2 An example of 2-anonymity, 3-diversity

Figure 6.2 shows an example of 2-anonymity, 3-diversity. In the example, the network has one real host and two proxy (decoy) hosts, while every real host and proxy host have an identical shadow host. Here we have 3-diversity for OS, HTTP service, and database service. 3-diversity for OS is achieved with Win Server 2016, Win Server 2012, and Win 10; for HTTP Services is achieved with IIS 7, IIS 8, and Apache; for database services is achieved with SQL Server, MySQL, and Oracle.

In order to find the satisfiable CONCEAL configuration, one needs to solve the following questions:

Question 1: What m, k, and l are good for a given network? How to quantify benefit and cost for a given (m, k, l) triple?

Question 2: Given m, k, and l, what are the optimal decoy services and configurations needed?

Note that these two questions are not independent. Both of them are related to budget and risk constraints. We will try to find the solution for both of them.

6.3.2 *CONCEAL Effectiveness Metrics*

We define the measurement for CONCEAL effectiveness (CE) as the combination of the following three metrics:

(1) Concealability measure (CM) that is the likelihood that the attacker will fail to identify the target despite his stealthiness.

(2) Detectability measure (DM) that is the likelihood that the attacker being detected despite his success.
(3) Deterrence measure (TM) that is the cost of deception on the adversary (e.g., number of probes).

CE is defined to be the combination of the three metrics:

$$CE = \alpha_1 \cdot CM + \alpha_2 \cdot DM + \alpha_3 \cdot TM \tag{6.1}$$

where α_1, α_2, and α_3 are appropriate coefficients which can be determined by different system security requirements.

As the risk of an asset increases, CE must also increase to offer cost-effective and scalable deception. In RHM, if the scanning speed (m_1) is the same as the mutation speed m, the probability that attackers fail to identify/compromise a host (CM) is the probability that the host is not identified by scanning due to mutation plus the probability that a proxy (not a real host) is hit:

$$CM = e^{-1} + (1 - e^{-1})(1 - \frac{1}{kl}) \tag{6.2}$$

The quantity e^{-1} results from invisible hosts due to mutation [11], and the quantity $(1 - \frac{1}{kl})$ results from proxy or shadow hosts due to k-anonymity and l-diversity.

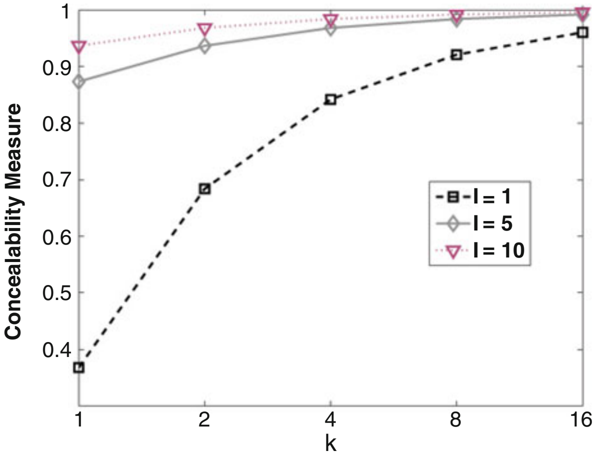


Fig. 6.3 Concealability measure

Figure 6.3 shows the concealability measure of CONCEAL with different k and l.

Deterrence (TM) is defined as the number of round of scan attempts to make the probability of hit more than the desired threshold. If one has only mutation, then the probability of hit (denoted as T) with n round of scans is known to be [11]:

$$T = 1 - (e^{-1})^n \tag{6.3}$$

For example, to guarantee $T > 0.9$ one needs to make $n > 2.3$ rounds of scanning.

In general T can be computed as:

$$T = 1 - (CM)^n \tag{6.4}$$

For example, when $k = 2$, $l = 4$, to guarantee $T > 0.9$, one needs to make $n > 22$ rounds of scanning.

Now we can calculate TM to be:

$$TM = log(1 - T_h)/log(CM) \tag{6.5}$$

where T_h is the required hitting threshold. Note that here we assume every round of scan is an independent event.

Figure 6.4 shows the T value of CONCEAL with different k, l, and T_h. We can see that for RHM the number of required scans (TM) is very small but even for CONCEAL with relative small k and l (such as 6) TM can be very high (more than 100). This means that the composition of mutation, diversity, and anonymity has much better effectiveness than applying only RHM.

Detectability (DM) is defined to be the probability that a compromise attempt is detected:

$$DM = (1 - \frac{1}{kl}) \cdot d_1 \cdot d_2 \tag{6.6}$$

Here $(1 - \frac{1}{kl})$ is probability of hitting a shadow or proxy host, d_1 is probability of detecting the attack attempt at a shadow or proxy host (IDS true positive), and d_2 is the quality/robustness of deception (which means $(1 - d_2)$ is the probability that the deception is detected by the attacker).

Figure 6.5 shows the detectability measure of CONCEAL with $d_1 = 1$, $l = 2$, and different d_2 and k.

