CSC 790 A: Project 1
Deception Game v1.0

Due
12:30pm Thursday, 2/20

Project Description

As discussed in class, deception conceals or obscures an entity's existence or attributes so to intentionally mislead adversaries. Several deception methods for computer systems have been developed and researched; examples include address hopping/shuffling, honeypots, and honeynets. One aspect of deception that has not been fully considered is the deployment and subsequent management of these methods. For example given a computer network to protect, what type of deception is best? How often should address shuffling be performed? What is the minimum number of honeypots required to be effective? Will address shuffling and honeypots in combination provide better security? For this project you will develop a mathematical model to analyze the performance of deception to better understand deployment and management. In other words, the best defense strategies for a set of defense and attacker options.

Network and Attacker Description

Assume you are the administrator of a computer network consisting of vulnerable computers (like Fletcher's), secure computers, and empty addresses. The attacker seeks to map your network and find the vulnerable computers using a *scan* which may or may not be followed by an *exploit attempt*. Instead of using specific amounts for the number addresses, number of computer, etc... use variables. We're creating models after all.

- There are n total addresses available to the administrator.
- There are m computers in the network, of which v are vulnerable; therefore $v \leq m \leq n$.
- The attacker is aware of the address space n and seeks to find at least α percent of the v vulnerable computers; therefore, $0 < \alpha \leq 1$.
- The attacker will try k scans of the network to find vulnerable systems.

Available Deception Techniques

Several deception techniques are possible, however this assignment will only consider address shuffling and honeypots. Furthermore to help analysis you can ignore some implementation details however you will still need to consider the revenue and costs (administrative and attacker) of each technique. Revenue and costs are difficult to quantify. Identifying the cost of deployment is possible, although there are fixed costs, variable costs, ... *What is the administrative revenue associated with a successful defense? What is administrative the cost of a failed defense?* One possible solution is to consider utility instead of actual monetary amounts.

Address Shuffling

Address shuffling (or hopping, I can't decide) simply replaces a system's IP address with another. Once this occurs attacker reconnaissance information becomes invalid. An important administrative cost to consider is the loss of legitimate connections.

Honeypots

Honeypots are systems designed to detect unauthorized access. The system mimics an actual computer server and can record the source of an attacker (IP address) and type of exploit. As a result attackers seek to avoid these systems (so there is a high attacker cost). There is also a nontrivial administrative cost associated deploying and managing honeypots.

Models and Games

As previously described, you will develop mathematical models of defense strategies for a computer network. Such models are very useful for predicting the performance of a system that cannot be easily tested empirically. Verifying the model with empirical results an important, and necessary step, but will not be done for this assignment.

What type of model should you consider? We are interested in the likelihood of attacker success and given the amount of chance associated with a scan, probabilistic models are probably best suited. For example the NASR paper modeled the probability of attacker success as $\frac{v}{n}$, which is the probability that attacker will randomly scan one of the v vulnerable machines[1]. This is a start, but this simple equation does not take into account k scans are probably performed. Specifically each scan can provide the attacker knowledge (for example, whether or not to attempt the address again). If no defense is provided and the attacker can attempt n scans, then the probability that the attacker will find v vulnerable computers is 1. The NASR model does not directly provide this insight.

Developing the probability equations for different scenarios is possible, but perhaps a better approach would associate another known model with the network system. *Can the system be modeled as a dice game?* Selecting a computer to scan (attackers point of view) could be modeled as rolling a n sided die, where each side represents an address. Consider the implication of shuffling and how that would map to a dice game. *Can the system be modeled as a card game, a roulette wheel, keno game*[2]*, ...?* Using these known models an equation for the attacker success probability can be determined, *I think.* I hope you will select one model for all the defense strategies.

Applying a Bad Introduction to Game Theory

This project is interested in determining the best defense strategy given a set of attacker strategies. As such, game theory is best suited for determining the best strategies if they are described mathematically (the point of the previous section). You will consider the system as a two player (administrator and attacker), non-cooperative, competitive game.

Player Actions, Payoffs, and Best Strategies

Using game theory you must define the actions (strategies) available to each player. For example depending on the question asked in the assignment, the administrator could have the following strategies: do nothing, address shuffling, or honeypot. The attacker may have the option to do nothing, scan, or scan and attempt exploit. Using this information a *payoff* matrix can be developed where the matrix entries are the payoff values for the defender and attacker. An example payoff matrix is given below.

