Designing Good Deceptions in Defense of Information Systems

Neil C. Rowe
Cebrowski Institute, U.S. Naval Postgraduate School
Code CS/Rp, 833 Dyer Road, Monterey CA 93943 USA
ncrowe@nps.edu

Abstract

Since attackers trust computer systems to tell them the truth, it may be effective for those systems to lie or mislead. This could waste the attacker's resources while permitting time to organize a better defense, and would provide a second line of defense when access controls have been breached. We propose here a probabilistic model of attacker beliefs in each of a set of "generic excuses" (including deception) for their inability to accomplish their goals. We show how the model can be updated by evidence presented to the attacker and feedback from the attacker's own behavior. We show some preliminary results with human subjects supporting our theory. We show how this analysis permits choosing appropriate times and methods to deceive the attacker.

1. Introduction

Access controls are not currently doing a good job of protecting computer systems as is witnessed by the many attacks that can subvert controls. Access controls have been studied for a long time, and significant innovations are now rare. So it is valuable to examine secondary "lines of defense" for once access controls have been breached.

Intrusion-detection systems, computer forensics, and honeypots [12] are secondary lines of defense, but they are relatively passive and focused on data collection. Closing connections, ports, and services automatically during attacks stops them but tells the attacker we recognize the attack and encourages a different attack. Trying to trace connections from an attacker is very difficult with most Internet routers and only really successful within a subnetwork with specialized router software. Counterattacks are generally illegal and unlikely to find the right target given the difficulty of tracing connections.

One second line of defense does avoid these disadvantages, however: deception. Information systems could lie, cheat, and mislead attackers to prevent them from achieving their goals [19]. Since people expect computers to tell them the truth, such deception can be very effective with minimal resources. Deception is particularly important with time-critical military-style attacks such as those by cyber-terrorists or state-sponsored information-warfare teams, where just delaying the attack with deceptions could be critical while finding a permanent defense. Deception also can be equally effective with attacks by insiders as well as outsiders.

Most attacks on computer systems use some form of deception. Classic methods include masquerading as someone or something (like a system administrator, an audio file, or an IP address), concealing unexpected components within something innocuous (like Trojan horses), and asking for unneeded resources (like denial-of-service attacks). So it seems fair to respond to attackers with similar methods. Deception is a common feature of human nature and it occurs frequently in animals and plants in many visual and behavioral forms [2].

As a defensive tool for information systems, deception has been used in honeypots [5] and honeynets as a way to keep the attacker busy. Honeypots are systems with no purpose except to encourage attacks so data can be collected, and honeynets are networks of honeypots. Deceptions like fake files are used in some honeypots to keep attackers interested for a while. But we want to explore more sophisticated deceptions, and we want deceptions on ordinary computers where they could protect those systems – honeypots don't try to fend off most attacks.

We have been studying the more general problem of providing deceptive behavior for the protection of computer systems and networks. Deception is a way to foil attacks much like directly fighting them off. Deception may convince an enemy to go away without any fight. Using an intrusion-detection system, we monitor user behavior for suspiciousness. As suspiciousness increases, we first provide minimum deceptive measures, and then increase their frequency and severity. The trick is to use deception sparingly and consistently to keep the attacker fooled as long as possible, tying up their resources while reducing their chances of successful attack. To do this, we need some planning. Deception is only useful when we are at least moderately sure we are under attack since otherwise we risk hurting legitimate users (see section 2.6). But deception may be useful even when we are very sure of an attack, as a delaying tactic, although on non-honeypots it may be safer to disconnect the attacker.

Some examples of these kinds of deceptions that we have implemented are [18, 19]:

- A Web site that, when under denial-of-service attack from too many processing requests, delays still further in responding to those requests to give the impression that it is more affected by the attack than it really is.
- A Web site that provides files of data compiled at random from real files to confuse spies into seeing nonexistent connections.
- A file-transfer utility that, when it sees a signature of a common attack, pretends to succumb by responding in the same way an affected system would.
- An operating system that, when it recognizes an attacker is downloading a rootkit to install on it, deletes the rootkit some time after download without telling the attacker.

Ethical problems arise in regard to initiating deception, but most ethical theories endorse deception to protect against a serious harm [3, 15]. Destruction of the functionality of a computer system by an attacker can be argued to be such a serious harm.