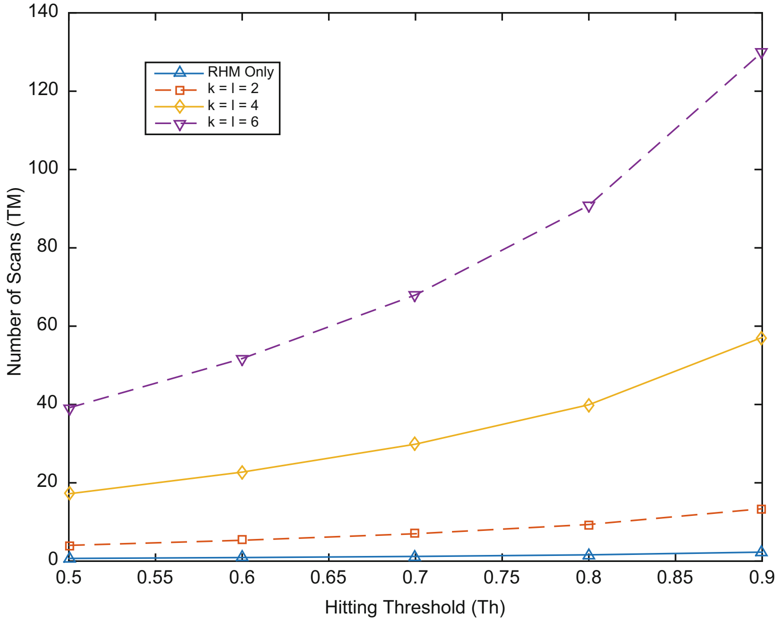


Fig. 6.4 Deterrence measure

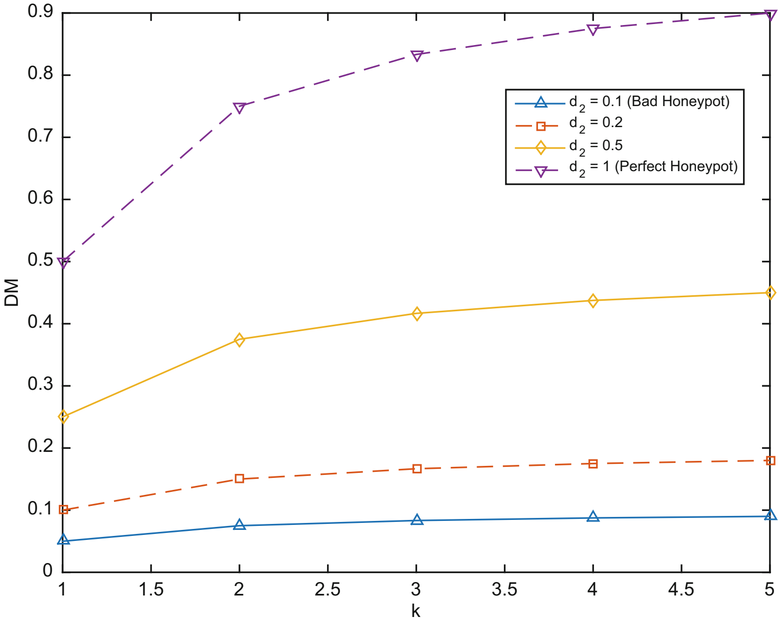


Fig. 6.5 Detectability measure

6.3.3 CONCEAL Architecture and Planner

Figure 6.6 shows the overall CONCEAL architecture. In CONCEAL architecture, clients communicate with the target network through the CONCEAL gateway of the target network, which is located in front of the honeypot cloud which includes real and proxy (shadow) hosts. The configuration of real and proxy (shadow) hosts is mutable and adjusted by the CONCEAL planner which determines the satisfiable configuration for mutation, anonymity, and diversity.

Figure 6.7 shows the framework of CONCEAL planner. Here the CONCEAL engine takes user input of available address space, existing host configuration (number of hosts, OS, service type, vulnerabilities), cost of proxies and shadows, budge limit, vulnerability impact, risk bound, and operational input, which includes feasible configuration, mission/business-related configuration (such as context and consistency constraints). These inputs are converted into CONCEAL constraints and sent to the SMT solver. If the problem is solvable, the results are returned to CONCEAL engine, which will generate the detailed host configurations. If the problem is not solvable, then one needs to relax some of the inputs such as increasing budget or decreasing risk bound.

The following is the formal definition of the CONCEAL planning problem. The input of the problem includes the configuration of n real hosts $C_1, \ldots, C_n$, where

$$C_i =< OS_i, s_{i,1}, \ldots, s_{i,k} > \tag{6.7}$$

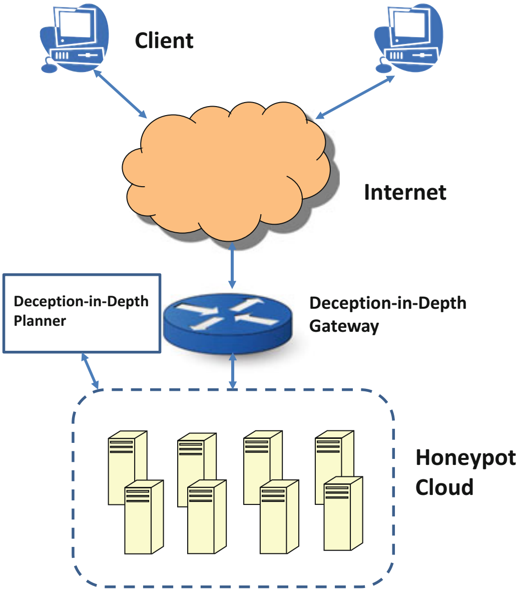


Fig. 6.6 CONCEAL architecture

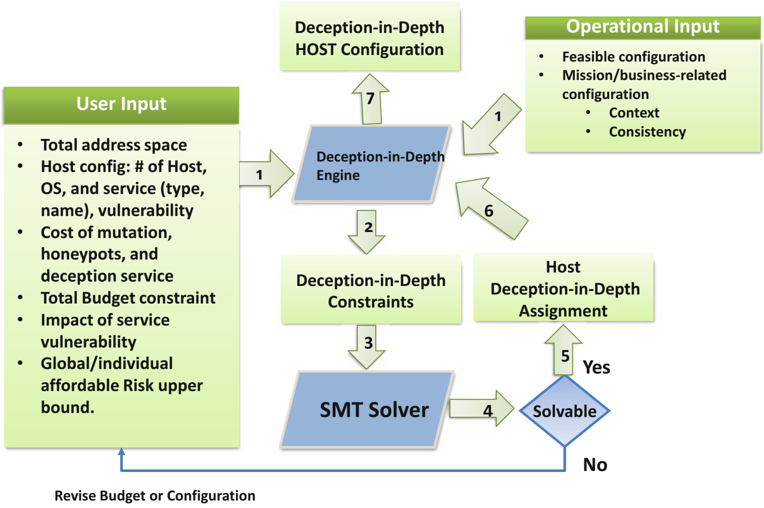

Deception-in-Depth HOST Configuration
Operational Input
Feasible configuration
Mission/business-related configuration
Context
Consistency
User Input
Total address space
Host config: # of Host, OS, and service (type, name), vulnerability
Cost of mutation, honeypots, and deception service
Total Budget constraint
Impact of service vulnerability
Global/individual affordable Risk upper bound.
7
1
1
Deception-in-Depth Engine
6
2
Deception-in-Depth Constraints
Host Deception-in-Depth Assignment
3
5
Yes
SMT Solver
4
Solvable
No
Revise Budget or Configuration

Fig. 6.7 CONCEAL planner framework

where $s_{i,j} \in T_j$, T_j is the service type, and set of vulnerabilities for every service, impact of vulnerabilities, cost of deploying mutation (C_m), proxy (C_p), diversity service (C_d), anonymity (shadow) service (C_a), budget (B), and risk bound (R). The output includes the appropriate m, k, l, number of required proxies (p), and configuration of the proxies.