Admin / Attacker	Static (no deception)	Shuffle	Honeypot
Scan	(E_s^d, E_s^a)	(E_{sh}^d, E_s^a)	(E_h^d, E_s^a)
Scan & Attack	(E_s^d, E_a^a)	(E_{sh}^d, E_a^a)	(E_h^d, E_a^a)

The values E_j^i is the expected payoff for player i using strategy j, for example E_s^d is the expected payoff for the defender using static addressing. This value is dependent on the attacker success probability and can be calculated as

$$E_s^d = (1 - \beta_s) \cdot r_s - \beta_s \cdot c_s$$

where β_s is the probability the attacker is successful given a static defense, r_s is the revenue of the defense, and c_s is the cost of the defense. The remaining expected payoffs can be determined in a similar fashion. Using the completed matrix the best strategies (dominate, Nash-equilbria) can be found, *I hope.*

[1] In all fairness, the NASR paper defended against hitlist attacks, so there were no scans and as a result their simple model is correct. But how was the list acquired?

[2] Maybe not keno, as according to Cody, you never lose in keno.

Project Requirements

The deliverable for this project will be a report describing your model (probabilistic and game) and your answers to the following sections. Your report must be written succinctly and neatly (yes, typeset).

1 Static Addressing

Develop a model to analyze static addressing which is equivalent to no defense. An important part of the model will be the probability of attacker success (β) for finding at least α percent of v vulnerable computers in a network of n addresses and m computers total given k attacker scans. Yes, a lot of variables to consider.

Questions (*15 points*)

You must correctly answer the following questions to receive full credit for this part of the project.

1. What is the equation for attacker success (β) for static addressing? Describe the equation and how it correctly models the system.

2. Assume the attacker only needs to find one vulnerable machine. Graph the attacker success rate as the number of scans (k) increases given a \16 (class B) network, where 25% of the addresses have machines and 10% of these machines are vulnerable. Comment on your results.

3. Provide a 3-D graph of the attacker success rate (z-axis) as the number of scans (k, x-axis) and attacker find percentage increases (α, y-axis). Assume a \16 (class B) network, where 25% of the addresses have machines and 10% of these machines are vulnerable. Comment on your results.

2 Address Shuffling

Develop a model for address shuffling where all n addresses are shuffled. An important part of the model will be the probability of attacker success (β) for finding α percent of v vulnerable computers. Again, the a network consists of n addresses and m computers, and the attacker can perform k scans. In addition your model should consider the frequency of shuffling denoted as γ.

Shuffling frequency does imply a temporal component to the model, which is a non-trivial addition. To simplify your initial analysis, you can consider the frequency to be associated with the scan attempts instead of over fixed time intervals (although you can use intervals if you like, *it might be easier*). Therefore $0 \leq \gamma \leq 1$ where $\gamma = 0$ is static addressing and $\gamma = 1$ is *perfect shuffling* which means the addresses are shuffled after each scan attempt. Therefore with *perfect shuffling* given k scans, k shuffles would occur which provides the best shuffle defense but has a very high cost.

Questions (*25 points*)

You must also correctly answer the following questions to receive full credit for this part of the project.

1. What is the equation for attacker success (β) for perfect address shuffling? Describe the equation and how it correctly models the system.

2. Assume the attacker only needs to find one vulnerable machine and the system is defended with perfect shuffling. Graph the attacker success rate as the number of scans (k) increases given a \16 (class B) network, where 25% of the addresses have machines and 10% of these machines are vulnerable. Compare with static addressing and comment on your results.

3. What is the equation for attacker success (β) for shuffling with frequency γ? Describe the equation and how it correctly models the system.

4. Provide a 3-D graph of the attacker success rate (z-axis) as the number of scans (k, x-axis) and shuffling frequency increases (γ, y-axis). Assume a \16 (class B) network, where 25% of the addresses have machines, 10% of these machines are vulnerable, and the attacker is required to find at least 25% of the vulnerable computers. Comment on your results.

5. An interesting question is how many of the addresses should be shuffled. Would only shuffling vulnerable machines with empty addresses provide the same protection as shuffling the entire network? Defend your position, graphs can help the argument.

3 D-Quan's Dance Groves and Address Shuffling

Researchers from UNCC created a MT system that relies on Software Defined Networks (SDN) to provide a MT defense. One requirement of their approach is that computers must hop/shuffle in a subnet (which is a portion of the complete network). For this part of the assignment, you will create a model of this approach to determine the defense performance, then compare it to a more traditional notion of address shuffling (defined in the previous section).

