A Treatise on Informational Warfare

Written by Eric Knight, C.I.S.S.P.

Publication First Release Date: August 20th, 2003

Table of Contents

Forward

The task of being a security researcher for me was a chosen one, there is just something fascinating about the problems that exist in the creation of new technologies that has captured my imagination when I was a child, and has continued throughout my entire career. I hope my efforts are perceived as devotion toward an ideal, not for any particular purpose except understanding the direction of the future.

Some people see in computer security a cat-and-mouse game, or a battle of wits, a chance for notoriety, or for some people with a keener perception – power. These objectives are true and obtainable. In my mind, the structure of technology and its flaws permeates the social structure of the entire world, and with it, the realization that all people will become affected by it.

The first instinct that I had, over a decade ago, was that the 'playing field' of computer security was finite. Limited vendors, limited technology, limited computer types. I spent a lot of effort trying to solve for a generalized method of identifying problems and that lead to my book "Computer Vulnerabilities" (as some people call it, "white-paper") where I created a new taxonomy for computer vulnerabilities that has held reasonably solid over time.

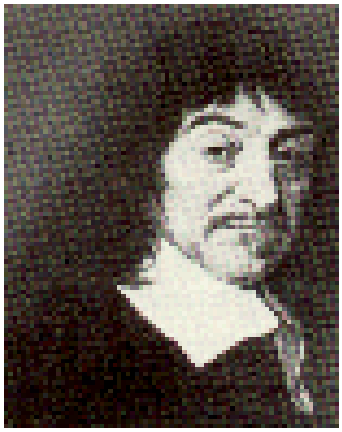
In the last 5 years, the environment of the Internet became global in all levels and the world became more educated about the Internet and its flaws. The political environment changed, and new issues emerged. Likewise, I noticed a change in strategies among international participants – hostilities are at an all time high. The landscape of technologies will continue, and the finite world of security of the past will continue to expand. The perceived finite world of the current technology is actually infinite because of competition and new invention.

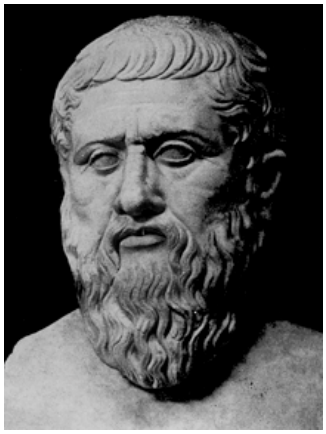
Realization of a structure that explains Information Warfare was not a flash-inspiration, it required research and time for me to prove connections and existence of particular components. Like physical warfare, all the pieces fit into place but the "final answer" is ultimately the same as all warfare.

I've added a very large number of analogies to this publication, mostly because I feel its important for people to understand the nature and connections that are more connected to us as a society. We are aware of our history, and can relate to it.

With that in mind, this "Treatise on Informational Warfare" is both a technical and educational document. I ask only of the reader to consider that the analogies and perceptions that I use to help explain are less important than the technical solutions within.

Introduction





A computer by itself is an inanimate object. Ideally, it does what it is supposed to do, processing commands from the computer's user, or from another computer, and returns calculations that can be used to represent anything imaginable.

Rene Descartes would have enjoyed this miracle landscape, a clean slate of imagination where points, lines, shapes, and imperceptible concepts all come together. Humans have perceptions, and computers do not.

Computers have proved useful for all areas of human life, from cooking your food in a microwave oven to the exact second, allowing for people to write letters and documents without spelling or typing mistakes, to helping people visualize collected knowledge in the forms of charts and graphs. They are wonderful entertainment for playing computer games and generating realistic special effects in movies and television. They improve the quality of recording and sound, they remember contacts and telephone numbers, and have vastly improved communication across the world.

One of the functions computers do is collect information diligently. With routine backups of information, information collected will never be forgotten. Newspapers age, but data is becoming timeless. "What goes on your record stays on your record."

As networks grew, so did the sharing of information about people. Credit reports and bank records are sent digitally, and connections were made to form an international collection of identities for everyone with a bank account. Later, criminal reports and legal matters were created internationally as well. Fingerprints were collected and stored for fast comparison to identify criminals from the general population without bothering to alert suspects.

E-commerce, the word of the late 90's, caused an incredible stir in the economy, building growth for new companies and promises for investors. People can purchase anything

through the Internet without leaving their homes. Marketing became simple because everyone knew they could ask questions on the vast Internet, and to purchase anything they had to provide only their identity and a credit card. This information is collected and stored on computers, and has to be handled carefully or else the purchaser may find themselves a target of exploitation.

The information collected on computers can be used to influence others, its "power". Enough power collected at one point becomes a target for those who feel its existence threatens them or can provide them with more power of their own if they control access to it. At this point, the computer itself becomes the desired target and influence over the computer becomes the desired goal.

Money, identities, customers, advertising, entertainment, sound, visuals and the perceptions to our lives they provide are only a subset of the still relatively empty imaginary landscape of possibilities inside a computer. It still feels no loyalty, no imagination, no sense of self-preservation, and no desire to protect your life from the control and influence of others. It does as it's instructed to do.

People are the influence of computers and the computers are ready to behave as the behavior of the person influencing it. Computers are easily 'tricked' into obeying the commands of the unauthorized by exploiting a vulnerability in the security system programmed into the computer.

The fact that others can influence a computer used in a situation that requires trust violates trust that people have in computers. Trust in computers is horribly misplaced, as computers have no behaviors to place trust in. The definition of "Trusted Computing" is always connected back to its users and creators.

Philosophy has connected war to almost all things desired including self-preservation. From "natural selection" to "all is fair in love and war." Its "human nature", so asking people to 'just be honest' with their construction of the hardware and software of computers is impossible without making people computers themselves.

Likewise, asking people to "be perfect" or "be honest" when it comes to the actual technology they create and distribute to others is equally impossible. It was theorized by Plato that in the universe there is an ideal form for all things, and many people hold onto this perception that something can indeed be made perfect. Our perceptions shape our universe, and if there was an 'ideal' for anything, the first question I'd ask to refute such a claim would be "what color is it?" If the other viewpoint was true, and there is an ideal form for a computer, in the absence of all desires, the ideal computer would be a simple mechanical switch.

Because of this "dismissal" or "resignation" from ideal theory that's inherent in human nature, computers are programmed with logic flaws, oversights, weaknesses, and environment limitations that conform to the needs of our environment. Typically, it's

"my environment is a little short on cash right now", but the greedy aspects of why these problems exist is not the core reason for the flaws.

Because the desire is to influence computers the initiator of the attack does not have total influence over, the attack involves an intrusion of some form. Usually through networks, by utilizing a computer against computer attack through the communication channels that computers communicate through.

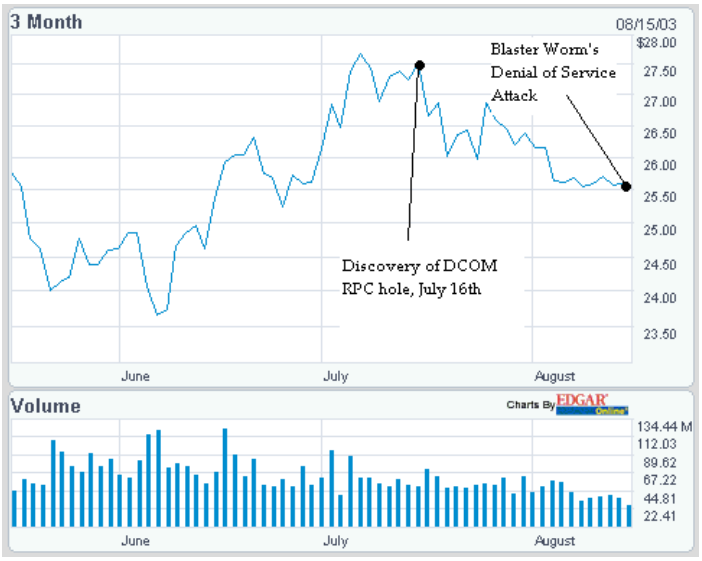
Although the battles seem bloodless and without loss of life, computers attacking computers has been a trend that has been going on for almost two decades. If real wars are fought people vs people with computers aiding, then informational warfare is fought computer vs computer with people aiding.

As is all warfare intent, it is still a people against people war and the desired results of gaining and loss of power and influence are still present. The only difference is how the computers are doing the fighting and people cannot begin to fathom the details and speed of the actual battle. The computers have no creativity in the course of the fight, so the results of the battle depend on the attackers and defenders capabilities.