2. Deceptive excuses

Deception planning for information systems can benefit from the experience of professionals who plan deliberate deceptions on a routine basis. The best examples are stage magicians and military planners. Both note that plausibility is the key to the effectiveness of a deception [8, 14, 24]. Plausibility is enhanced when deceptions fit into familiar patterns [11]. This says that the good deception planner must anticipate the false theories that the deceived could likely believe and try to encourage them. Examples from stage magic are "magician can read minds" and "magician can teleport objects"; examples from military planning are "attack will be at the pass" and "enemy has more resources than you do". Encouraging such false theories or "excuses" exploits the tendency of the human mind to easily see patterns where none exist, an idea supported by the popularity of astrology and psychics.

2.1 Generic deceptive excuses

False excuses are a simple and effective form of deception [20]. Good generic excuses for an information system to refuse to do something that an attacker wants include:

1. "Communications breakdown": Communications problems between the attacker and the system cause systematic misinterpretation of attacker commands. An example is dropping the first character of every command line.
2. "System crash": A computer system has stopped working.
3. "Software broken": Parts of the software of a computer system are not working.
4. "Network down": A network connecting the attacker to the target is not working.
5. "Buggy system": A system has bugs that prevent it from working correctly. An example would be the message "Cannot find executable" anytime the attacker attempts to execute a system command.
6. "System testing": A system is being tested or installed.
7. "Hacked": A system has been compromised and is now controlled by another attacker. An example indicator would be a welcome message to the system that mentions a hacker alias.
8. "Practical joker": A system is being controlled by someone deliberately trying to irritate the attacker. An example would be the message "You lose, stupid!" with failure on attacker commands.
9. "Policy enforcement": Security policy prevents the actions from being accomplished. An example would be refusal to download executables in general when the real reason is that the particular executable has a known suspicious name.

Besides these, two other hypotheses can be held by an attacker: that deception is being practiced on them by the information system (the one hypothesis we do not want them to have), and the "null hypothesis" that the system is in normal operation.

Each of these excuses has specific requirements for how they can be used:

- Communications breakdown: Can happen anytime with interactive software, especially new software. Should persist to the end of the session, and perhaps over other sessions of the same user.
- System crash: Can happen anytime. Should persist for a significant duration.
- Software broken: Can happen after an unusual command, to suggest the "you broke it" hypothesis which plays on the attacker's sense of guilt. Should persist for a long time but need not be very consistent.
- Network down: Can happen with network commands, and is most convincing for the first network command of a session or first network command involving considerable data transfer. Should persist a while.
- Buggy system: Can happen with any command used for the first time, and must be used consistently from then on. Should persist a while.
- System testing: Similar to "buggy system".
- Hacked: Can happen with any command involving basic facilities of the operating system. Should persist a long time.
- Practical joker: Can happen anytime it can be manifested in verbal abuse of the attacker. This excuse can be quixotic, appearing and disappearing inconsistently.
- Policy enforcement: Must happen consistently from the start of a session. Should persist a long time.

2.2 A probabilistic model of belief and suspicion

Since it is often good to suggest generic excuses indirectly, and system events may suggest more than one generic excuse, it helpful to estimate the probability of an attacker's belief in an excuse. We will use a Bayesian belief-update model since Bayesian models are often the simplest for many applications. This is motivated not so much by the belief that people do classical Bayesian reasoning as the observation that Bayesian methods are flexible enough to be tailored to many situations. This approach is influenced by the theory of trust in [22] which postulates that trust is a bet on future contingencies. Our approach to deception modeling can be contrasted with the more linguistic approach of [7], the simple set-theoretic one of [16], or the three-valued "subjective logic" of [13] which distinguishes belief, disbelief, and uncertainty.

We assume the attacker or information system formulates general hypotheses about whom with which they interact, from the results of their commands. The key hypotheses we will label as:

- H_{e1}, H_{e2}, H_{e3}, etc.: the hypothesis by the attacker of each of the generic excuses;
- H_d, the hypothesis by the attacker that deception is being practiced on it by the information system;
- H_n, the hypothesis by the attacker that the system is behaving normally; and
- H_m, the hypothesis by the information system that its user is malicious.

Then their probability of belief in the hypothesis H as a function of the previous evidence E and the current evidence E_n can be calculated from its odds $o(H \mid (E_n \& E))$ in the odds form of Bayes' Rule as:
$o(H \mid E) p(E_n \mid (H \& E)) / p(E_n \mid (\sim H \& E))$
where $o(X) = p(X)/(1 - p(X))$ and p is probability. Since all the hypotheses considered here are rare occurrences on computer systems, we can eliminate the ~H term to get:
$o(H \mid E) p(E_n \mid (H \& E)) / p(E_n \mid E)$.
When E is not associated with the hypothesis, such as the action of opening a local file normally and the hypothesis "network is down",
$p(E_n \mid (H \& E)) = p(E_n \mid E)$ and the odds of H are not changed by the evidence.