Assume the address space consists of n addresses and are divided into s equal subnets. In addition, assume that each subnet will consist of the same number of computers, $\frac{m}{s}$, and vulnerable $\frac{v}{s}$ per subnet, where m and v are multiples of s (so everything divides evenly). Develop a model for subnet address shuffling where all n addresses are shuffled. An important part of the model will be the probability of attacker success (β) for finding α percent of v vulnerable computers. Again, the attacker can perform k scans. In addition your model should consider the frequency of shuffling denoted as γ.

Questions (*15 points*)

You must also correctly answer the following questions to receive full credit for this part of the project.

1. What is the equation for attacker success (β) for D-Quan's perfect address shuffling? Describe the equation and how it correctly models the system.

2. Assume the attacker only needs to find one vulnerable machine and the system is defended with perfect shuffling. Graph the attacker success rate as the number of scans (k) increases given a \16 (class B) network, where 25% of the addresses have machines and 10% of these machines are vulnerable. Compare with static addressing and traditional (no subnet) shuffling, then comment on your results.

3. Is this method better, worse, or the same as traditional shuffling? Are your results consistent with the results reported in the UNCC paper?

4 Honeypot

Develop a model for honeypots. In this system the attacker must avoid honeypots, if a honeypot is contacted the attack must stop (very high cost to the attacker). Therefore an important part of the model will be the probability of attacker success (β) for finding α percent of v vulnerable computers using k scan without contacting any honeypots. This will depend on the number of honeypots deployed which will be denoted as h, where $m + h \leq n$.

Questions (*30 points*)

You must also correctly answer the following questions to receive full credit for this part of the project.

1. What is the equation for attacker success (β) for h honeypots? Describe the equation and how it correctly models the system.

2. Assume the attacker only needs to find one vulnerable machine and the system is defended with honeypots. Graph the attacker success rate as the number of honeypots (h) increases given a \16 (class B) network, where 25% of the addresses have machines, 10% of these machines are vulnerable, and the attacker is allowed to scan 10% of the network. Compare with no defense and shuffling, then comment on your results.

3. Provide a 3-D graph of the attacker success rate (z-axis) as the number of scans (k, x-axis) and number of honeypots increases (h, y-axis). Assume a \16 (class B) network, where 25% of the addresses have machines, 10% of these machines are vulnerable, and the attacker is required to find at least 25% of the vulnerable computers. Comment on your results.

5 Shuffle or Honeypots

The probabilities of success for the different defenses can be used to determine the expected payoffs, and complete the payoff matrix. The matrix can then identify the best defense (or attack) strategy given the alternatives. As described earlier, an important part for developing the matrix will be to identify utility values for defenses and attacks. Using a generic security credit, instead of `US AMERICAN DOOLARS`, can simplify this process. If you decide to take this approach you will need to justify the various values associated with deploying defenses, attempting an attack, attacker success, and defender loss, etc...

Questions (*15 points*)

You must also correctly answer the following questions to receive full credit for this part of the project.

1. Assign revenue and cost variables for the different attacks and defenses, then determine the payoff equations and complete the matrix.
2. Make realistic assumptions about the comparative revenue and costs (for example shuffling is less expensive than honeypots) and determine the best strategies. Comment on your results, note graphs help.

Solutions

1.1

In this mathematical model, we need to assume that αv is an integer and $k \geq \alpha v$.
So, the whole question is equivalent to this: consider that there are n candy boxes in total (the IP address space), m of which have candies inside (each box contains either only one candy or is empty). There are two kinds of candies among them, the red ones (the vulnerable computers, with the number v) and the green ones (the secure computers, with the number $m-v$). Now, a boy randomly picks k boxes and only those red candies can meet his interest because he is so naughty (attacker), so those boxes contain green candies are the same with those empty boxes to him (IP addresses associated with secure computers and no computers are the same to the attacker). His choice space is $\binom{n}{k}$, suppose $x(0 \leq x \leq k)$ of his picks have red candies, then the target space is $\binom{v}{x}\binom{n-v}{k-x}$. To make this naught boy happy, there must be at least αv red candies, thus the possibility β that he is happy (attacker success) is:

$$\beta = \sum_{x=\alpha v}^{\min(k,v)} \frac{\binom{v}{x}\binom{n-v}{k-x}}{\binom{n}{k}}.$$

1.2

Here, $n = 2^{16} - 2 = 65534$, $m = \frac{n}{4} = 16383.5$, $v = \frac{m}{10} = 1638.35 \doteq 1639$ and $\alpha v = 1$, in R we have:

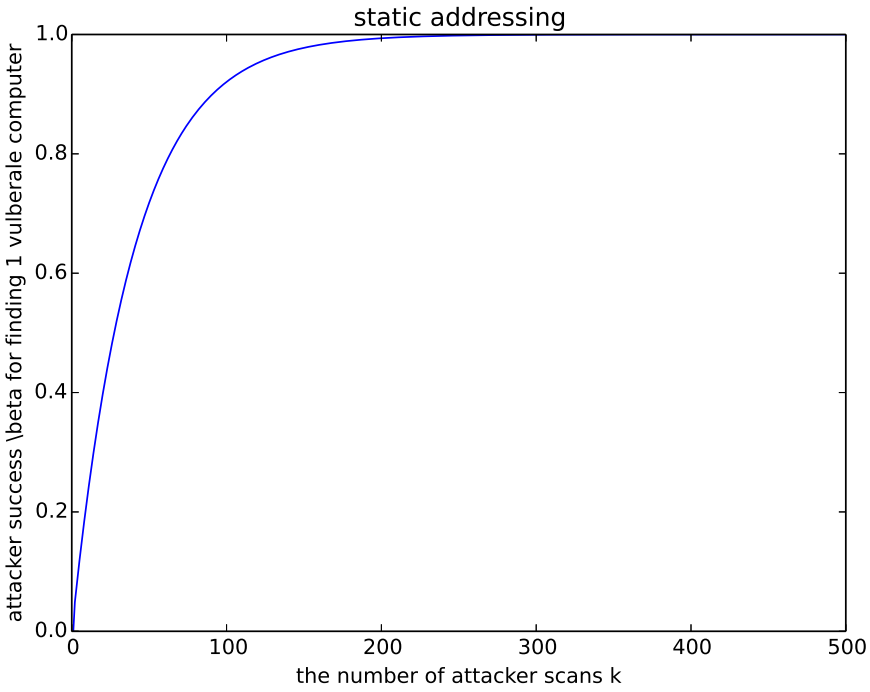

Figure 1: Plot of question 1.2, static addressing

From the figure above the attacker will almost always find the first vulnerable computer successfully within about 200 scans, that's because the total vulnerable computers are $\frac{1}{40}$ of the whole IP addresses space, the average scans the attacker need to find the first vulnerable computer is 40, thus with more than 200 scans could almost guarantee the attacker find at least one vulnerable machine.

1.3

Still, using Python we have the attacker success rate β as the number of scans k and attacker find percentage α increases:

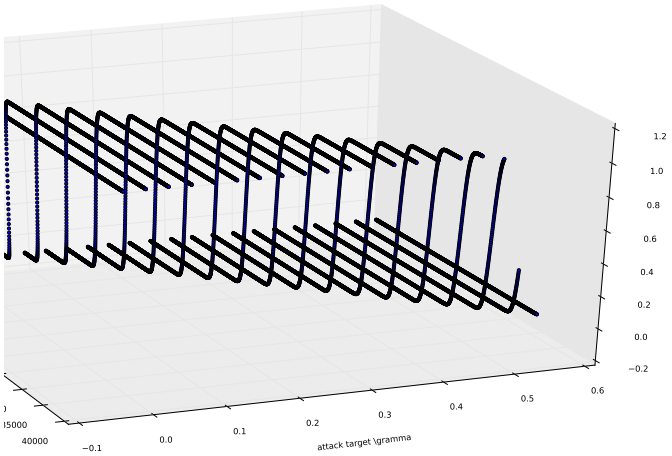

Figure 2: Plot of question 1.3, static addressing

As we can see here, with the attacker find percentage α increasing, the attacker needs more and more scans k to achieve a higher attack success. But when the attacker find percentage α becomes too high, say over 60%, then it is just too hard to achieve a acceptable attack success β, that's will require a significant big number of scans.
// which is very similar to the accomplishment of this project, if one (like, me) wants to finish 100% of it, // it's nearly impossible within :)
(Note: the paragraph above is commented out such that no one can read it unless he failed an inverse Turing Test).

2.1

For the perfect shuffling where $\gamma = 1$, now we have k shuffles, each shuffle after each scan. Here, we want to point out that for each successful scan (that the attacker find a vulnerable machine) the attacker makes, it's possible that the attacker could find the same vulnerable machine with what he found before, when this situation occurs, we assume that the attacker will view this as a new vulnerable machine, this is not only for the simplicity of building up models, but also for the fact that when the Id address space n is big enough, the possibility that the attacker find "enough many" same vulnerable machines is very small so that we can afford to omit it.