"Capabilities" is a very, very broad term. Computer speed, memory, network speed, programming, environment, and purpose are all capabilities that can influence the result of an attack.

A computer has as much chance at surviving an ambush attack by another computer as a human being does surviving an ambush attack by another human being. The strategies for protecting computers are similar to protecting our own well being from hostile influences – create computer equivalents for walls, shields, locks, boxes, doors, and even self-defense techniques.

Like a mirror of the history of warfare, individual champions have become obsolete by the organization of multiple computers against a single computer. Tonight, as I write this chapter, Microsoft has been forced to terminate its update domain name and computers due to the network worm "Blaster" that will attempt a worldwide attack from against 350,000 computers simultaneously. Although the truth won't be known if Microsoft's system had the "capabilities" to survive the attack, the assumption was clear – it was going to be annihilated.



Microsoft creates the operating systems for 98% of the world's computers. They are always going to be a potential target for cyber-terrorism. Was the effect of the worm devastating? The value of Microsoft stock fell a value of $180,405,753 in the day before the attack. How about how much the value of Microsoft fell since the announcement of the security hole that lead to the creation the worm? The total loss is an incredible $20.5 billion dollars. In fact, the exact day the security hole was announced is the exact time that Microsoft's stock trend changed its direction. That is a 'significant' change of influence.

Although this attack will have an impact on society on a grand scale by present day terms, 350,000 computers will be 350,000,000 in the relatively near future, and in the far future – 350,000,000,000 computers that are far more advanced than the ones today will make up the same percentage.

Is a society with a several trillion computers reasonable? With toys, electronics, and other daily tools becoming computerized and networking in wireless ways, the numbers of computers will increase exponentially. Modern cellular telephones provide Internet access, downloadable programs, telephone network support, video cameras, keypad and speaker. They are quite usable for portable informational warfare, and there seems to be a tremendous demand for more advanced gadgetry.

The next evolution is clearly the question of using multiple computers in defense of one another. This evolution causes computers to become warriors in legion, and needs direction and response similar to soldiers on a battlefield. The questions then become, what "skills" in terms of programming do computers need? Are there ideal forms of protection? And how do emotionless, uncreative, and extremely fast entities work together?

Once again, they have to perform how they are instructed to. In this case, they have to function like people, fight wars like people fight wars. They have to become aware of

their function, their mission, their tools, and their roles. They need to have ways of communicating with each other, need to have ways of understanding commands given, and need to be able to report their findings. They also need to be able to take action on their own to identify and fight off the enemy computers, even those computers that were once trusted.

The challenge in understanding informational warfare is trying to identify all the relationships between seemingly unrelated problems. Terms have been created for each of the elements, such as "security overlap". Designing a system with proper systems in place that monitor other security components creates a stronger security system, but exactly what overlaps what, and what doesn't?

The remainder of this treatise focuses on showing the relationships of security products, models, and components in a simplified and universal form. The theory presented focuses on relationships and design of informational warfare.

The reader is advised to know that many of the tools and their functions that are described have never been developed. I did my best to explain the functions of these 'hypothetical' tools and how they interact. However, I don't believe these are works of science-fiction, I didn't write anything in this paper that I feel I couldn't program myself.

Informational Warfare Model

The concept of warfare needs to be addressed first; computers need a structure for organization. Designs for secure environments such as the Common Criteria have focused on trust relationships and secure design but not on warfare.

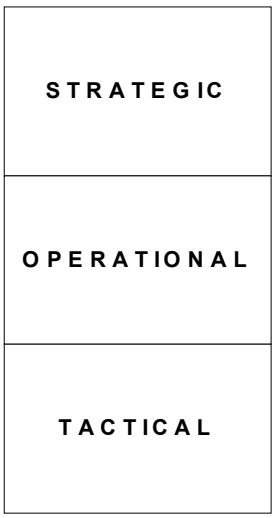
When a war is thought of, there are soldiers, weapons, leaders of many types – sergeants, lieutenants, captains, colonels, generals, and all sorts of different components for aiding the battle – medical, communication, transportation, surveillance, and has not surprisingly reached the depth of a "military specification silverware" for the battlefield.

All of these pieces of regular warfare fit a basic, three layer model of conceptualization. These are the Strategic, Operational, and Tactical layers of warfare.

There is 'strategic' warfare, where entire armies are on the move to conquer vast areas. The assessments at this level are often vague, with generalized assessments of capabilities and sizes. Communication, command, and control are most critical in order to take advantage of opportunities that present themselves in the course of a battle.

There is 'operational' warfare, where the elements of the war become important – organization of tanks, planes, soldiers, heavy artillery, supplies and medical units need to be arranged to fight in an environment effectively.

Finally, there is 'tactical' warfare. At this layer, in physical warfare, tactical efforts can be considered to be the way a person shoots a gun, hand-to-hand combat, what type of weapon is ideal for defeating a particular target, and what are the weaknesses in an opponents' weapon. Decisions need to be made such as if a grenade more useful than a rifle.



The three layers are the beginning of the conceptualization process for informational warfare, the relationships between existing tools is not clear yet from the information presented.

In terms of strategy, the existing tools of informational warfare would be 'master consoles' for reporting, Trojan horse control programs, encryption of network traffic, authentication components such as PKI that are used to identify the computers involved in the 'battle', as well as analysis programs and command functions.

In terms of operational, programs such as firewalls, anti-virus, intrusion detection systems, access control lists, and network vulnerability scanners are some of the operational considerations.

In terms of tactical, vulnerabilities are the most critical and underlying force. Without a known vulnerability, the computer will be safe from assault. However, vulnerabilities are always assumed to be present. Also, the systems needed to exploit and/or discover the vulnerabilities also reside in this layer.

The next step in conceptualization of this model is to expand on the layers in order to determine the flow of information. Each layer, and its rationalization for existence, is explained in chapters of this publication.

This representation forms the Six Layer model for informational warfare, each of the levels at this point begin to become clearly defined.

Command Layer

The command layer consists of components used to process information about the strategic battles. Visual consoles, histograms, automated and manual responses for giving orders to computers involved in an attack, information storage, and analysis are all parts of the command layer.

Communications Layer

The communications layer consists of many components associated with communications between the Strategic layer and the Operational layer. Encrypted channels, private channels, protocol handlers, authentication and verification of identity are all elements that exist in the communication layer.

Agent Layer

The agent is like the "soldier" application, responsible for activating the components necessary for fighting its battles. It must communicate with command, and be adept at controlling security functions. Idealistically, the agent must be responsible for providing as concrete, processed, and accurate information as possible to the command layer in order to lesson the burden of processing.

Functional Layer

The functional layer consists of the security components engineered for a task. A firewall performs the task of filtering network traffic, and therefore provides a complete security function. An anti-virus program performs the task of keeping the computer free from viruses. Functional tools are collections of facilitators.

Facilitators Layer

Conceptually, all facilitators are inherent to the security environment they were designed for. They are programs that do very little on their own except perform a very specific task that is a subset of a larger security issue. They are not vulnerabilities, and they are not complete tools. A tool that changes a firewall setting is a facilitator, or an exploit that takes advantage of a computer vulnerability is a facilitator.

Vulnerabilities Layer

Vulnerabilities are the 'atomic' level of computer security problems. They fall under the headings of social engineering, policy oversight, logic errors, and weaknesses. They have severity, consequence, cause of fault, a tactic for exploitation, and authentication requirements. On their own, vulnerabilities do nothing until manipulated by facilitators.

The six layers can be then conceptualized by showing types of tools associated with each layer in the model:

STRATEGIC	COMMAND	REPORTING	LOG ANALYSIS	LOG REPOSITORY	SCHEDULING
	COMMUNCATION	ENCRYPTION	NETWORK TRANSPORT	SECURE CHANNELS	PKI
OPERATIONAL	AGENT	LOG REDUCTION	EVENT COLLECTION	MONITORING	PROFILING
	FUNCTIONAL	FIREWALL	ANTI-IVIRUS	INTRUSION DETECTION	FORENSIC
TACTICAL	FACILITATORS	LOGGING	FILE SEARCHING	SYSTEM HEALTH	ACCESS CONTROL LISTS
	VULNERABILITIES	SOCIAL ENGINEERING	LOGIC ERRORS	WEAKNESS	POLICY OVERSIGHT

Inherent Layer Characteristics

The following characteristics are inherent in the model:

- All layers must have some form of awareness of a level 6 problem or else they serve no purpose
- All layers must interact with at least a single component from the layer below them.
- All layers cannot directly interact with a layer that isn't adjacent – that is, an agent (layer 3) cannot look directly for a vulnerability (layer 6) without having at least one functional tool (layer 4) and one facilitator (level 5). I.E., even if its only one line of code, the request itself is a layer 4 component, and the process of proving if the vulnerability exists is a layer 5 component.
- The definition of "security overlap" (that is, software that monitors the health of a different security component) is the responsibility of the layers above it. Ideally, the responsibility of determining the health of a component falls immediately on the layer above it.
- Layer 1 has no 'ceiling', it may respond to higher levels as necessary by utilizing the Level 2 layer functions to communicate with a superior or redundant authority.