6.3.4 Formalization of Constraints

We formalize the CONCEAL constraints as the input for SMT solver.

First, we have the *Budget constraint*

$$C_p \cdot p + C_d \cdot l + C_a \cdot k + C_m \cdot m \leq B \tag{6.8}$$

Next, we have the *Risk constraint*

$$\sum_i S_i \cdot (1 - CE) \cdot I_i \leq R \tag{6.9}$$

where S_i and I_i are the severity and impact of the vulnerability of service i, respectively.

Diversity constraint is formalized as

$$\forall i, \ dis(a_{ijk}) \geq l \tag{6.10}$$

where a_{ijk} is the binary variable that denotes that service j with type i is deployed in proxy k, and $dis(a_{ijk})$ denotes the number of distinct services of type i in all possible proxies.

We also have the *unique assignment constraint*

$$\forall i, k, \ \sum_j a_{ijk} \le 1 \tag{6.11}$$

which means that every proxy can only contain one service of a fixed type.

The *context constraint* can be formalized as

$$\forall i, j, v, k, \ \ (b_{vk} = 1) \rightarrow (a_{ijk} \ne 1) \tag{6.12}$$

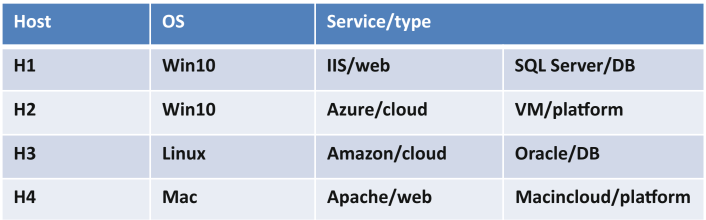
which means that OS type v should not contain service j of type i, where b_{vk} denotes that OS type v is deployed in proxy j. For example, OS Linux should not have any type of IIS services.

The *consistency constraint* can be formalized as

$$\forall k, \ \ (a_{i_1 j_1 k} + a_{i_2 j_2 k}) \le 1 \tag{6.13}$$

where j_1 and j_2 are incompatible or inconsistent services to co-exist in the same proxy.

As an example, suppose we have four real hosts in the system as shown in Fig. 6.8. The cost values of C_p, C_d, C_a, C_m are 4K, 2K, 1K, 20K, respectively, and budget limit B is 28K, risk bound R is 8K, and $m_1 = 0.2$. We also have context constraint that any SQL server cannot be deployed in Linux OS.



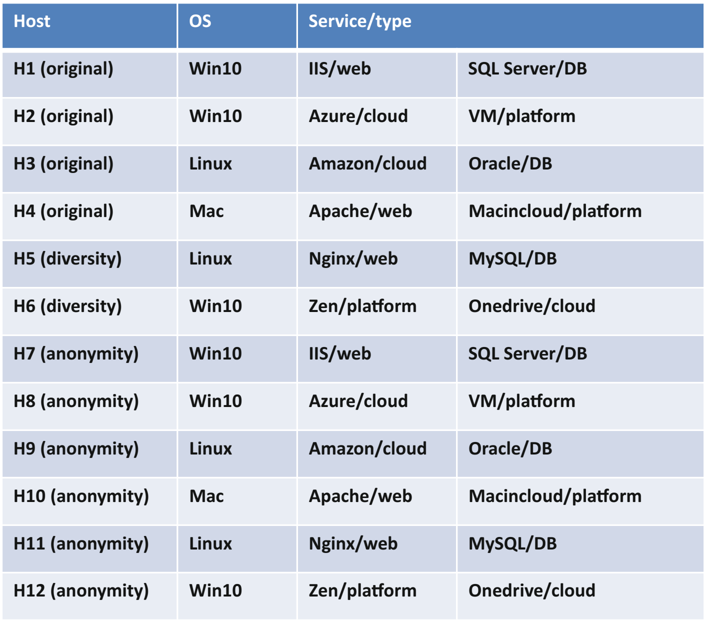
Host	OS	Service/type	
H1	Win10	IIS/web	SQL Server/DB
H2	Win10	Azure/cloud	VM/platform
H3	Linux	Amazon/cloud	Oracle/DB
H4	Mac	Apache/web	Macincloud/platform

Fig. 6.8 Original host configuration

The SMT solver returns that $l = 3$, $k = 2$, $m = 0.2$. The solution has two proxy host H_5 with Linux OS, web service Nginx, and DB service MySQL, and H_6 with Win 10 OS, platform service Zen, and cloud service Onedrive to achieve 3-diversity (for simplicity, we only consider service diversity here). Additional shadow hosts H_7, H_8, H_9 ,H_{10}, H_{11}, H_{12} need to be created for hosts H_1, H_2, H_3, H_4, H_5, H_6 to achieve 2-anonymity. Total cost is

$$(4 * 2 + 4 * 2 + 6 * 1 + 0.2 * 20)K = 26K \ < B = 28K$$

Figure 6.9 shows details of the solution.



Host	OS	Service/type	
H1 (original)	Win10	IIS/web	SQL Server/DB
H2 (original)	Win10	Azure/cloud	VM/platform
H3 (original)	Linux	Amazon/cloud	Oracle/DB
H4 (original)	Mac	Apache/web	Macincloud/platform
H5 (diversity)	Linux	Nginx/web	MySQL/DB
H6 (diversity)	Win10	Zen/platform	Onedrive/cloud
H7 (anonymity)	Win10	IIS/web	SQL Server/DB
H8 (anonymity)	Win10	Azure/cloud	VM/platform
H9 (anonymity)	Linux	Amazon/cloud	Oracle/DB
H10 (anonymity)	Mac	Apache/web	Macincloud/platform
H11 (anonymity)	Linux	Nginx/web	MySQL/DB
H12 (anonymity)	Win10	Zen/platform	Onedrive/cloud

Fig. 6.9 SMT solution of CONCEAL hosts

6.4 Evaluation

We evaluate the effectiveness and scalability of the CONCEAL framework. The evaluation work is done in a machine of Intel Quad Core 3.4GHz machine with 16G memory. We use the Yices SMT solver [2] to find the solutions to the constraints.