Using the same analog we stated in the question 1.1, the naughty boy will play the "seeking for red candies" game k times, but instead of picking all k boxes at the same time in one game, he will pick only one box at each one game and he is given n brand new boxes at each game. Since the number of red candies at each game are always v, thus the possibility of the boy gets a red candy at each game is $\frac{v}{n}$, then the possibility that he get x red candies in those k games is $\binom{k}{x}(\frac{v}{n})^x(1-\frac{v}{n})^{k-x}$. Still, at least αv red candies can make him happy, thus:

$$\beta = \sum_{x=\alpha v}^{\min(k,v)} \binom{k}{x}(\frac{v}{n})^x(1-\frac{v}{n})^{k-x}.$$

2.2

Here, $n = 2^{16} - 2 = 65534$, $m = \frac{n}{4} = 16383.5$, $v = \frac{m}{10} = 1638.35 \doteq 1639$ and $\alpha v = 1$. In order to compare its graph with static addressing, I plot both curves in the graph using Python, as follows

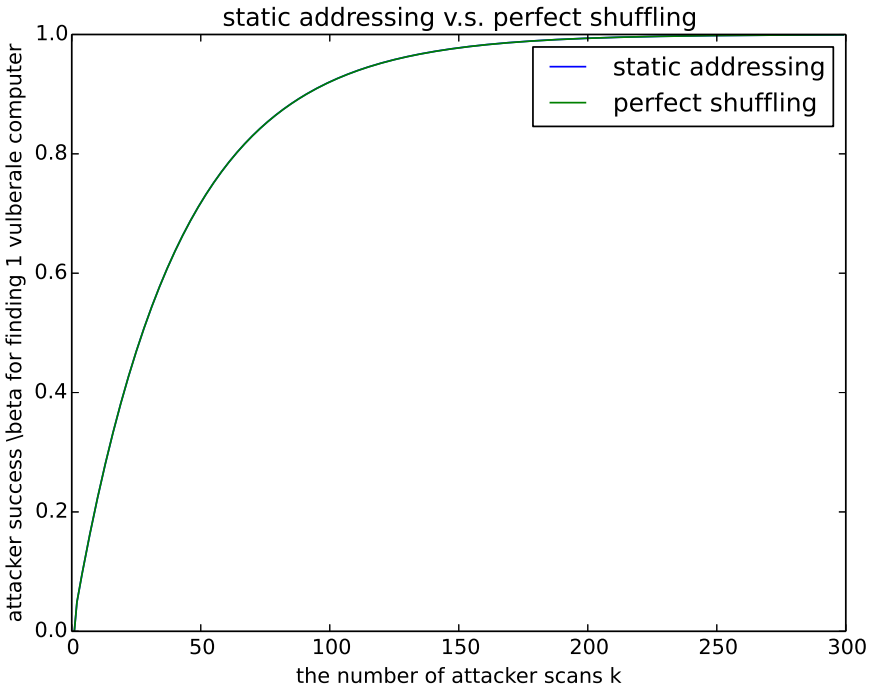

Figure 3: Plot of question 2.2, using address shuffling strategy

As we can see that, those two curves are overlapped, can't really tell a differences visually. But deep inside we tend to believe that perfect addresses shuffling should do slightly better than static address (in terms of decreasing the attacker success given the same scans), otherwise our effort is in vain and there will be no point of calling this method a "defense". Theoretically, the attacker is playing the game without replacement, the possibility that he to get a vulnerable machine at the kth scan is depending on his previous "$k-1$" scans. While under perfect shuffling strategy, the attacker is playing k independent games, that his posteriori scans will not be affected the results of his prior scans. Thus, as the scans k becomes bigger, the difference between those two methods should become more obvious:

$scans k$	$static - address(S)$	$perfect - shuffling(P)$	$winner$
50	0.72512198370323211	0.72506160021762656	P
70	0.83439473355419358	0.8342980136319853	P
100	0.92258098492144669	0.92247064238466114	P
150	0.9782164209012405	0.97813764357046307	P
$\vdots$	$\vdots$	$\vdots$	P

Table 1: Comparison of static address and perfect shuffling given bigger scans k .