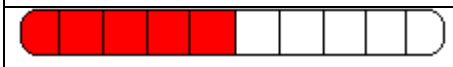
Layer Design Idealisms

Each of the layers in the present day are filled with tools that utilize their own code without sharing the layers below with other tools. Although this doesn't pose a problem with properly written components, if the model were to be followed exactly, the performance, security and reliability of the model structure can be improved.

- Idealistically, information flow should be added to all information detected between level 6 to level 1.
- Most facilitators can be used by other security components, such as Windows™ Registry access, file access, ACL access, and other environment specific tools. A tool that finds an exploit can be used for internal scans, external network scans, and patch management tools.
- Layers that have 'security overlap' functions (such as a file integrity system watching a virus scanner) have ideal performance if they don't have to monitor multiple versions of the same functional component.
- Layers should have omni-directional controls for the layers above and below them.
- Level 1 should be a single entity. In 'the real world', it isn't possible for a single computer to control billions of computers across the globe and be fully aware of all the events that transpire. However, if there was a single computer powerful enough to do this task, it would require the least amount of computation time and communications to perform the task compared to sharing the responsibilities with millions of other computers.

Effectiveness Measurements

Throughout this document, many of the components are measured in some capacity to the level they will assist the security network. .

	Anything with a level of 1 is not good. It is either slow, tedious, offers low protection, or otherwise has a considerable drawback.
	A level 5 indication means that there is a tradeoff taking place, but there is a degree of effectiveness. This may mean that the process is time consuming, speculative, or not entirely effective given the situation
	This effort is extremely fast or extremely effective.

> In terms of **complexity**, the indicator means the **opposite**. A full red bar means that the mechanism is difficult to operate. Typically, this means that either a lot of preparation or human influence is required!

The absence of a security tool or technique is definitely NOT a benefit to the security posture of the Informational Warfare Model. Even a system that is not very effective, time consuming, and complicated is better than not having the system at all.

The values for each element are highly subject to debate. Due to the lack of actual supporting data for each "measured" element, experience and a short mental 'game strategy' was done to at least produce an 'indication'. Like all warfare components, the real proof will come in the actual battles, and an ineffective tactic may become more or less effective depending on the environment.

If you are trying to determine the overall strength of you own security posture, and are using a person to handle the attack, it can be assumed that the complexity level will always be high, the speed will always be the slowest possible, but the effectiveness will be the maximum level. It is always possible to throw man-power to a given problem, but a single computer will outperform calculations done by human effort.

To consider, a security tool such as Internet Security System's "System Security Scanner" (bundled with Microsoft Windows™ Resource Kit), hundreds of security settings can be checked in seconds. For a human to perform the same tasks without the assistance of a tool, it would take days.

The logical approach toward Informational Warfare is having at least -something- capable of helping turn the 'tide of battle' in case the need appears.

This treatise contains detailed descriptions of each of the security layers, components, interactions, events, and warfare evaluations. Calculations, estimations, and structures for many components of each layer are given, as well as effectiveness against targets (Human vs Agent, Agent vs Agent, Security Network vs Security Network, and so on.)

Command Layer

The first layer of the informational warfare model, the Command Layer, is the center for control of a collection of agents. There is a myth of this being simply a person behind a computer screen acting like a military General, but the actuality is that in a real computer attack, the human influence will not be fast enough to control all the subcomponents necessary and automation will be at least 95% of the battle.

The Layer 1 model consists primarily of componant that determine the nature of computer attacks and lend support to the agents residing at Layer 3, while protecting the assets of the ***security network***.

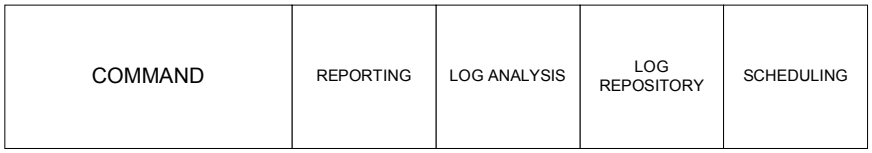
> **Security Network** is a term used throughout this document that refers to all elements between Layers 1 and 3 of the Informational Warfare Model that are connected together. This is a logical, not a physical network.

There are a nearly infinite number of elements that could exist in the aide of command and control. Where the vulnerability layer (Layer 6) is reasonably Boolean in conceptualization (either the problem is there or it isn't there), Layer 1 has to consider the collaboration of all vulnerabilities, all events, all computers, all users, and as all points in time that are measurable. It needs to be concerned about when a problem started, when a problem was discovered, when the problem was reported, and if and when the problem was corrected, how the problem is corrected, the nature of the problem, recording the problem, responding to the problem, and the reaching impacts of the problem. That is, presuming the 'problem' is even a problem at all.

Ideally, the Layer 1 solution will be able to take all information collected at Layers 2-6 and be able to process them for the Enterprise. It should be able to reply every security even in the Enterprise like a video recording. This requires an intense amount of computing power and programming. The reality is that Layer 1 relies on the abilities of the other layers to create 'digested information' that is easier for it to do calculations on. It may still collect tremendous amounts of raw information from lower layers, but call on this information only when the situation requires it (or it has enough free cycles to do 'final processing' before purging the information.)

Layer 1 can be a single computer, but often is not. With present day implementations, Layer 1 is often a collection of computers handing specific tasks – backup facility, database, log collection, master console, analysis computer and collection points.

Many of these pieces are not connected except my human intervention, and in the next 5 years the existing security systems are going to seem primitive and disconnected in comparison to unified and specialized systems.



The command layer is filled with components that work with each other, such as the example above. Each piece has the purpose of having an interaction with either elements of the same layer, or directly with Layer 2, the Communications layer.

The most basic components for an ideal Layer 1 system would include the components listed below.

Command Console

This console is responsible for the human interaction involved with directing Layer 1. This interface should be able to show processed reports, real-time analysis, security events. Likewise, it should allow for sending detailed commands to agents and allow configuration and examination of all elements in the Security Network.

Log Repository

The Log Repository is where all the logs, notifications, etc. of security related events are stored. This system will have the responsibility to fetch, receive, and archive logs. The primary component of this is a simple database tailored to the format of the logs being stored.

Analysis Components

There isn't a true concept for analysis components except that they are meant to simplify understanding and identification of events. They are often basic components such as a "delta" filter, showing changes in security posture, playing a severity on problems, identifying relationships between attacks and attackers, and so on.

History Analysis

History analysis has show to be incredibly time consuming and most of the effort of cross-relating events has to do with solving time considerations. Consider that the following time-critical events all need to be considered when assessing the critical nature of a security problem:

- When was the problem discovered?
- When did the problem get entered into the system?
- When was problem first actively exploited?
- When was the problem reported after it was discovered?
- When was the problem processed?
- When was the administrator notified?
- When was the problem eliminated?
- When was the problem "worked around" if it wasn't eliminated?

Time is a hard concept for computers to grasp, they "think" linearly, while our calendar and time is circular. People expect attacks to happen on Friday or Saturday around midnight, or maybe during Christmas Day or New Year's Day when nobody is in the office and nobody would come to work even if it was burning down. Computers don't associate these periods with higher risk of danger unless programmed to, and programming the awareness of these "special times" is a complicated effort. During some periods of time, such as daylight savings time or leap year on February 29th, the analysis effort can become very muddled with coding mistakes.

Game Theory

The origin of game theory in computer security started with a tool called "Kwang" that started with a simple premise on the access control list of UNIX computers and tried to advance its level of security to 'a desired target', usually 'root'. The implementation of game theory can be used in Layer 1 much the same way by creating war simulations based on the security environment.

Elements of "the game" can be summarized by standard Risk Management or an access level, assigning values to computer elements and letting the computer determine if new vulnerabilities discovered pose a significant risk. Any time a "winning strategy" appears,

the administrator needs to be aware that a possible and known breach of security can occur.