CONCEAL Effectiveness As we discussed, CE is the metric to measure the effectiveness of CONCEAL. Figures 6.10, 6.11, and 6.12 show the concealability measurement (CM), detectability measurement (DM), and deterrence measurement (TM) values for different number of services and budget bound B, respectively. For the TM value, the threshold T_h is set to be 0.9. For the DM value, d_1 and d_2 are set to be 0.95. We can see that all the CM, DM, and TM values increase with high values of B and high number of services. However the increase of B has less significant impact when the number of services reaches some point.

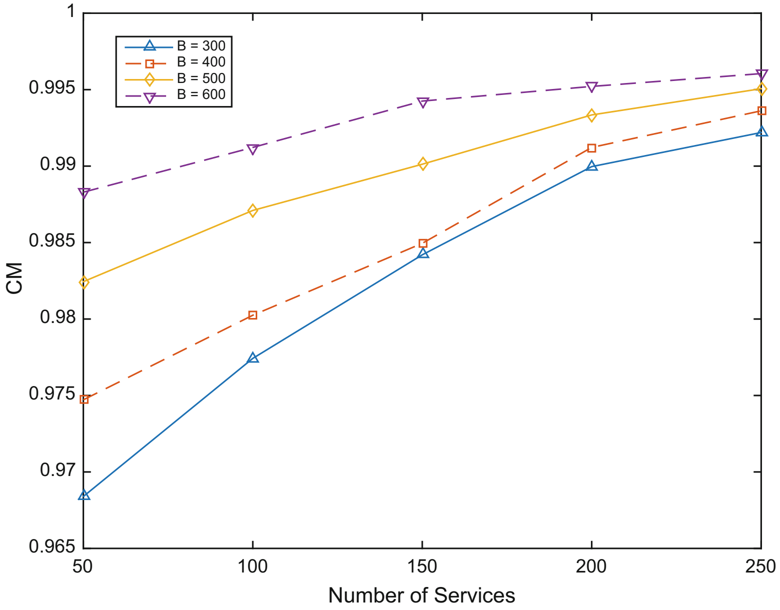


Fig. 6.10 Concealability measurement

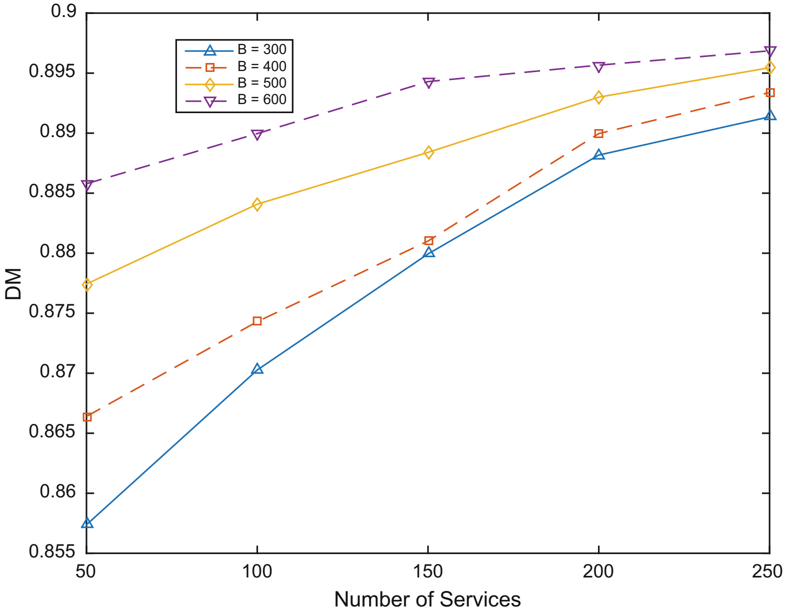


Fig. 6.11 Detectability measurement

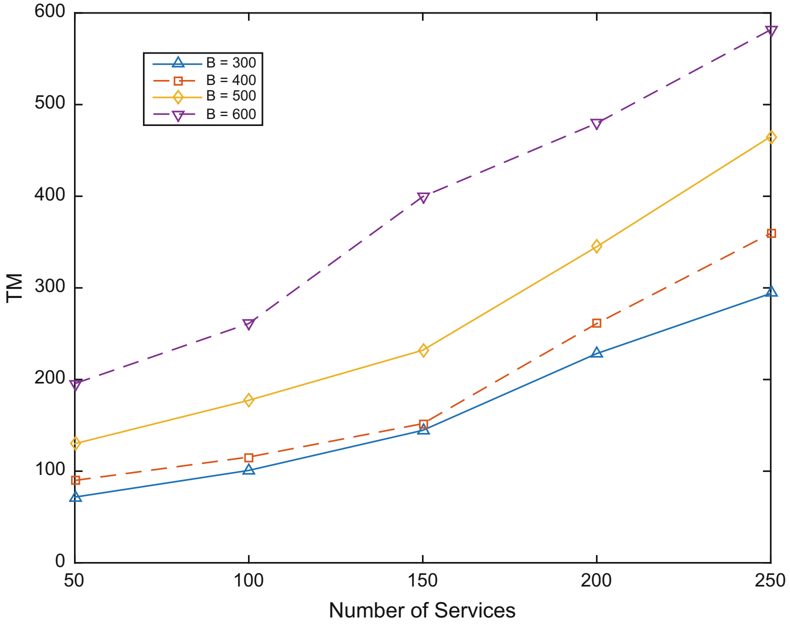


Fig. 6.12 Deterrence measurement

Saving of Proxies for Different *B* The CONCEAL framework can save the number of actual used proxies since multiple services can share the same proxy as long as they are compatible. Figure 6.13 shows the percentage of saved proxies with different B. We can see that as B decreases the percentage of saved proxies will increase. This is because smaller B will cause the SMT solver to find smaller value of p to reduce cost. However, smaller B may make the problem unsolvable and increase the solving time.

Saving of Proxies for Different Number of Correlation Figure 6.14 shows the percentage of saved proxies with different number of correlations. The X-axis cor in the figure is the percentage of consistent services where consistent means the services cannot exist in the same host. We can see that as cor increases the percentage of saved proxies will decrease. This is because higher cor will cause the SMT solver to use higher value of p to satisfy the constraints.

Solving Time for Different *B* Figure 6.15 shows the SMT solving time for different B. We can see that as B increases the solving time will decrease. This is because higher B means more resources and it will be easier for the SMT solver to find the solution to satisfy the constraints.