2.3

With shuffling frequency γ, then we will perform γk shuffles during the attacker's k scans, thus we will perform one shuffle after $\frac{k}{\gamma k} = \frac{1}{\gamma}$ scans. During each $\frac{1}{\gamma}$ scans before the next shuffling, the possibility that the attacker will get x vulnerable computers is: $\frac{\binom{v}{x}\binom{n-v}{\frac{1}{\gamma}-x}}{\binom{n}{\frac{1}{\gamma}}}$, thus the final total number of the vulnerable computers the attacker can get in the end is the sum of those x_is $(0 \leq i \leq \gamma k)$ from each $\frac{1}{\gamma}$ scans, then the

attacker success β is:

$$\beta = \sum_{x_{\gamma k}=0}^{\min(k,v,\frac{1}{\gamma})} \cdots \sum_{x_2=0}^{\min(k,v,\frac{1}{\gamma})} \sum_{x_1=0}^{\min(k,v,\frac{1}{\gamma})} \prod_{i=1}^{\gamma k} \frac{\binom{v}{x_i}\binom{n-v}{\frac{1}{\gamma}-x_i}}{\binom{n}{\frac{1}{\gamma}}}, where \sum_{i=1}^{\gamma k} x_i \geq \alpha v$$

2.4

It's toooooo hard to plot this one :(

When I showed my equation in 2.3 to you Monday afternoon, you said forgot about it ...

2.5

Shuffling the entire network provide more protection. In the equation of question 2.3, only shuffling vulnerable machines with empty addresses will make the denominator smaller, hence the attacker success will become bigger, thus shuffling the entire network is better.

3.1

For each subnet i, $1 \leq i \leq s$, suppose the attacker also performs y_i scans, thus $\sum_{i=1}^{s} y_i = k$. Since it's perfect shuffling $\gamma = 1$, the whole subnets shuffl after each scan. Thus the possibility that the attacker find a vulnerable with each scan at each subnet is still $\frac{v/s}{n/s} = \frac{v}{n}$. Also also, suppose the attacker will get x_i vulnerable computers from the ith subnets within y_i scans, we have

$$\beta = \sum_{x_s=1}^{\min(y_s,\frac{v}{s})} \cdots \sum_{x_2=0}^{\min(y_2,\frac{v}{s})} \sum_{x_1=0}^{\min(y_1,\frac{v}{s})} \prod_{\substack{i=1 \\ \sum_{i=1}^{s} y_i = k}}^{i=1\ldots s} \frac{\binom{\frac{v}{s}}{x_i}\binom{\frac{n-v}{s}}{y_i-x_i}}{\binom{n}{y_i}}, where \sum_{i=1}^{s} x_i \geq \alpha v$$

3.2

When s is small enough, this plot is quite similar to question 2.2. (The reason why s can't be too big will be further addressed in question 3.3.) Because no matter how the defender is diving the network into subnets and do shuffling, there is not making much differences to the attacker, and the attacker may not even be aware of this.

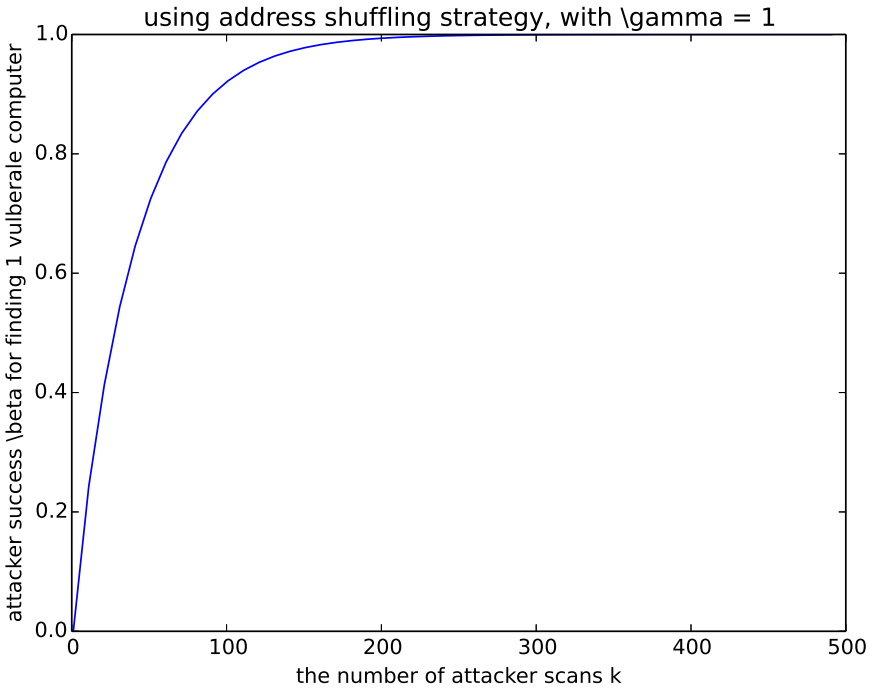

Figure 4: Plot of question 3.2, using D-Quan's address shuffling strategy

3.3

From my point of view, I think the method is worse or no better than the traditional shuffling. Let's consider an extreme example, that when s is too big, say $s = 2^8$, then with a \16 (class B) network, you are going to get $\frac{2^16}{2^8} = 2^8$ subnets (class C), now the total available IP addresses are $2^8 \times (2^8 - 2) = 2^16 - 2^9$, while originally we have $2^16 - 2$ available IP addresses, this diving leads to $2^9 - 2$ less than original IP addresses space, which make the denominator of the attacker success equation small, hence make the attacker success β bigger.