The environment is more complicated than a chessboard with a lot more "pieces" to consider. Two approaches toward creating the ideal game theory engine would be to either write better software to 'think' the process better, or utilize faster equipment to decrease the processing time. The primary use of game theory is preventive, and probably won't be active response for a considerable time.

Expert Engine

Although game theory is an idea well ahead of its time, expert engines are great for cause/effect automated responses and are widely used in the present. It is a collection of rules to handle special situations such as "If a host is compromised, then perform the following task." Rules based responses are pre-approved by human direction, and therefore allow for fast response to a critical security situation.

Unfortunately, Expert Engines will handle only the most basic of events, and the ability to proclaim situations such as "When a network attack discovered that places an agent at the host, perform then attempt to perform the following actions, in order… but only if the agent does this… but if it responds with this, attempt this… but if it responds with this… do this… etc. etc." While waiting for each cause-and-effect response from Layer 1, the targeted computer is being pummeled by the agent that invaded. Eventually the analysis systems inside of the agents need to handle the commands themselves and not be delayed by listening to Layer 1 commands to instruct them on every detail.

Heuristic and Statistic Reporting

In most cases, the complexities of Informational Warfare are too much for an administrator to completely understand. Although the computer is warning that events are taking place, sometimes the 'events' are vague and don't appear to be important. By having heuristic and statistic reports, unusual behavior appears more obvious to the people in charge of administrating the Command Layer.

Scheduling

Not all security components are presently able to do their duties 'in real time' because of the intensity of the examination. Were a desktop computer to be actively scanning for viruses, performing forensic analysis, running intrusion detection, searching the registry for alterations, and otherwise in a constant hunt for all possible intrusions, the computer would be unusable. For 'point check' security, a regular schedule of execution and log collection needs to scheduled by the Command Layer.

Account Management

Access control and account management needs to be maintained at a centralized and secure location. This is already the normal in most operating systems, although Command Layer security is required to maintain tight control over its configuration. It is very likely that in the future developments of security server software, account management for the enterprise will be included.

Network Component Awareness

If new elements are added or removed from the network, currently they are trusted to communicate but not necessarily allowed access to everything. However, they might be a launching point for an attack that is difficult to shut down. Examples would be if an intruder entered the network though a wireless connection, or someone in the building with a notebook computer plugged into a "hot" network port.

Likewise, routes can be altered, configurations of critical network devices can be changed, and understanding the hazards of these alterations are critical to the server. Conceptually, the tools used to discover problems are Layer 4 (Functional Tools), but the 'big picture' of the network security and its policies are handled at the Command Layer.

Security Policy Management

The security policy, defined usage limits, and system health are all Command Layer functions. Agents in the security network are expected to follow the rules defined by the Command Layer as to how much access they allow the users and the tolerance levels associated with the computer use.

Security Tool Repository

Eli Whitney's interchangeable parts for guns changed the nature of warfare considerably, having a tremendous effect on the American Civil War. The same is true of the nature of security tools in the Informational Warfare Model. All Agents need to be able to call upon the tools they need, when they need them. The tools distributed to agents needs to be maintained effectively, and the most effective place to maintain them is at the Command Layer.

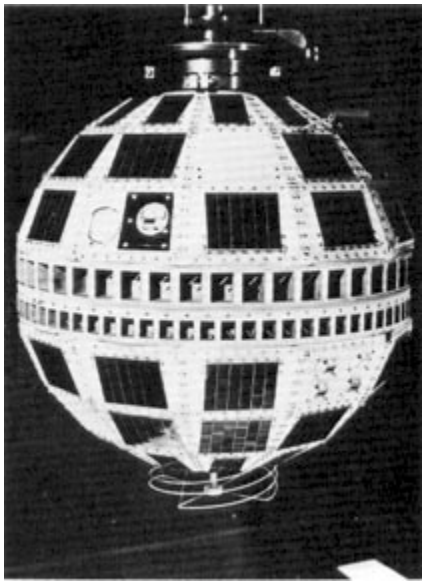
Early Warning System

The ideal construct of any Command Layer model will be to function as an early warning system for attacks. That is, to determine if there is an attack happening as quickly as possible so that a response can be generated to prevent escalation.

Having a complete log and a nice report detailing the attack against a computer that happened six hours ago and having nothing done about it is a horrible waste of intelligence gathering.

Real-time information collection, analysis, and reporting are critical for the Command Layer to allow the security network to be able to respond promptly to threats.

Communications Layer



Communications are required for maintaining a secure network environment. The contents of communications can be modified, altered, perverted, disrupted, forged, redirected, intercepted, and countless other attacks. The critical nature of communications between the Command Layer and the Agent Layer are critical.

Channel Communications

The method the communications layer has to communicate with agents determines part of the security posture of the security network, be it offensive or defensive. The method of communicate requires the selection of one or more channels of communication.

The ideal is the definition of trusted channel that stems from the Common Criteria, Part 2, Section 13, page 167.

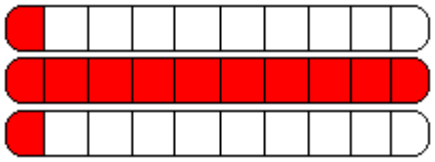
> *"...a trusted channel is a communication channel that may be initiated by either side of the channel, and provides non-repudiation characteristics with respect to the identity of the sides of the channel."*

Although the best solution is that all agents have a solid, trusted path to the Command Layer, this concept is neither easy or always useful for all situations, so several methods of channels that may be useful to information warfare are presented in this chapter.

Open Channel

In the present, most attacks are initiated from open-channel attacks, meaning that almost anything watching network activity can see the attack take place and interfere appropriately. Open channel attacks trigger Intrusion Detection Systems, and its often weaknesses in open channel environments that allow attacks to initiate. Open channel is required at some level by most organizations in order to allow outside business transactions, e-mail, web service, and other common network activities.

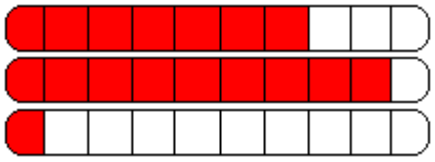
Protection: 1
Speed: 10
Complexity: 1



Secure Channels

A secure channel is a method of communication that is meant only between the Command Layer and the Agent Layer and is highly resistant to eavesdropping attacks. This is usually done through encryption. Virtual Private Networks/IPSec are a good examples of a secure channel.

Protection: 7
Speed: 9
Complexity: 1



Isolated Channels

A trusted channel is a path for communication that maintains a degree of trust of security between the two points, that is, it has a very obvious reason why it cannot be different than what is expected of it. For example, it's a direct wire between the two systems.

Protection: 9
Speed: 10
Complexity: 1

Covert Channels

A covert channel is a path of communication between two systems that doesn't appear to carry the traffic expected by two security systems. It hides the information being sent in some matter. This can be embedded in the TCP/IP stack as set bits or through creative use of out-of-bound communication.

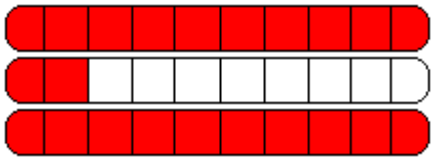
Protection: 5
Speed: 5
Complexity: 5

Polymorphic Channels

A polymorphic protocol is a covert channel that changes it nature periodically, that is, it deliberately might send information in one channel method, then switch to another channel method, then switch again to another channel. In the process, it will change the nature of the meaning of the received traffic. That is, 8 bits of received information from one channel technique might translate differently than the previous 8 bits from the same channel. The higher the frequency of switching and the greater degree of randomness will cause the channel to be nearly impossible to follow linearly or by timestamp.

The downside to polymorphic channels is that if an error occurs, the communication path falls out of syncronization, and the channel will have to restart itself in some fashion while the that didn't experience the error waits and ultimately times out.

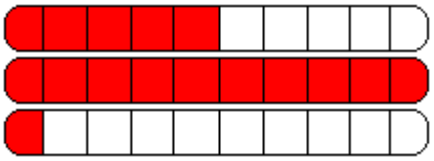
Protection: 10
Speed: 2
Complexity: 10



Alternative Channels

Alternative channels can be arranged so that if one channel is attacked, compromised, or rendered inoperative, the security network remains intact. An opposing security network that is invading a network usually needs the main open channel to operate, so attacks against the channel are usually contrary to its mission. However, in many cases, having a secret 'back way out' for a network can help considerably and redundancy is good.