For example, consider the attacker's hypothesis that the network is experiencing problems. If the attacker requests something requiring the network and it fails to happen, it could be that they did it incorrectly or the report of failure is incorrect. So let us suppose the attacker believes with a priori probability 0.1 that the network is down, they try to download a file, and it fails to appear in the destination directory although no error messages appear. Assume that the probability that the file failed to appear and no error message appeared when the network is down is 0.5, and the probability of those two things when the network is not down is 0.1. Then the odds of network being down is (0.1/(1-0.1)*0.5/0.1 = 0.56, which corresponds to a probability of 0.36, so the probability increased significantly.

2.3 Estimating the attacker's hypothesis probabilities

To use this model we need to calculate probabilities for three rather different things, each of which needs different methods.

First consider the excuse hypotheses. The evidence we show an attacker for each of them should be strong; for instance, the message "The network is down" is strong support for the hypothesis that the network is down. However, everyone familiar with software knows that messages can be wrong, as the software issuing the message may have bugs or faulty information. In reverse, the excuse can usually be constructed so that it always occurs in a real system having that associated problem; for instance, the same "network down" message can be used that occurs when the network is actually down. Then we have $p(E_n \mid (H_{ek} \,\&\, E)) = 1$ for all uses of the excuse, and $p(E_n \mid E) = p(E_n)$ for the first manifestation of the excuse and 1 for uses of the excuse thereafter. This gives
$o(H_{ek} \mid (E_n \,\&\, E)) = o(H_{ek} \mid E) / p(E_n)$.
But in general,
$o(H_{ek} \mid (E_n \,\&\, E)) =$
$o(H_{ek} \mid E) p(E_n \mid (H_{ek} \,\&\, E)) / p(E_n \mid E)$ is needed to cover cases such as those where "network down" can be manifested by several different error messages. The probability that the system is behaving normally can be found as one minus the sum of the all generic-excuse probabilities and the deception probability.

For the probabilities that the system has for the user being malicious H_m, we can also use Bayes' Rule. However, better estimates can be obtained from an intrusion-detection system [17] if we have one. We can use reports from a network-based system for outsiders and a host-based system for insiders. Anomalies can be mapped to a probability of being malicious through expert-systems methodology; misuse signatures can be mapped to large but not certain probabilities. However, if we have detailed knowledge of the plan of particular attacks, we can do better in recognizing when they are being used from observing sequences of user actions.

For the probability that the attacker perceives that he or she is being deceived, this will depend somewhat on the personality of the attacker and their experience. Nonetheless, much of the estimate of the probability can be based on what the attacker sees at the time of their attack, since rarely do attackers have reason to be suspicious, and experiments have shown that people are poor detectors of deliberate deception [9]. We postulate that the attacker will be suspicious proportional to the suspiciousness of the systems' responses to them, which is a function of the appropriateness and frequency of the response in the circumstances. So for instance, an attacker trying to download a suspicious file will be more suspicious after a "router error" message than a "network down" message because router errors are rarely announced. We also expect an attacker to be suspicious proportional to the likelihood that a defender will use deception. For instance, if it is known that the defender uses honeypots, the attacker will be more suspicious of an inability to export attacks than otherwise. Finally, we postulate that the attacker will be suspicious of the system proportional to the attacker's self-perception of their own suspiciousness, since the more obvious the attacker's attack, the more they expect some retaliation. For example, we expect an attacker to be more suspicious of an error message after an attempt to copy the system password file than after an attempt to copy one of their own text files. In part this is what psychiatrists call "projection" of the attacker's own self-assessment onto other people, and in part this is a pragmatic assessment that dishonesty usually is found out.