4.1

In this question, there is one situation that, the attacker could achieve his success (by finding αv vulnerable machines) without contacting any honeypots (yes, he is lucky) before using up all the k scans, then this occurs, we assume that he will continue scanning until finishing his k scans, during this process he could gain more vulnerable machines but also he may contact honeypots, in this situation we will view this as a failure even though he already got enough vulnerable machines, because he is caught (never be too greedy).

Now, first of all, we want to know the possibility that the attacker performed k scans without contacting any honeypots:

$$P(k - scans - without - honeypots) = \frac{\binom{n-h}{k}}{\binom{n}{k}}.$$

Thus,

$$\begin{aligned}
\beta &= P(find - \alpha v - vulnerablecomputers | k - scans - without - honeypots) P(k - scans - without - honeypots) \\
&= \sum_{x=\alpha v}^{\min(k,v)} \frac{\binom{v}{x}\binom{n-h-v}{k-x}}{\binom{n-h}{k}} \frac{\binom{n-h}{k}}{\binom{n}{k}} \\
&= \sum_{x=\alpha v}^{\min(k,v)} \frac{\binom{v}{x}\binom{n-h-v}{k-x}}{\binom{n}{k}}.
\end{aligned}$$

4.2

Here, $n = 2^{16} - 2 = 65534$, $m = \frac{n}{4} = 16383.5$, $v = \frac{m}{10} = 1638.35 \doteq 1639$, $\alpha v = 1$ and $k = \frac{n}{10} = 6553$, in Python we have:

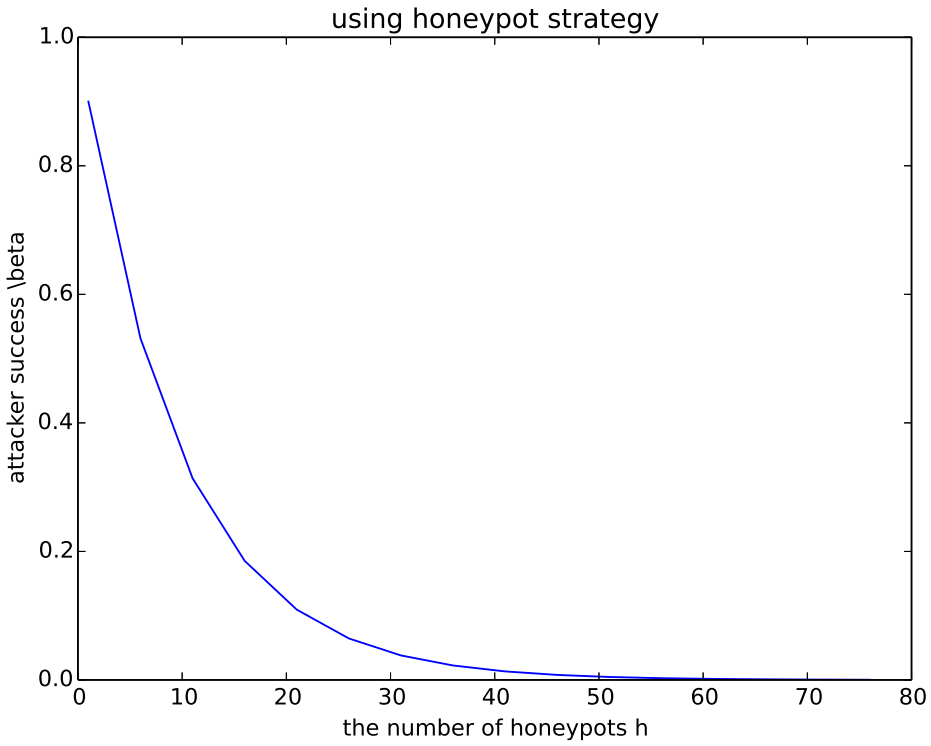

Figure 5: Plot of question 4.2, using honeypot strategy

With the $k = 6553$ scans, from the result of Figure 3 in question 2.2, we know that the attacker will almost necessarily find at least one vulnerable machine. But with the number of hoenypots increasing, the attacker would be less likely to avoiding contacting honeypots. When the number of honeypots is big enough, say great than 60 as we can tell from the graph, the attacker will almost necessarily to fail if he insist to finish all the scans.