Protection: 5
Speed: 10
Complexity: 1

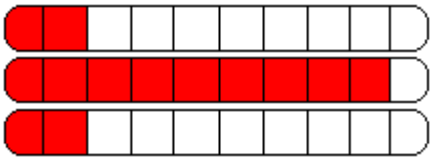


Switching Channels

Switching Channels is a form of covert communication often used in radio technology. The communication quickly changes randomly between existing channels. For radio, this may mean communication on frequency 55.1, then move rapidly to 55.8, then down to 55.3, etc. in a random pattern that shares a common 'seed' between the two systems.

In terms of network use, switching between channels can be difficult because unlike radio, they don't share the same properties. Latency time increases as switches are made between network paths. Changing of ports and sockets can add some confusion to a collection system, but since they are all on the same open channel without a 'frequency' to search, the packets can be collected in a time sequence and reassembled trivially.

Protection: 2
Speed: 9
Complexity: 2



Public Key Infrastructure

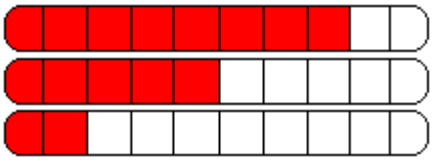
Public Key Encryption and Public Key Infrastructure add significant security enhancements to the security network. By using these methods, trust relationships can be established through mathematical proofs, and stronger 'conventional' keys can be distributed between hosts securely. The downside to PKI is that it needs to be managed and keys need to be generated and distributed through a Certificate of Authority in order for them to be used –properly-.. That is, they can also be used to increase security without a CA, but they become vulnerable to attacks and trust relationships become rendered useless.

Protection: 10
Speed: 8
Complexity: 5

Conventional Encryption

By itself, conventional encryption applied properly in the security network can be an astounding effective system. When used not as a part of the channel, it can introduce an additional layer of protection for security data moving between Layer 2. Arrangement and protection of conventional keys does present a significant security challenge to keep them from being compromised or stolen, and so use of conventional encryption is not without some potential concerns.

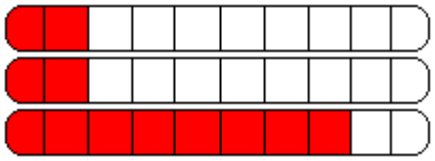
Protection: 8
Speed: 5
Complexity: 2



Trust Relationships

Besides the server, other forms of trust relationships can be programmed into the communications system. For example, it determines which hosts are allowed to communicate on the network. Early examples of this are done at enterprise firewalls where authentication with user names and passwords are required to communicate with other computers. Trust relationships are explicitly defined and managed. Without mathematical verification, these systems are susceptible for impersonation.

Protection: 2
Speed: 2
Complexity: 8



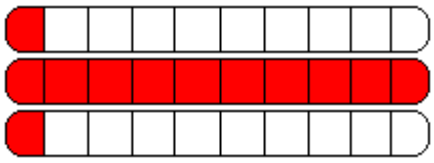
Protocol

The method in which Layer 1 and Layer 3 communicate needs to be defined in a way that fits one or more protocol standards. The protocol is the way the components communicate, such as "I'm sending you a file", or "perform this instruction". Ideally, this information should not be easy to understand in case the channel security is compromised.

Uniform Standard Protocol

This protocol is a standard communications channel, always consistent, and always of the same format.

Protection: 1
Speed: 10
Complexity: 1



Covert Protocol

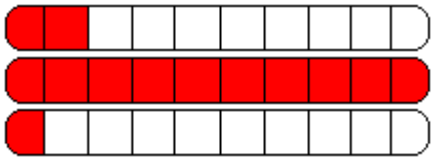
This protocol is embedded inside of another protocol, such as commands are received through e-mail or World Wide Web. Covert protocols allow information to pass through many security systems that monitor open channels.

Protection: 5
Speed: 5
Complexity: 5

Alternative Protocol

An alternative protocol may be used to confuse eavesdroppers but the effect is only slightly more useful than the original protocol. However, the increased diversity does require attackers to the ability to understand all alternative protocols, and that can be difficult for an outside agent to handle.

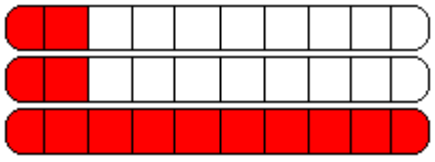
Protection: 2
Speed: 10
Complexity: 1



Polymorphic Protocol

Related to polymorphic channels, polymorphic protocols will change their format and structure based on a seemingly random order. However, the protocol itself is much more vulnerable to interpretation than the channel is, and therefore isn't quite as effective.

Protection: 2
Speed: 2
Complexity: 10



Agent Layer

The Agent Layer is best considered as the 'soldier' of informational war, utilizing all the tools and components at its disposal to best attack or defend the computer. Agent programs of the current day are very simplistic and have virtually no awareness of much more than a single tool and an ability to communicate.

In ideal practice for informational warfare, an agent is given its tools of war (anti-virus, firewalls, forensics, log analysis, intrusion detection, searching tools) and set forth to guard or attack according to its function.

The difference between a human agent and a computer agent is obvious; computer agents lack the great sophistication of sensory perception and neural awareness. They need to be programmed with an artificial survival instinct, as well as "eyes", "ears", and a degree of intelligence.

As difficult as that is to fathom, in the present day technology the pieces exist to create the framework for the agent. The eyes are forensic tools, the ears are intrusion detection and firewall tools. Eyes and ears are an analogy, however. Sensory perception for the Agent will be limited to the detection components, that is, "File System Awareness", "Network Traffic Awareness", "Memory Awareness", "Access Awareness", "Process Awareness", and so on. A very good measure of the effectiveness of an agent is the depth of its perception and the number of sensors it possesses.

Being inside of the "imaginary landscape of possibilities" inside of a computer, there is no limit to the number of possible perceptions an Agent can have, the only real limitation is the capabilities of the computer and its programming to support having awareness. This creates the problem of what happens when an agent exceeds its own environment and creates "agent overload".

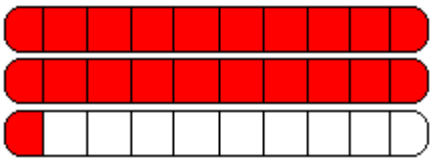
Having an agent perform the security for a host is advantageous compared to having a human controlled 'puppet' agent. Agents are fast, methodical, and reside entirely on a

single computer. The tactics Agents have to prevent human influence are numerous. Agent against Agent battles are considerably more difficult. See the chapters on **Agent vs Human** and **Agent vs Agent** combat.

Command Interface

The command interface is the system in which the agent handles commands that arrive from Level 2. The more robust the command interface, the more actions can be done to protect the host. Likewise, controls need to be placed on the command interface to prevent unnecessary commands (such as five of the same task being performed at a time). The command interface needs to be flexible but not complicated. The ability to maintain and control multiple commands rapidly is highly desirable before other commands complete, and being able to handle both automated and non-automated requests is ideal.

Protection: 10
Speed: 10
Complexity: 1



Host Console

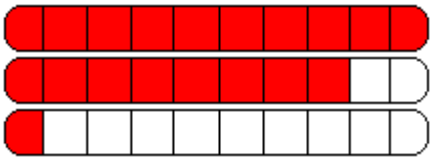
When the agent has a significant problem that requires a disconnect from the security network for human interaction, a console application should be present that doesn't require network resources to operate. This is mostly a design consideration and does very little to effect the host security posture or increase warfare performance.

Protection: 1
Speed: 1
Complexity: 7

Response Reporting

When a task is completed, the results of the tasks should be reported to the Communications Layer. This is usually a simple matter, with a simple reply such as "task <id> completed". In cases where there is a Communications layer and no Command layer, the reporting should be much more specific, possibly including a list of information retrieved such as "discovered passwords: id:pw, id:pw, id:pw, id:pw".

Protection: 10
Speed: 8
Complexity: 1



Mission Intelligence

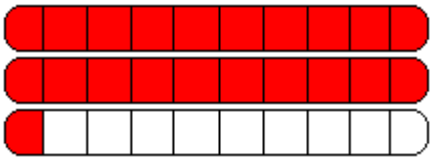
Mission intelligence is a catch-all for achieving the purpose for which the agent was intended. All agents are implied to have a mission, but they can also be defined and programmed into the agent in case a mission needs to change. See the chapter on "Mission Goals" to understand more.