There are also two distinct ways for attackers to assess their own suspiciousness for the last factor above: In what just happened (a local measure), or in the cumulative impact of everything that happened (a global measure). Copying the password file would be a single suspicious action; downloading 1000 documents to an external site would be a cumulative action. Attackers will differ as to how they weight the two measures. For instance, suppose an attacker copies the password file, runs a cracker program on it, tries to download it to another computer, and receives the error message that "The network is down." This would be much less suspicious to an attacker with a high weight on the local measure than copying the

the password file and immediately getting the error message. So if c is the weight on the local measure, our model of attacker suspiciousness is:

$$c*[p(H_m \mid (E_n \,\&\, E)) - p(H_m \mid E)] \\ + (1-c)[p(H_m \mid (E_n \,\&\, E))] \\ = p(H_m \mid (E_n \,\&\, E)) - c*p(H_m \mid E)$$

The three abovementioned factors for suspiciousness are generally independent, so it makes sense to multiply them to get an overall suspiciousness measure. Then our model is:

$$\Delta p(H_d \mid (E_n \,\&\, E)) = \\ [p(H_m \mid (E_n \,\&\, E)) - c*p(H_m \mid E)]* \\ p(H_d \mid H_m)*p(H_d \mid E_n)$$

where E_n includes both the event of the attacker's nth action and the system's nth response.

We use a delta in the above equation because we believe distrust is additive. Many writers have noted that trust and distrust are asymmetric. Both [22], from a qualitative perspective, and [13], from a quantitative perspective, model trust as easily increasing and decreasing with circumstances, but distrust as something that generally only increases since incidents creating distrust are remembered for a long time as traumatic. So we should use the preceding formula as a positive amount by which to increase the probability of deception. To prevent it exceeding 1.0, we can use the formula for disjunctive combination of independent probabilities:

$$p(H_d \mid (E_n \,\&\, E)) = \\ 1-(1-p(H_d \mid E))*(1-\Delta p(H_d \mid (E_n \,\&\, E))$$

We can compare this value to the probabilities estimated for the other hypotheses by the attacker. If the "normal behavior" hypothesis is strongest, the attacker should stay logged in. If the total strength of the generic excuses is stronger than either the "normal behavior" or "deception" hypotheses, the attacker should give up and go elsewhere, since they mean that attacker goals will never be achieved. But if the deception hypothesis is the strongest, that is bad: The attacker will likely stay logged in and cause new kinds of mischief. We must design our deceptions to prevent this.

2.4 Estimating the parameters

The parameters necessary to apply the above formulae can be obtained from human subjects by giving them scenarios and asking them to estimate the probability of a given hypothesis. This is especially useful for our special model of the deception hypothesis. We can ask subjects to estimate probabilities in the same situation but with different responses by the system, or different situations with the same responses, to approximate key parameters easily. The probabilities for H_m come from an intrusion-detection system, and the a priori suspiciousness of an action comes from statistics of typical user behavior on the system.

We administered a questionnaire to 11 subjects to do this estimation, 7 students and 4 faculty knowledgeable about information security (see the Appendix for the questionnaire and results). We gave them six scenarios involving an attacker on a computer system, each three or four steps long, and asked them to estimate the probability the attacker was being deceived by the system at each step. There were clear individual differences in the responses, ranging from an average of 0.10 for a tolerant subject to 0.77 for a suspicious one. Nonetheless, the average response to each question over the subjects was a consistent indicator, and was consistent with our theory presented above. In particular, an estimate of c/(1-c) can be obtained from the ratio (2b-2a)/(3c-3b), so c was 0.51 for our subjects. Note that the monotonically increasing nature of distrust was substantially confirmed: In all but the last two questions (for which a misunderstanding may have been involved as to whether the estimate was cumulative), suspicion increased monotonically through the sequences.

2.5 An example

As an example, consider sequence 3 from the questionnaire. An attacker logs into a computer system, tries to copy an executable to the system binary directory, gets a protection-violation error message, tries to transfer the file to another site, and gets the last character of their command deleted by the system. Then they try to list their home directory and again get the last character of their command deleted. Table 1 shows our estimates of four attacker hypothesis probabilities in this situation: the probability of a buggy system (pb), the probability that communications defaults are set wrong (pc), the probability that the attacker is being deceived (pd), and the probability that everything is normal (pn, assuming no other hypotheses are possible here). Probabilities are given just after the command response.

Table 1: Example of hypothesis probabilities.

Command	Response	pb	pc	pd	pn
1: Log in	Normal	.010	.010	.000	.980
2: Copy executable to bin directory	Protection error	.015	.010	.020	.955
3: Transfer executable externally	Communications error	.018	.020	.033	.929
4: List home directory	Communications error	.035	.091	.033	.841

Here we assume the a priori probability of the first three hypotheses were 0.01, 0.01, and 0.00, the ratios of probability of each event given a buggy system to its a priori probability were 1.0, 1.5, 1.2, and 2.0, and the ratios given communications defaults were wrong were 1.0, 1.0, 2.0, and 5.0. That is, protection errors for the "bin" directory are a little suspicious, and this kind of communications error is also suspicious (especially in listing a home directory); but all other actions appear close to normal system behavior.