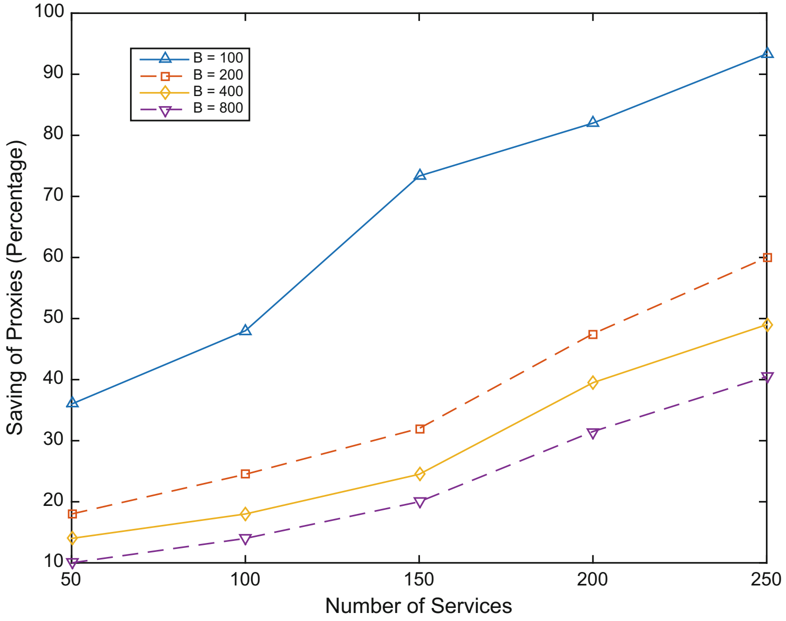


Fig. 6.13 Save of proxies with different *B*

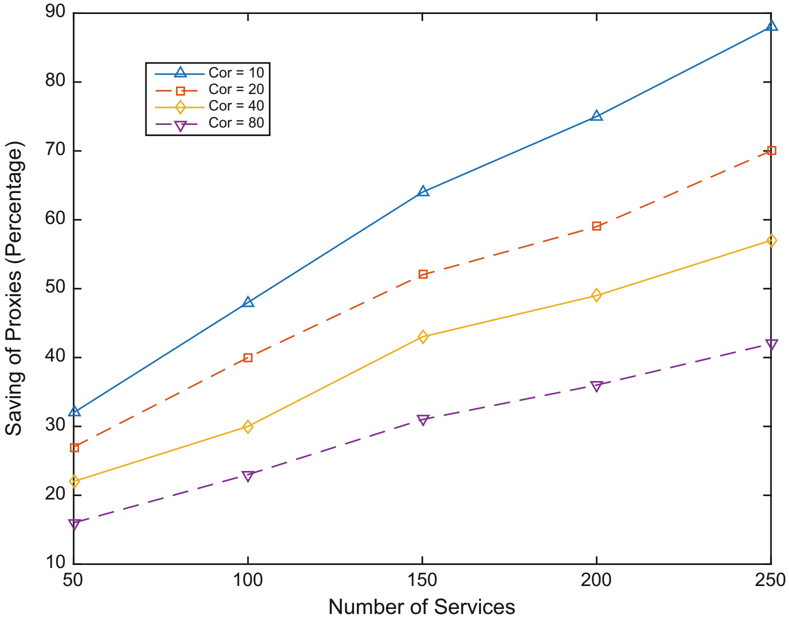


Fig. 6.14 Save of proxies with different correlation

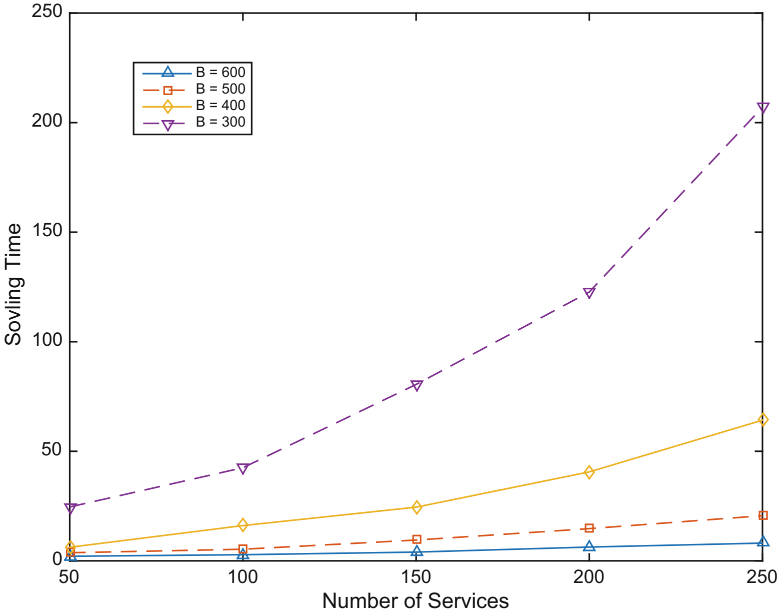


Fig. 6.15 Solving time with different *B*

Solving Time for Different Number of Correlation Figure 6.16 shows the SMT solving time for different number of correlations *cor*. We can see that as *cor* increases the solving time will increase. This is because higher *cor* means more constraints for the SMT solver to find the satisfiable solution.

6.5 Implementation

We implemented CONCEAL as a deception of service that can be accessible through a web interface. The CONCEAL implementation mainly consists of three different components: (1) The web interface, (2) SMT Solver, and (3) ActiveSDN; shown in Fig. 6.17. Each of these three services running in different process can communicate with each other through rest APIs. Although the API descriptions are out of the scope of this chapter, however, interested reader can look into our implementations [1].