4.3

Using Python we have:

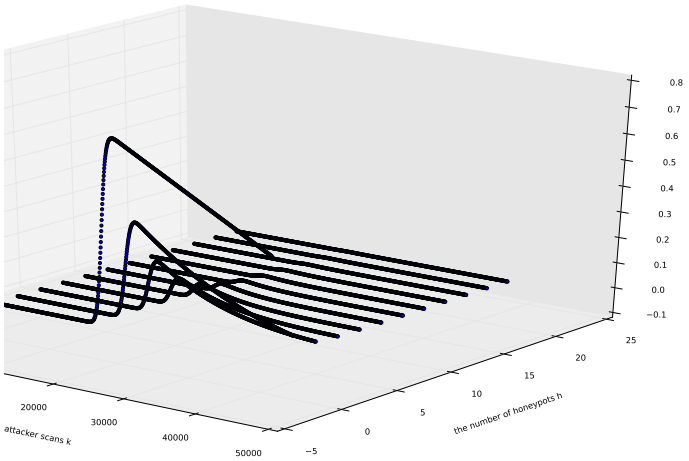

Figure 6: Plot of question 4.3, using honeypot strategy

Since it is require and find at least 25% of the vulnerable machines, from question 2 we know that attacker need a significant many scans to accomplish this (somewhere above $20,000$). But with the numbers of honeypots increases, it's harder and harder and finish those scans without contacting any honeypots, as you can see from the graph above, the attacker success β approaches to 0 when there are more than 20 honeypots deployed.

5.1

The defender side:
revenue:
$1000 per win using any strategies
cost:
$2000 per lose
no cost using static address
$5 per shuffling using addresses shuffling address strategy
$100 per honeypot deployment using honeypots strategy

The attacker side:
revenue:
$2000 per win
cost:
$1000 lose per lose
$50 per honeypot deployment using honeypots strategy

5.2

Suppose a \8 (class C) network, with 100 machines and 20 of them are vulnerable, given a attacker wants to scan $k = 50$ scans to get at 25% of the vulnerable machines, which is 5. Now consider the attacker success β:

For static addressing:

$$\begin{aligned} \beta &= \sum_{x=5}^{20} \frac{\binom{20}{x}\binom{254-20}{50-x}}{\binom{254}{50}} \\ &= 0.35 \end{aligned} \tag{1}$$

For perfect shuffling:

$$\begin{aligned}\beta &= \sum_{x=5}^{20} \binom{50}{x} (\frac{20}{254})^x (1-\frac{20}{254})^{50-x} \\ &= 0.35.\end{aligned}$$

For honeypots:

$$\beta = \sum_{x=\alpha v}^{\min(k,v)} \frac{\binom{v}{x}\binom{n-h-v}{k-x}}{\binom{n}{k}}. \tag{2}$$

We have:

$$\begin{aligned}\beta &= 0.28, whereh = 1 \\ \beta &= 0.23, whereh = 2 \\ \beta &= 0.18, whereh = 3 \\ \beta &= 0.14, whereh = 4 \\ \beta &= 0.12, whereh = 5\end{aligned} \tag{3}$$

Based on those costs and revenues we assigned above, we have the expected income for the defenders is that:

$$\begin{aligned}E(static) &= (1-0.35)\times 1000 + 0.35 \times (-2000) \\ &= -50 \\ E(perfect-shuffling) &= (1-0.35)\times 1000 + 0.35 \times (-2000) - 50 \times 5 \\ &= -300 \\ E(honeypots|h=1) &= (1-0.28)\times 1000 + 0.28 \times (-2000) - 100 \\ &= 60 \\ E(honeypots|h=2) &= (1-0.23)\times 1000 + 0.23 \times (-2000) - 100 \times 2 \\ &= 110 \\ E(honeypots|h=3) &= (1-0.18)\times 1000 + 0.18 \times (-2000) - 100 \times 3 \\ &= 160 \\ E(honeypots|h=4) &= (1-0.14)\times 1000 + 0.14 \times (-2000) - 100 \times 4 \\ &= 180 \\ E(honeypots|h=5) &= (1-0.14)\times 1000 + 0.14 \times (-2000) - 100 \times 5 \\ &= 140\end{aligned} \tag{4}$$

More honeypots will definitely improve more defence, but deploying honeypots are also very expensive. In our example, we decide to deploy 4 honeypots to maximize our revenue.