The probability that the user was malicious based on their actions was assumed to be 0.00, 0.20, 0.10, and 0.00 respectively from each action alone. Assume the probability of maliciousness is additive (as is often true for low probabilities). Then the probability the user is malicious is 0.00 after step 1, 0.20 after step 2, and 0.30 after steps 3 and 4; the probability the user thinks that they appear malicious, using c=0.51, is 0.00 after step 1, 0.20 after step 2, 0.25 after step 3, and 0.15 after step 4.

Assume the probability of deception given that the system observes malicious behavior is 0.5 (the attacker could estimate this by recalling how many honeypots they have visited). The probability the system is using deception is then 0.00 after step 1, 0.02 after step 2, 0.02+0.0125=0.033 after step 3, and 0.033+0=0.033 after step 4. So deception by the system should not seem likely given the uncertainty of maliciousness and the lack of immediate correlation to the attacker's suspicious actions. So normal system operation is the most likely hypothesis, and we expect the user to continue with the system for a while.

2.6 Two theorems

The model developed above has several implications for finding the best way to deceive an attacker.

Theorem 1 (Legitimate-user penalty). Assume that the probability of a malicious user at some point is p_m, the benefit of preventing a malicious user from achieving their ends is c_m, and the cost of preventing a nonmalicious user from achieving their ends is c_n. Then prevention of the attack is desirable at that point by even partially successful means if $p_m > c_n / (c_n + c_m)$. Proof: Then the expected benefit of preventing an unknown user from achieving their ends is $p_m c_m - (1 - p_m) c_n$. This will be positive when $p_m > c_n / (c_n + c_m)$. If prevention of the attack has only a probability of success (as by a deception), that probability multiplies a positive number c_m and still leaves a positive number. QED.

An important issue for the defender is when to stage deceptions. The following gives a useful criterion for delaying them, simplifying the number of places we need to consider.

Theorem 2 (Excuse delay): Assume the model of attacker belief in generic excuses and deception given in section 2.3. Assume we can apply a generic excuse as justification for failure to execute a user command at some state S after which the user has just done something suspicious, or we could apply it at state S2 which follows S and a subsequent action A by the user. It is always preferable to apply the excuse to S2 provided A is not suspicious and does not increase damage to the system. Proof: If A is not suspicious, then $p(H_m \mid (E_n \,\&\, E)) - c * p(H_m \mid E)$ will be less after S2 because the subtracted term will be larger. At the same time, the $p(H_d \mid H_m)$ term will be unchanged because it is a characteristic of the defender's general psychology, and the $p(H_d \mid E_n)$ term will be unchanged because the deceptive response will be the same. In addition, the longer one can wait, the more accurate can be one's assessment of whether a user is malicious, and the less likelihood of penalizing an innocent user. QED.

3. Implementation

3.1 The modeling approach

To be most effective in using deceptions against attackers, we should anticipate what they will do using attack models such as that of [23]. In [18] we used a hierarchical-planning approach where attackers had goals and subgoals, and knew methods that could possibly achieve them. Our approach postulates a set of actions (e.g. login, copy file across the network, execute buffer overflow, decompress, erase system logs), each with a set of preconditions and postconditions. Postconditions can be random and/or contextual. In addition, each action has goals for which they can be recommended.

The clearest way to specify this information about actions is in predicate calculus. Then finding a plan can be done with resolution theorem proving, or in most cases, logic programming like Prolog. But this is computationally expensive since finding such a plan in general is an NP-hard problem; we need to be able to quickly foil attackers. So we instead run the action specifications and planning machinery many times to create an approximate Markov model of attacks. This is a large graph with nodes labeled with the states found in the runs and the transitions representing actions; probabilities on the branches are proportional to the observed frequencies. Such a graph simplifies tracking attackers, although they can digress temporarily from it. Such a graph is an approximation of the plan space since it cannot represent variables and quantification, nor most rare possibilities. Nonetheless, it can represent most possible attack phenomena.