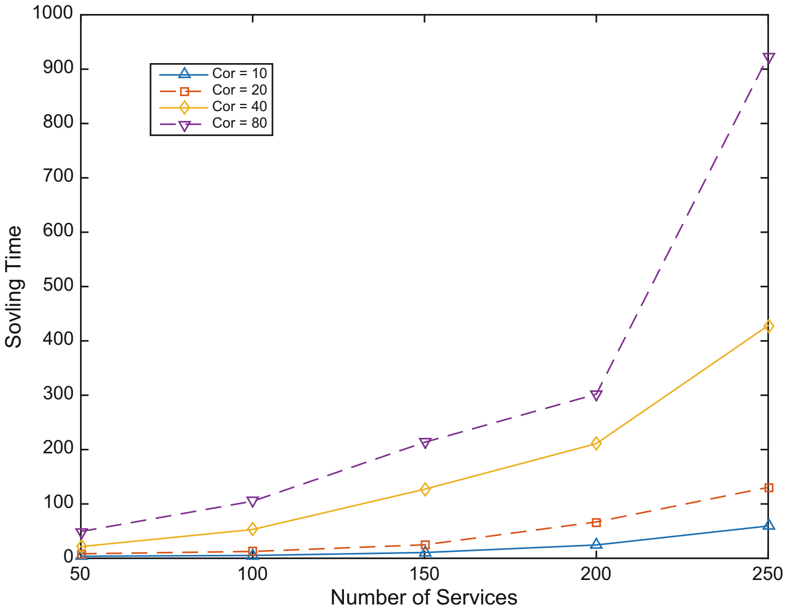


Fig. 6.16 Solving time with different correlation

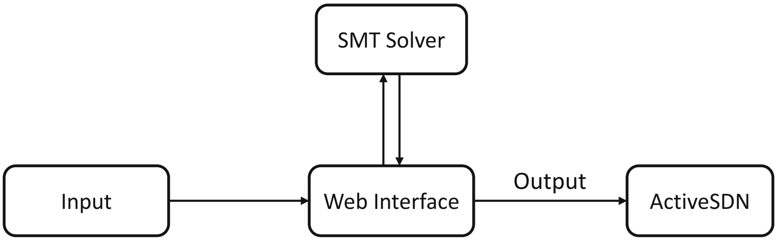
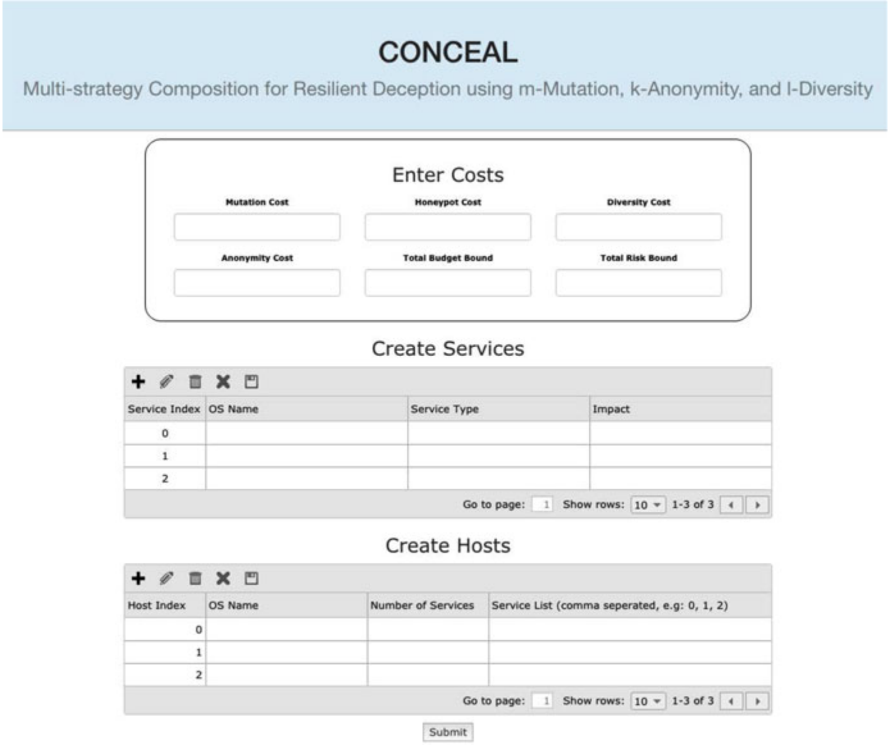


Fig. 6.17 CONCEAL implementation

6.5.1 Web Interface

The web interface is divided into three sections: (1) costs, (2) services, and (3) hosts. The user will provide all of the costs like mutation, anonymity, diversity, and proxy costs along with total risk and budget costs. For the *services*, user can create as many services as she wants and those services will be added in the *hosts* section. After submitting, the constraints will be solved by the SMT solver which is running in a different process and if there exist any solution it will be shown as output, else user need to relax the constrains and submit again. Figure 6.18 shows the web interface.



CONCEAL

Multi-strategy Composition for Resilient Deception using m-Mutation, k-Anonymity, and l-Diversity

Enter Costs

Mutation Cost	Honeypot Cost	Diversity Cost
Anonymity Cost	Total Budget Bound	Total Risk Bound

Create Services

Service Index	OS Name	Service Type	Impact
0			
1			
2			

Create Hosts

Host Index	OS Name	Number of Services	Service List (comma seperated, e.g: 0, 1, 2)
0			
1			
2			

Fig. 6.18 CONCEAL web interface

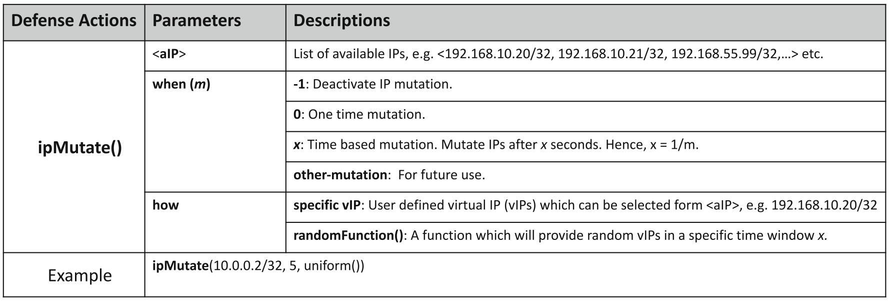
6.5.2 *SMT Solver*

We implement our constraint satisfaction solver using Yices [2] in C++. All the constraints mentioned in the Sect. 6.3.3 are implemented in our solver. The solver is running in a different process that can be accessible through a web interface.

6.5.3 *ActiveSDN*

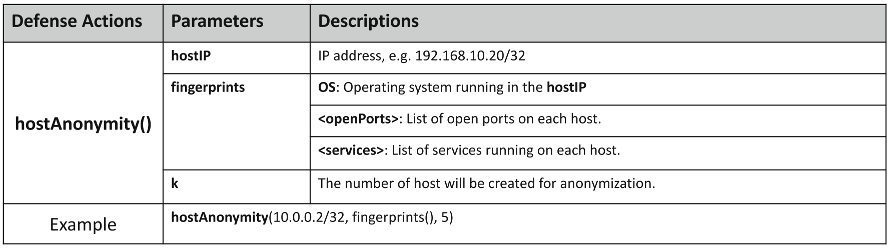
ActiveSDN is a decision-making OpenDaylight controller [16] where we implement all the services required for CONCEAL, like mutation, anonymization, and shadowing. After submitting the user inputs through web interface, the SMT solver will generate the output configuration, which will be used in ActiveSDN to deploy conceal in a *Mininet* network [21]. For conceal, we have three services implemented in ActiveSDN: *IP mutation*, *host anonymization*, *service diversity*.