Protection: 3
Speed: 3
Complexity: 10

Process Control

An agent should have the ability to manage the execution and termination of security functions. This is required to use any external tools that the agent is integrating with. This includes being able to know when a tool is processing, failed to execute, and all needed security verification functions that provide overlapping security.

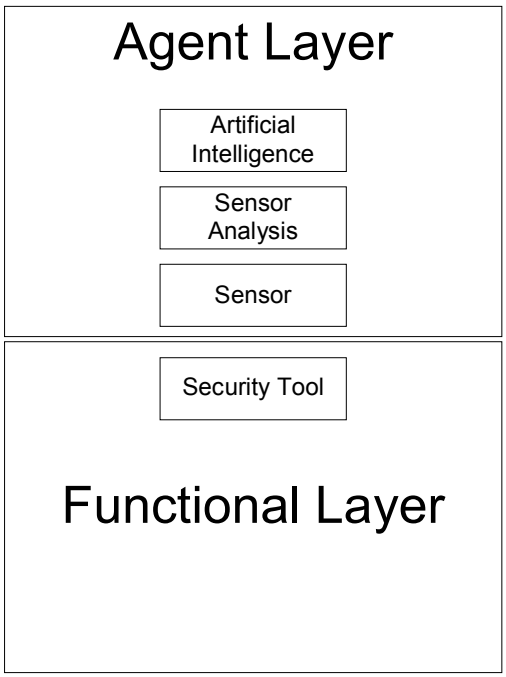
Protection: 10
Speed: 10
Complexity: 1



Sensors and Sensor Analysis

The process of managing sensors and their processing of information collected is an Agent Layer function. To imagine the situation for a soldier on the battlefield, they are told by radio that the enemy approaching to the north and they need to retreat, and also hearing gunfire from the east where they also suspect more of the enemy is located. Quickly, the soldier decides the safest retreat is to move to the Southwest. Similar 'processes' need to be handled by the agent as well, logical responses to information that is reported to it.

The effect of sensors and sensor analysis can be showing in the informational warfare model as follows:



Agent Sensors

Agents must be aware of the Layers below it (4-6) or else its completely blind. Agents can have very simple sensors, such as "does this one vulnerability exist?" Agents with a limited sensory perception are fast, focused, and not capable of determining if they are in danger of failing their mission.

In the current levels of technology, the ideal of "faster is better" may be true, but if a system becomes attacked and the defending agent can dispose of the attacker in one minute after the millisecond attack took place, that is a tremendous speed improvement over existing "days waiting" for human processing to solve the problem.

The types of sensors required by an agent should be limited to control over the computer environment, and typical sensors for prototype agents would be:

- Intrusion Detection Sensor
- File Change Sensor
- Virus/Malicious Code Sensor
- Computer Abuse Sensor
- Configuration Change Sensor
- Network Awareness Sensor
- Network Host Vulnerability Sensor
- Access Control List Sensor
- Process Sensor
- Memory Sensor
- Security Network Status Sensor
- File Content Sensor

Many of the sensors are repetitive in tools, for example, file content sensor may be performing the task of searching for files that contain 'dirty words' (military terms for classified information), but can also be used for discovering viruses and malicious software, web service abuse, and so on.

Future design of Level 4-6 tools will be determined by how well they improve the quality of the sensor, but for the time being combinations of tools and information pulled from different tools will be used to consolidate into a single sensor at the cost of performance due to repetition in searches and additional calculation time.

Is it possible to have a single "File Sensor"? Or a single "Network Sensor"? In future models, I suspect that the sensory perception will be more categorized to the environment functions.

- File Sensor (All possible file contents)
- Access Control List Sensor (User Accounts)
- Configuration Sensor (Windows™ Registry, UNIX configuration files)
- Memory Sensor (process control, memory, virtual memory)

In this case, the sensors will gain depth of analysis through its artificial intelligence components.

Point of Clarification: The Agent isn't responsible for discovering the information collected from the sensors, it is responsible for processing, analysis, and responding to the situations discovered by the sensors.

Sensor Analysis

Each sensor needs to be able to have analysis performed on it to determine the nature of the attack. In most cases, the Functional Layer provides this analysis. However, the Agent can look for additional details and add or remove important details about the environment and from other sensors. Each sensor should have a corresponding analysis component. This process should be entirely factual in nature, not implying any forms of "fuzzy" logic.

Artificial Intelligence

At some point during the process of a threat, the agent needs to make decisions based on the nature of the threat. The attack itself will probably have an identifying signature the signature can be used for a response.

Using chess programs as an example, they are often programmed with the 'time honored' opening strategies. Where a computer may recognize the movement of a pawn, then a knight, and then another pawn and conclude it can follow a recognized strategy and follow that strategy, saving computation time, until the game progresses beyond the pre-programming.

Like the game theory engine present in the Command Layer, the A.I. for the Agent needs to be able to play its own 'war games' at Layer 4-6. By programmed 'instinct', the A.I. will attempt to hunt for unknowns.

If the A.I. is utilizing game theory, memory will be allocated that allows the agent to theorize more about its attacker and how to remove it. For example, if one sensor identifies the presence of a new executable file, a 'binary scan' of the file looking for procedures it calls identifies 10 "Layer 5" facilitators in the code. The agent can then measure their 'attack capabilities' based on the procedures, and then determine a course of action.

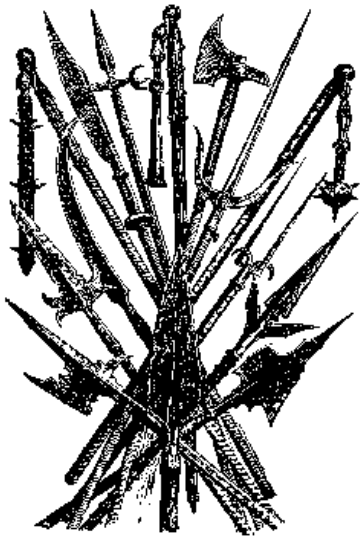
The methods of A.I. are numerous and there is much to be done in the field of researching this environment. My instinct is to focus on game trees and checklists, and use expert engine databases for as much as possible until a truly efficient A.I. is developed.

Agent Overload

The most obvious problem with agents is that they could be anything, but cannot be everything. If a soldier on the battlefield were given an airplane, a tank, heavy artillery, all the guns and rifles for all situations, land transports, sea transports, battleships, aircraft carriers, satellites, missiles, and then be asked to carry them all around with them and the mission was to spy on your target – the agent wouldn't be able to walk let alone perform its mission.

In this capacity, even if agents are capable of performing many complicated tasks at once, their design is idealistically minimal in nature. They should be given the optimal tools with the optimal performance to carry out their mission. As is demonstrated in real warfare, this is often ineffective for an individual. Support for agents needs to be provided by the Command Layer and the Communications Layer.

Functional Layer



The Functional Layer is best considered a collection of specific security tools. Many of them are robust enough to require Agent Layer properties, and overall they are not tremendously complicated in construction or design. Each component in the Functional Layer is a single or collection of tools from Facilitator Layer.

To expand on the definition of a Functional Tool, the tool must handle a specific aspect of security. The scope of the tool depends on the facilitators that it implements. For example:

An anti-virus package is a functional tool- it stops viruses.

The scope of the anti-virus package is:

- File infections
- Memory infections

The tools for locating file infections and memory infections are *Facilitators*. Another anti-virus package may also search e-mail and inspect downloaded web pages containing scripts for malicious code as well.

A brief list of existing types of Functional Layer tools are:

- User management
- Authentication Management
- File Access Management

- Anti-Virus
- Firewall
- Intrusion Detection
- Patch Management
- Vulnerability Detection
- Malicious Code Detection
- Stack Protection
- Spyware Detection
- Forensic Tools
- Policy Management
- Proxy Services
- I/O Monitoring Tools
- Encryption Tools
- System Abuse Tools
- System Health Monitoring
- Backup and archiving tools
- Database security tools
- System Logging Services
- Log reduction
- Network security scanning tools
- Backdoor / Trojan installation tools
- Content Filters / Content Identifiers
- Secure File Removal / "Undelete"
- Process Priority / Control / Ownership
- Sandboxes
- Information Guards

This list is somewhat tailored for normal personal computers in a network environment, the reality is that there are new security functions with category of computer. A cellular phone, for example, has its own additional security considerations.

Lets consider the cellular phone example – modern cell phones are essentially small computers. They have Internet access, they can record information spoken into the speaker for "personal notes", software can be uploaded to them and they connect to the telephone network as any other phone.