3.2 Generating an example graph for a rootkit attack model

To test the ideas proposed, we used an example attack model we built for the main steps of the classic hacker strategy of installing a rootkit on a computer system. It has 19 types of actions that can be instantiated to 93 distinct actions. The predicate-calculus specifications refer to 115 distinct facts (and in some cases their negations), and permit 13 kinds of random events.

To generate the Markov graph, we ran the predicate-calculus planning specifications 500 times using our planning software written in Prolog. The goals of the plans were to install the rootkit and a backdoor and then log out, and the system being attacked was considered predictable (so no errors or deceptions were considered). We chose random starting states from a set of 3072 intuitively reasonable ones. Besides the randomness introduced by random events, each action's duration was determined by an evenly distributed random variable on a specified range. These three kinds of randomness resulted in 21,720 states in the 500 runs, of which 10,276 were distinct, for an average of 42.4 steps per run. The latter were used to build a Markov model of 10,276 nodes and 10,103 branches.

3.3 Choosing where to apply generic excuses

Each generic excuse of section 2.1 can only be begun or used at particular branches in the Markov state graph because the excuse must be causally related to the preconditions and postconditions of the action. Much of this can be checked automatically with the predicate-calculus specifications of the actions. For instance, the generic excuse "ftp software broken" only affects actions of initiating and closing FTP connections and making file transfers, since they are the only actions with a precondition that the ftp software is working.

Other generic excuses apply to all actions but are particular about when they are initiated. For instance, "communications breakdown" and "network down" should be used for every applicable communications or network command once started -- but to be convincing, they should start with sufficiently complex commands that could cause their failures, like runs of new executables. "Communications breakdown" is most likely when changing communications defaults in login or changing accounts; "network down" is most likely in transferring large files across the network.

Finally, broad generic excuses like "buggy system" and "system testing" tend to be long-term problems. Thus, they are not very convincing when initiated after a session has started, and should only be a last resort. But they can be used effectively if a system has advance knowledge of certain kinds of attacks by from warnings by sibling sites about ports and methods of entry.

For our specific state graph for rootkit installation and a buffer overflow, initial suspiciousness of the attacker's actions is low until a buffer overflow is done. Thus Theorem 1 rules out deception before the overflow, a total of 684 branches. 2760 subsequent attacker actions are ruled out since only defender actions can involve deception, and 116 are ruled out as occurring too late (after the rootkit has been installed). 1506 of the remaining branches are ruled out by Theorem 2 where they are followed by less-suspicious attacker actions. (The suspicious actions are overflowing the buffer, obtaining root privileges, and installing the rootkit.) This leaves 5210 branches as suitable starts of a generally-applicable excuse, which then would be offered consistently at every related attacker command during the session. So we track users on the attack graph and initiate deception if a user reaches one of these 5210 branches. Some of these commands will be better for certain excuses than others, those whose actions are most semantically related to the excuse. It would also be a good idea to vary the occurrence of the start of the excuse from session to session to avoid a different kind of suspicious consistency, in an unexpected event.

4. Related work

Deception plays an important routine role in many important sectors of human activity, including law, business, entertainment, and the military [9]. It is important in military science and cyberattacks are a form of warfare. [8] divides military deception into concealment, camouflage, demonstrations, feints, ruses, disinformation, lies, displays, and exploitation of insights about the enemy. [19] argues that only the last three work well for defense in cyberspace. Lies can concern system resources and status; displays can show the enemy things that aren't there; and insights can figure how best to foil an attack plan. This paper has focused on lies and insights, while other work we have done has focused on false displays.

Stage magic provides many ideas about planning deceptions [14] and some of these provide lessons for computer systems [24]. To perform magical feats, at least one deception must occur in an act, so the magician's goal is to conceal the necessary deceptions as much as possible. Tactics include creating dramatic structure, using consistency in theme, manner, and characterization, controlling pacing, controlling attention of the audience, using words and appearances carefully, and using special magic devices. Nelms pays special attention to "reducing departures", those things necessary to the deception but which can arouse suspicion in the audience. For instance, the magician may need a subject to choose a particular card from a deck; having the magician supply their own unshuffled deck would be an implausible departure, but having the subject inspect and shuffle the deck and then substituting a different deck surreptitiously would be a lesser departure. Nelms' analysis has inspired our approach of estimating the implausibility of deceptions in planning.

For information security, defensive deception has first been done for honeypots; [5] and [12] provide two approaches to building them. Honeypots can be easy to recognize without deception, since attackers can easily see a lack of normal file structure and lack of temporary files indicating activities like email, Web, and other forms of Internet use. Most hackers today have heard about honeypots, and will recognize these symptoms and leave if they encounter them. This prevents defenders from collecting useful data on them.