Figure 6.19, 6.20, and 6.21 show the IP mutation, host anonymization, and service diversity functions in ActiveSDN, respectively. For IP mutation, *aIP* indicates the list of available IP addresses that will be mutated using *vIPs* mentioned in *how*. The mutation parameter *when* has an attribute *x* which indicates the mutation cycle (mutate *aIPs* after *x* seconds). The host anonymity function takes the IP address of a host as an argument *hostIP* and its corresponding *fingerprints*. For the *hostIP*, *k* new host with identical fingerprints will be created. The service diversity function architecture is similar to host anonymity function.



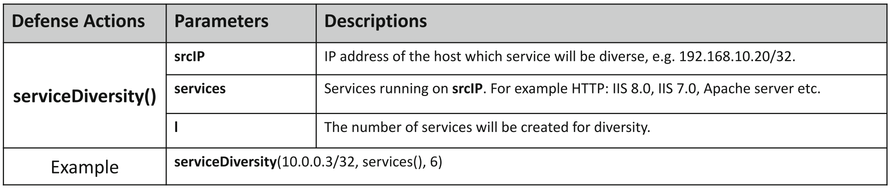
Defense Actions	Parameters	Descriptions
ipMutate()	**<aIP>**	List of available IPs, e.g. <192.168.10.20/32, 192.168.10.21/32, 192.168.55.99/32,…> etc.
	when (*m*)	**-1**: Deactivate IP mutation.
		0: One time mutation.
		x: Time based mutation. Mutate IPs after *x* seconds. Hence, x = 1/m.
		other-mutation: For future use.
	how	**specific vIP**: User defined virtual IP (vIPs) which can be selected form <aIP>, e.g. 192.168.10.20/32
		randomFunction(): A function which will provide random vIPs in a specific time window *x*.
Example	**ipMutate**(10.0.0.2/32, 5, uniform())	

Fig. 6.19 IP mutation function



Defense Actions	Parameters	Descriptions
hostAnonymity()	**hostIP**	IP address, e.g. 192.168.10.20/32
	fingerprints	**OS**: Operating system running in the **hostIP**
		<openPorts>: List of open ports on each host.
		<services>: List of services running on each host.
	k	The number of host will be created for anonymization.
Example	**hostAnonymity**(10.0.0.2/32, fingerprints(), 5)	

Fig. 6.20 Host anonymization function



Defense Actions	Parameters	Descriptions
serviceDiversity()	**srcIP**	IP address of the host which service will be diverse, e.g. 192.168.10.20/32.
	services	Services running on **srcIP**. For example HTTP: IIS 8.0, IIS 7.0, Apache server etc.
	l	The number of services will be created for diversity.
Example	**serviceDiversity**(10.0.0.3/32, services(), 6)	

Fig. 6.21 Service diversity function

6.6 Related Works

Proactive techniques for disrupting reconnaissance could be broadly divided into two categories: MTD-based and deception-based approaches. Moving target defense relies on randomizing static system parameters, to invalidate attacker's reconnaissance information.

In this direction, several approaches have been proposed [11–13] for mutation of IP addresses, especially over time. NASR [6] relies on DHCP to randomize IP addresses over time. RHM [12] and OF-RHM [11] propose using DNS for address randomization without changing the actual IP address of network hosts, in order to prevent TCP/UDP sessions from being broken. RHM [12] identifies additional need for mutating MAC addresses to defeat LAN-level reconnaissance, and mutating domain names to counter reverse-DNS queries.

The notion of mutable networks as a frequently randomized changing of network addresses and responses was initially proposed in [3]. The idea was later extended as part of the MUTE network which implemented the moving target through random address hopping and random fingerprinting [4].

Existing IP mutation techniques include dynamic network address translation (DYNAT) [14, 17, 18], random host IP mutation (RHM) [12], OpenFlow random host IP mutation (OF-RHM) [11], etc.

DYNAT is a technique developed to dynamically reassign IP addresses to confuse any would-be adversaries sniffing the network. They obfuscate the host identity information (IP and Port) in TCP/IP packet headers by translating the identity information with preestablished keys. Both BBN [14] and Sandia [17, 18] have done research work on DYNAT. BBN ran series of red-team tests to test the effectiveness of DYNAT, while Sandia's DYNAT report [17, 18] examines many of the practical issues for DYNAT deployment.

RHM [12] and OF-RHM [11] use DNS for address randomization without changing the actual IP address of network hosts and random mutate host IPs while satisfying mission, operational, and cost constraints.

While these works are effective against automated scanners and worms, in our evaluation we show that they have limited effectiveness against human attackers.

On the other hand, *deception-based technologies*, trying to attract attackers to decoy hosts such as honeypots, can mislead the attacker and prolong reconnaissance and attack progress. The work in [19] shows that honeypots are highly effective in slowing down worm breakouts, and decreasing amount of attacks on production hosts.

Dynamic honeypots were first proposed by Budiarto [7]. Dynamic honeypots discover production systems on a network and use this information to create honeypots that are similar to production hosts and mix in the surrounding network. *Shadow* honeypots were proposed by Anagnostakis et al. [5], which is an identical copy of production hosts, and used as an strategy to deploy honeypots in a production environment.

The work in [20] correctly identifies the need of strengthening address mutation with decoy services against sophisticated adversaries, but their approach neglects important theoretical aspects of anonymization such as mutation and anonymization of host/honeypot fingerprints. Moreover, efficacy of their proposed model is not tested against real human attackers.

None of the above approaches recognizes the need and synergy of combining the multi-dimensional proactive defense for defeating adversarial reconnaissance performed by skilled adversaries.

Our preliminary work of proactive multi-dimensional deception technique in [10] tries defeating reconnaissance by skilled attackers, through mutating configuration (name, addresses, fingerprint) of network hosts, while populating address space with moving honeypots with strategically designed but randomly changing configurations. However it does not define the clear metric for multi-dimensional deception effectiveness and provide no synthesis to find the satisfiable combination of the multi-dimensional techniques.