The components are all integrated, so is it possible for an Internet initiated attack against a cellular phone able to upload a program that allows the recording and re-broadcast of telephone conversations back out to the Internet? Could cellular phones be the next origin point for a global-scale denial of service attack? If this is true, then security elements need to be added to cellular phones – "Cellular Protection" – that is custom to the security procedures, protocols, and environment of the particular phone and service.

As new informational warfare technology is invented, the Functional Layer will expand tremendously. The tools required for agents to protect and battle each other will constitute a tremendous spectrum of this layer. For information on these tools, please read the Agent vs Agent chapter.

The properties of the Functional Layer are:

- A Functional Tool represents an aspect of security for an enterprise
- The tool is conceptually a manager of facilitators
- The ideal tool will completely handle the aspect its supporting
- The tool is responsible for the integrity of its facilitators
- The tool is responsible for producing useful output
- The tool must contain at least one facilitator
- The ideal tool will distribute information 'in real time', and 'in fastest order of discovery'. Please read the chapter on "Fastest Order of Discovery" for more information.

Conceptually, a facilitator cannot perform any task without a Functional tool. For example, if a facilitator (lets say, an scripted exploit) is activated, it is not automatically a Functional Tool. The Functional Tool in this case is the "command prompt" which provides access to the facilitator. This makes perfect sense, and disabling the command prompt is a commonly used security precaution. Essentially, for anything to reside on the Functional Layer, it needs some form of shell to house the facilitators.

If a functional tool supports more than one aspect of security, it partially resides inside the agent layer. An intelligent agent will have to separate the information from 'multi-function' tools and handle the usage of each tool independently.

Conceptually, multi-function tools can be thought of like a gun with a bayonet. It is used as both a gun and a stabbing weapon. When being used as a gun, the weapon requires a specific set of combat skills (marksmanship, loading, and cleaning.) When used as a stabbing weapon, it requires a different set of combat skills (trusting, stabbing, hand-to-hand combat.) Likewise, the agent has to consider any multifunction security tool as a combination.

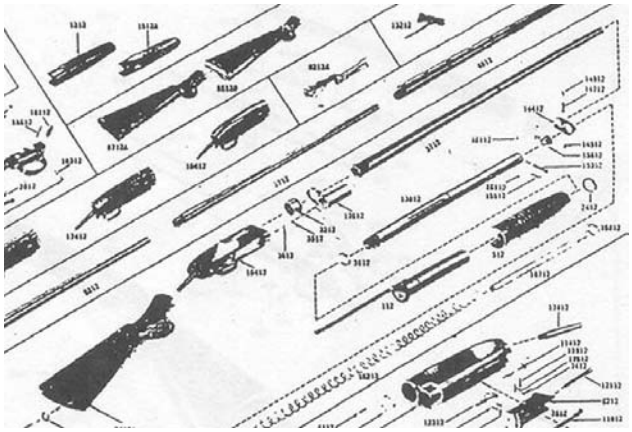
Layer Considerations

Most Functional Tools can be used both for and against the computer. If a tool that finds security problems on the host is reporting them, an intruder can "read" the report and take advantage of the knowledge.

When a hostile agent is on a host, ideally it will use any and all security tools at its disposal to dispatch, infiltrate, disrupt, and/or bypass the present security system. Some tools at the Functional Layer will be able to dispatch an agent before it has the ability to alter the host (especially, but not limited to, anti-virus and malicious software detection.)

Ideally, all elements at the Functional Layer will provide some method of prevention for being activated by non-authorized users.

Facilitators



Facilitators the simple tools needed to perform a security task, this could be an examination, a configuration change, an exploit, a search, an analysis, etc. They are the element that is most aware of a security vulnerability.

It is impossible to list even a fraction of the possible facilitators in this document, as there are tens of thousands already created. Many facilitators are recyclable for other uses, and many are unique to a special situation.

For example, a facilitator that looks for a file signature to determine a vulnerability can be connected with a database to search for many vulnerabilities. However, a facilitator that does the task of speeding up the DES algorithm for finding weak UNIX passwords will probably be used only for that purpose.

Facilitator Layer ideals:

- Information should be time stamped
- The nature of the vulnerability(s) discovered should be tagged as part of the result
- All information pertaining to the vulnerability should be disclosed
- Code should be optimized for greatest performance
- Errors should be handled properly and failures notified
- Code should be as concise as possible

The primary concept behind all facilitators is that they are going to be used by other programs, idealistically, by more than just one.

To demonstrate the need for facilitator reuse lets use the example of a password cracking program from the Functional Layer. It is designed for attempting to find weak passwords for DES. It has the following facilitators:

- A dictionary reading program
- The DES algorithm
- A tool for comparing DES results to dictionary results
- A user list parser that extracts the encrypted password
- A dictionary filter

To add more robustness to the tool, if the facilitator for MD5 secure hashes is going to be added. If the tool for comparing DES results is written properly, it can also use the MD5 algorithm. Also, LM-Hashes, SHA-1 hashes, etc. can be added later. The other facilitators can be reused.

Later, more facilitators can be added to the package, such as network protocols and brute force sequences. The user list parser can have substitutes for extracting passwords from the Windows™ Registry, password protected files, and so on. This adds great conceptual power to Level 4, because the entire 'conceptual power' of finding weak passwords falls upon a single Functional Tool.

Fastest Order of Discovery

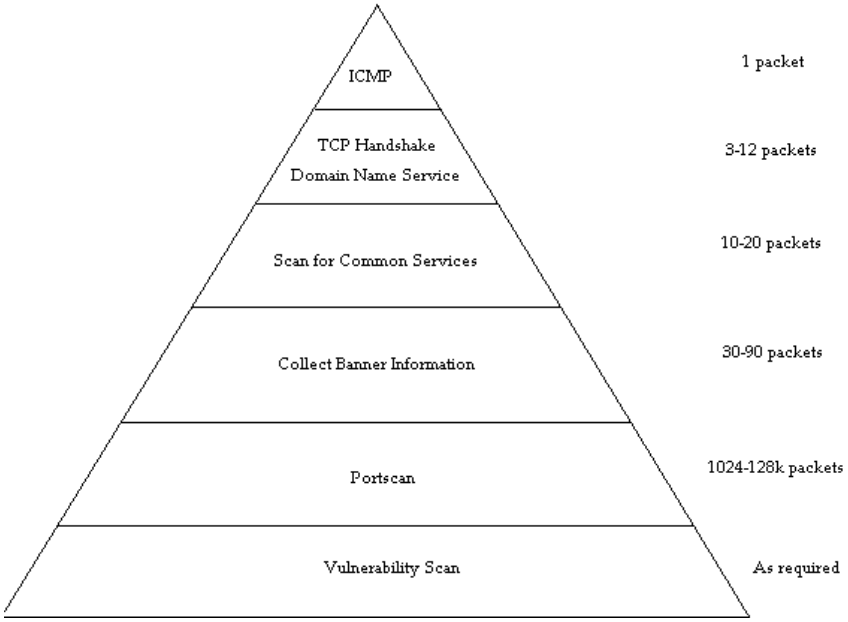
The environment inside the computer is fast, but all commands are not instant. Speed issues are important, and waiting time for any vulnerability, functional tool, or agent to complete an examination will undoubtedly expose an agent to risk.

Real time response from tools should be a requirement, but the structure of all the supporting security tools should be arranged so that the fastest way to identify the problem is attempted first, then followed by the next fastest way, and finishes with the slowest way.

To demonstrate this, a network vulnerability scanner is a great example of a tool that can be enhanced by reporting times. A network vulnerability scanner is a multi-purpose used to find, identify, and locate vulnerabilities on a remote computer.

Its processes include finding a host, using ICMP, fingerprinting the host by looking for TCP/IP stack characteristics unique to the system, asking the domain name service details about the host, scanning network ports looking for services, then communicating with services discovered to gather information about the host.

Typically, the report is generated at the end of the scan. However, to properly fit the ideal tool description for the best results for the agent, the steps required can be measured and sorted by speed. The following diagram represents this effect:



Even with very minimal packets, a lot of information can be obtained.

At the ICMP layer, an estimation of the distance of a host on a network can be determined by the TTL. If the TTL + time taken = 128, the host is a Windows computer, otherwise if the TTL + time taken is 255, it's something else. It also determines if the host is alive or offline.

The next layer requires a little fast interaction, which can include getting the domain name service name for the host and uses passive fingerprinting techniques to determine the host type. DNS something (rarely anymore) contains information about the host CPU and operating system. The passive fingerprint will uncover which operating system and/or hardware the computer is.