Because of this, [5] was first to propose deliberately deceptive activities on honeypot networks to keep attackers busy. One way is to change the router to recognize large numbers of fake IP addresses so the attacker will waste much time attacking virtual systems. The virtual systems could map to the same storage system, or virtual storage could be generated according to a stochastic grammar. Information for hackers to discover can also be manually created, and revealed in stages, to keep them interested [6]. How long will this fool an attacker? Probably not long, because it is hard to simulate an entire busy computer system, but it helps for defending critical systems.

Other projects in information security are beginning to examine deceptive tactics for defense. [10] examines automatic methods for creating fake documents for spies. [21] suggests delaying responses to suspicious commands to an operating system. Planning against an adversary has been introduced as "counterplanning" by [4] and applied to military settings by [1]. Finally, [7] provides an interesting alternative model of deception based on ideas from natural-language processing that deals more with the reasons why deception works than our effects-based model.

5. Conclusions

We have provided a theory for deception planning in defense of information systems. Our approach is to try to convince the attacker of a "generic excuse" which means that his or her attack plan cannot succeed, so they will give up and go away. To be effective, we must carefully plan when and how to deceive, while monitoring the attacker's beliefs in our proffered excuses. The methodology proposed here is more likely to convince an attacker than broad and unselective deception as with honeypots, and more likely to defend our systems. But we need to do further work to test reactions of people to these deceptions.

6. References

[1] Applegate, C., Elsaesser, C., & Sanborn, J., "An Architecture for Adversarial Planning," *IEEE Transactions on Systems, Man, and Cybernetics*, 20 (1), January/February 1990, pp. 186-194.

[2] Bell, J. B., & Whaley, B., *Cheating*, New York: Transaction Publishing, 1991.

[3] Bok, S., *Lying: Moral Choice in Public and Private Life*, New York: Pantheon, 1978.

[4] Carbonell, J., "Counterplanning: A Strategy-Based Model of Adversary Planning in Real-World Situations," *Artificial Intelligence*, Vol. 16, 1981, pp. 295-329.

[5] Cohen, F., "A Mathematical Structure of Simple Defensive Network Deceptions," at http://all.net, Info-Sec Baseline Studies, 1999.

[6] Cohen, F., & Koike, D., "Leading Attackers through Attack Graphs with Deceptions," *Computers and Security*, Vol. 22, no. 5, pp. 402-411, 2003.

[7] DeRosis, F., Castelfranchi, C., Carofiglio, V., & Grassano, R., "Can Computers Deliberately Deceive? A Simulation Tool and its Application to Turing's Imitation Game," *Computational Intelligence,* Vol. 19, No. 3, 2003, pp. 235-263.

[8] Dunnigan, J. F., & Nofi, A. A., *Victory and Deceit, second edition: Deception and Trickery in War*, San Jose, CA: Writers Club Press, 2001.

[9] Ford, C. V., *Lies! Lies!! Lies!!! The Psychology of Deceit*, Washington, DC: American Psychiatric Press, 1996.

[10] Gerwehr, S., Weissler, R., Medby, J. J., Anderson, R. H., & Rothenberg, J., "Employing Deception in Information Systems to Thwart Adversary Reconnaissance-Phase Activities," PM-1124-NSA, Rand National Defense Research Institute, November 2000.

[11] Heuer, R. J., "Cognitive Factors in Deception and Counterdeception," In *Strategic Military Deception*, ed. Daniel, D. C., & Herbig, K. L., New York: Pergamon, 1982, pp. 31-69.

[12] The Honeynet Project, *Know Your Enemy*. Boston: Addison-Wesley, 2002.

[13] Josang, A., "A Logic for Uncertain Probabilities," *Int. Jnl. of Uncertainty, Fuzziness, and Knowledge-Based Systems*, Vol. 9, No. 3, June 2001, pp. 279-311.

[14] Nelms, H., *Magic and showmanship: A handbook for conjurers*, Mineola, NY: Dover, 1969.

[15] Nyberg, D., *The varnished truth: truth telling and deceiving in ordinary life*, Chicago: University of Chicago Press, 1993.

[16] Park, H. S., & Levine, T. R., "A Probability Model of Accuracy in Deception Detection Experiments," *Communication Monographs*, Vol. 68, No. 2 (June), 2001, 201-210.