6.7 Conclusion

In this chapter we present a new deception as a service paradigm called CONCEAL that combines m-mutation for address anonymization, k-anonymity for fingerprint anonymization, and l-diversity for configuration diversification. We define three CONCEAL metrics such as concealability, deterrence, and detectability to measure the effectiveness of CONCEAL, and develop the framework to automatically generate the optimal CONCEAL configuration that satisfies related budget, risk, and operation constraints. We evaluated the effectiveness and scalability of the CONCEAL framework, and implemented CONCEAL as a deception of service based on ActiveSDN. Our implementation and evaluation validates the effectiveness and scalability of the CONCEAL framework. The CONCEAL framework can solve problem instances up to 250 services and the save of proxies can reach as high as 90%.

6.8 Exercise Problems

1. An SMT exercise—Download Z3 solver and create the appropriate SMT constraints to solve the following constraint satisfaction problem:

   Three connected chairs in a row (C1, C2, C3). We need to place aunt (180 lb), sister (120 lb), and father (200 lb) such that

   (a) Aunt doesn't want to sit near father,
   (b) Aunt doesn't want to sit in the left chair,
   (c) Sister doesn't want to sit to the right of father,
   (d) Any two adjacent chairs should not carry more than 350 lb.

2. Given CONCEAL parameters $(m, k, l) = (0.1, 3, 4)$, and $d_1 = d_2 = 0.8$, $T_h = 0.9$, $m_1 = 0.1$, calculate the following CONCEAL effectiveness metrics (1) CM; (2) DM; (3) TM.
3. In the example shown in Fig. 6.3, we can see that if k or l is greater than some threshold values, the CM value only increases marginally. Given $l = 5$, find the threshold for k such that doubling the value of k will only improve CM by less than 10%.
4. In the example shown in Fig. 6.8, is there any other satisfiable values of (m, k, l) for CONCEAL planning problem (Hint: Use SMT to encode CONCEAL planning constraints and add an additional constraint to exclude the obtained solution in the chapter)?
5. In the example shown in Fig. 6.8, we set $\alpha_1 = 1$, and $\alpha_2 = \alpha_3 = 0$, this means that we only consider CM for the effectiveness. If we set $\alpha_1 = 1$, $\alpha_2 = 0.01$, $\alpha_3 = 0$ and $T_h = 0.9$, and other values remain the same, is the problem be still solvable?

Acknowledgements This research was supported in part by United States Army Research Office under contract number W911NF1510361. Any opinions, findings, conclusions or recommendations stated in this material are those of the authors and do not necessarily reflect the views of the funding sources.

References

1. (2018) https://github.com/rakeb
2. (2018) Yices: An SMT solver. http://yices.csl.sri.com/
3. Al-Shaer E (2009) Mutable networks, National cyber leap year summit 2009 participants ideas report. Tech. rep., Networking and Information Technology Research and Development (NTIRD)
4. Al-Shaer E (2011) Toward network configuration randomization for moving target defense. In: Jajodia S, Ghosh AK, Swarup V, Wang C, Wang XS (eds) Moving Target Defense, Advances in Information Security, vol 54, Springer New York, pp 153–159
5. Anagnostakis KG, Sidiroglou S, Akritidis P, Xinidis K, Markatos EP, Keromytis AD (2005) Detecting targeted attacks using shadow honeypots. In: Usenix Security
6. Antonatos S, Akritidis P, Markatos EP, Anagnostakis KG (2007) Defending against hitlist worms using network address space randomization. Comput Netw 51(12):3471–3490, DOI http://dx.doi.org/10.1016/j.comnet.2007.02.006
7. Budiarto R, Samsudin A, Heong CW, Noori S (2004) Honeypots: why we need a dynamics honeypots? In: Information and Communication Technologies: From Theory to Applications, 2004. Proceedings. 2004 International Conference on, IEEE, pp 565–566
8. Davis M, Putnam H (1960) A computing procedure for quantification theory. J ACM 7:201–215, DOI http://doi.acm.org/10.1145/321033.321034, URL http://doi.acm.org/10.1145/321033.321034
9. Hutchins EM, Cloppert MJ, Amin RM (2011) Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. Leading Issues in Information Warfare & Security Research 1:80

10. Jafarian H, Niakanlahiji A, Al-Shaer E, Duan Q (2016) Multi-dimensional host identity anonymization for defeating skilled attackers. In: Proceedings of the 2016 ACM Workshop on Moving Target Defense, ACM, New York, NY, USA, MTD '16, pp 47–58, DOI 10.1145/2995272.2995278, URL http://doi.acm.org/10.1145/2995272.2995278
11. Jafarian JH, Al-Shaer E, Duan Q (2012) Openflow random host mutation: transparent moving target defense using software defined networking. In: Proceedings of the first workshop on Hot topics in software defined networks, ACM, pp 127–132
12. Jafarian JH, Al-Shaer E, Duan Q (2015) An effective address mutation approach for disrupting reconnaissance attacks. IEEE Transactions on Information Forensics and Security 10(12):2562–2577
13. Jafarian JHH, Al-Shaer E, Duan Q (2014) Spatio-temporal address mutation for proactive cyber agility against sophisticated attackers. In: Proceedings of the First ACM Workshop on Moving Target Defense, ACM, MTD '14, pp 69–78
14. Kewley D, Fink R, Lowry J, Dean M (2001) Dynamic approaches to thwart adversary intelligence gathering. DARPA Information Survivability Conference and Exposition 1:0176, DI http://doi.ieeecomputersociety.org/10.1109/DISCEX.2001.932214
15. McClure S, Scambray J, Kurtz G, Kurtz (2005) Hacking exposed: network security secrets and solutions, vol 6. McGraw-Hill/Osborne New York
16. Medved J, Varga R, Tkacik A, Gray K (2014) Opendaylight: Towards a model-driven SDN controller architecture. In: World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a, IEEE, pp 1–6
17. Michalski J, Price C, Stanton E, Lee E, Seah CK, TAN YH, Pheng C (2002) Final report for the network security mechanisms utilizing network address translation LDRD project. technical report sand2002-3613. Tech. rep., Sandia National Laboratories
18. Michalski JT (2006) Network security mechanisms utilising network address translation. International Journal of Critical Infrastructures 2(1):10–49
19. Rowe NC, Custy EJ, Duong BT (2007) Defending cyberspace with fake honeypots. Journal of Computers 2(2):25–36
20. Sun J, Sun K (2016) DESIR: Decoy-enhanced seamless IP randomization. In: INFOCOM 2016
21. Team M (2012) Mininet: An instant virtual network on your laptop (or other pc)