[17] Proctor, P. E., *Practical intrusion detection handbook*, Upper Saddle River, NJ: Prentice-Hall PTR, 2001.

[18] Rowe, N., "Counterplanning Deceptions to Foil Cyber-Attack Plans," Proc. IEEE-SMC Workshop on Information Assurance, West Point, NY, June 2003, pp. 203-211.

[19] Rowe, N., & Rothstein, H., "Two Taxonomies of Deception for Attacks on Information Systems," *Journal of Information Warfare*, Vol. 3, No. 2, July 2004, pp. 27-39.

[20] Snyder, C. R., Higgins, R. L., and Stucky, R. J., *Excuses: masquerades in search of grace*, New York: Wiley, 1983.

[21] Somayaji, A., & Forrest, S., "Automated Response Using System-Call Delays," Proc. 9th Usenix Security Symposium, August 2000.

[22] Sztompka, P., *Trust*, London: Cambridge University Press, 1999.

[23] Templeton, S., & Levitt, K., "A Requires/Provides Model for Computer Attacks," Proc. of the New Security Paradigms Workshop, Cork, Ireland, September 2000.

[24] Tognazzini, B., "Principles, Techniques, and Ethics of Stage Magic and their Application to Human Interface Design," Proc. Conference on Human Factors and Computing Systems (INTERCHI) 1993, Amsterdam, April 1993, pp. 355-362.

[25] Whaley, B., "Towards a General Theory of Deception," *Journal of Strategic Studies*, Vol. 5, No. 1, March 1982, pp. 179-193.

Acknowledgement: This work was supported by the U.S. National Science Foundation under the Cyber Trust program. Views expressed are those of the author and do not represent policy of the U.S. Government.

7. Appendix: Questionnaire and average responses

Instructions: Suppose you are a hacker attacking a computer system. At each step of the following sequences, estimate the probability (on a scale of 0.0 to 1.0) that you have been deceived by the computer system in some way. You can take into account the previous steps of the sequence, but not *the steps of different sequences.*

Sequence 1: You log into a computer system. It says that system testing is occurring and it may not work properly.
(1a) Deception probability: .421
You edit a text file in your own directory and it gives a strange error message.
(1b) Deception probability: .436
You try to transfer this file to another site on the Intranet and it says the network connection is down.
(1c) Deception probability: .450

Sequence 2: You log into a computer system. It says that the network connection is down.
(2a) Deception probability: .318
You try to copy an executable in your own directory to the system binary directory and the last character of your command is ignored.
(2b) Deception probability: .546
You try to transfer this file to another site on the intranet and the last character of your command is ignored.
(2c) Deception probability: .580

Sequence 3: You log into a computer system. Messages are normal.
(3a) Deception probability: .187
You try to copy an executable in your own directory to the system binary directory and it gives a protection violation and says it cannot save it.
(3b) Deception probability: .318
You try to transfer this file to another site on the Intranet and the last character of your command is ignored.
(3c) Deception probability: .536
You try to list your home directory and the last character of your command is ignored.
(3d) Deception probability: .568

Sequence 4: You log into a computer system and copy a file from a local intranet site to your home directory on the computer system. All messages are normal.
(4a) Deception probability: .182
You try to copy this file to the system binary directory and you get a protection-violation message.
(4b) Deception probability: .296
You issue a command to a system utility with an unusual long argument containing lots of nulls. The system gives a protection-violation message.
(4c) Deception probability: .391
You try to transfer another file from the other site on the Intranet and it says the outgoing network connection is down.
(4d) Deception probability: .694

Sequence 5: You log into a computer system and copy an executable file from a local intranet site to your home directory on the computer system. All messages are normal.
(5a) Deception probability: .276
You do a listing of the directory you copied it to and do not see it listed.
(5b) Deception probability: .513
You try the file transfer again. You do a listing of the directory and again do not see it listed.
(5c) Deception probability: .604
You try to copy the password file to the intranet site. The system says the outgoing network connection is down.
(5d) Deception probability: .623

Sequence 6: You use a buffer overflow to enter a computer system via port 445 with root status. You try to list the home directory and get a message "System down".
(6a) Deception probability: .673
You ask what your login name is and it refuses with the message "System down".
(6b) Deception probability: .723
You list the main binary directory with no error message.
(6c) Deception probability: .441
You ping port 80 (HTTP) and it appears to be running normally.
(6d) Deception probability: .427