With these first two layers, an entire "Class B" (~65000) computer network can be mapped and identified in a matter of seconds. A "Class C" (254 computers) network can be identified at a pace that's next to instant. This is a lot of information, up front, about potential attack points. It also reveals the existence of unidentified computers on a network, so this process isn't strictly for offensive computing.

Current network scanners tend to go 'host by host', finishing the entire examination for one computer before continuing on with the next. The process is slow and tedious. However, restructure of this tool by fastest order of discovery provides very useful

information to the Agent and thereby adds more capabilities and reaction time to the security network.

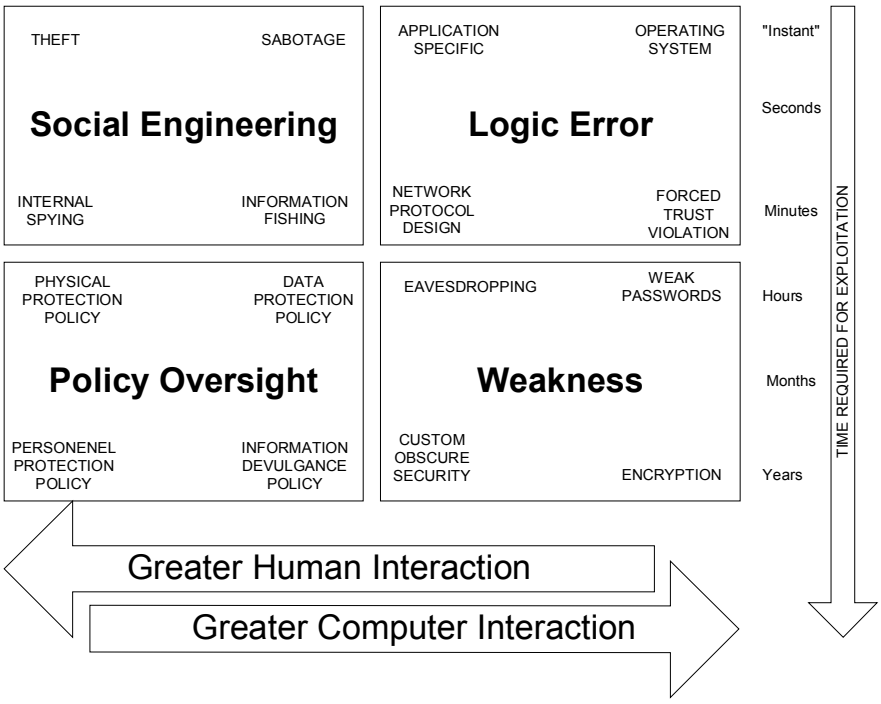
Overall, the majority of functional tools need to re-arrange their processing order in order to accommodate the ideal of the Informational Warfare Model.

Vulnerabilities Layer

A considerable amount of information about computer vulnerabilities is available in Eric Knight's previous online publication "Computer Vulnerabilities" which is (at the time of this writing) available on www.ussrback.com and is free for downloading.

To provide details about the nature of vulnerabilities as they relate to the Informational Warfare Model, they are considered a 'nature' and on their own, do nothing until an outside force acts upon them.

The 'vulnerability landscape' is a relationship of human and computer interaction against time required for the vulnerability to be utilized, and appears like this:



The first observation that can be made about the vulnerability landscape about it's connection with the informational warfare model is that the greater the human interaction required, the more complicated the facilitators will have to be in order to take advantage of a vulnerability. In fact, halfway across the "Social Engineering" and "Policy Oversight" landscape, even very generalized informational warfare fails completely.

This is ***not*** an oversight in the informational warfare model; it's the ***proof*** of the model. From the observations made at this level, there is clearly a traversal going on between the warfare landscapes between physical and computational.

Clearly from observations made in the Vulnerabilities Layer, computers can influence people, and people can influence computers. However, the ability to perform the tasks changes as the environment changes, thereby necessitating computer agents and ultimately a command structure.

The following theorems can be deduced from looking at this landscape in relation to the Informational Warfare Model:

- As programming becomes more advanced and simulated computer intelligence increases, computers will be able to expand their automated influence across the vulnerability landscape.
- As physical security becomes more computerized (biometrics, imaging, and robotics), informational warfare will extend into the physical warfare aspects just like physical warfare influences informational warfare now.
- There is no limit to the number of vulnerabilities that can exist across this landscape.

The vulnerability landscape clearly shows which aspects of security will first be targeted and used by a security network. The elements that are closest to the native environment will be utilized first, and will attempt to stay as close to its native environment as possible. Without an ability to evolve past it's programming, the entire security network will remain isolated to the computer side of the vulnerability landscape.

> **Editorial Comment**: At this level, with these theorems, and with the evidence of vulnerabilities and warfare presented already, this document is borderline science fiction, even if it does contain all the intermediary steps required to produce this 'final result'. "Computer Vulnerabilities", my past publication, does go through a lot of detail connecting and describing the influences of vulnerabilities and attempts to automate even at the far human influence end.
>
> I want the connections to be identified and peoples' mind stimulated. Robots, physical warfare, digital, artificial intelligence, and networks of battling computers fighting for dominance is not going to easily be accepted in the modern day as "reality".

However, I'm working on stemming active criticism about the viability of this structure in that my programming efforts have been driving the creation of this model. Much of what exists here already exists in a working defensive framework that has awareness of all layers, 1-6, thereby completing the informational warfare framework in the way it was intended before this publication was released. This project is called "Fatum" (named from the Greek mythos "Fate") and can be downloaded from the www.swordsoft.com web site.

Command Layer Construction

The Command Layer can be conceptualized as having four primary layers, although the layers often draw and communicate directly with one another in special events. Therefore, the conceptual sub-layers are generalized to their natural placement and may or may not have sub-layer overrides.

For example, in physical warfare, if an agent infiltrates an enemy and gets close to the opposing forces leader, 'higher command' will stop giving the agent general orders and then become quite specific and the agent will then report to a higher ranking officer for a duration of time.

Therefore, the construction of the Command Layer starts at its lowest sub-layer with the ability to track its agents. The next sub-layer deals with generalized processing of all agents in the security network structure that this commander has authority over, followed by the thinking capacity (artificial intelligence), finally reaching the top of its layer definition with an ability to receive command and control from a higher authority.

The four natural sub-layers are:

- Agent Status and Control Sub-Layer
- Command Control Sub-Layer
- Artificial Intelligence Sub-Layer
- Higher Authority Communication Sub-Layer

Additional sub-layers and inclusions in sub-layers can be numerous and each sub-layer can have a near infinite number of improvements and controls.

> **Author's note**: Command structures probably don't follow any particular ideal number of layers and components, ideally there shouldn't be a higher authority, and the computer should be able to make all judgments and handle all amounts of

workload. For the sake of "keeping it real", this chapter outlines a very basic example and evolutionary programming and new ideas will drive the construction of more complicated and effective models.

At the end of this chapter is a sample "Layer 1" model used by the Fatum Enterprise Server, the first working attempt in software to follow the Informational Warfare Model. The demonstration of the components used and their sub-layer associates are shown clearly.

Agent Status and Control

The Agent needs to be thought of a conceptual unit to the Command Layer, keeping a close eye on its status. Likewise, it needs to be aware of how the orders given to it are executed. For example, if Command instructs the Agent to perform a routine inspection of all of its slower security functions, and an attack is detected, there needs to be an effective way to tell the Agent to cease and desist the time consuming tasks and focus on the immediate danger.

Likewise, Agents need to identify and authenticate, providing the integrity features the agent needs to be checked, and so forth. These processes begin with the lowest sub-layer, closest to the communications layer, and call higher sub-layers to provide informational support.

Command Control

The process of determining when and where an order needs to be processed falls under the command control sub-layer. There are direct orders (from A.I. or higher authority), standing orders (scheduling functions), and conditional orders (priority events based on a condition that occurs.) Each of these situations requires a slightly different approach to how the orders are sent to the Agent Layer.

Likewise, when information is received, it needs to know if its real-time, collected over a period, its priority, severity, etc. The incoming information needs to be sent to the proper processing elements in the higher sub-layers.

Artificial Intelligence

This sub-layer consists of all the required processing logic in order to interpret the meaning of events and place the events in their proper environment. The included description includes an analysis components and a system for declaring 'priority events'. The best way to conceptualize the two is that the analysis system is the 'fuzzy logic' A.I. component, and the 'priority events' is the conditional command component.

Higher Authority

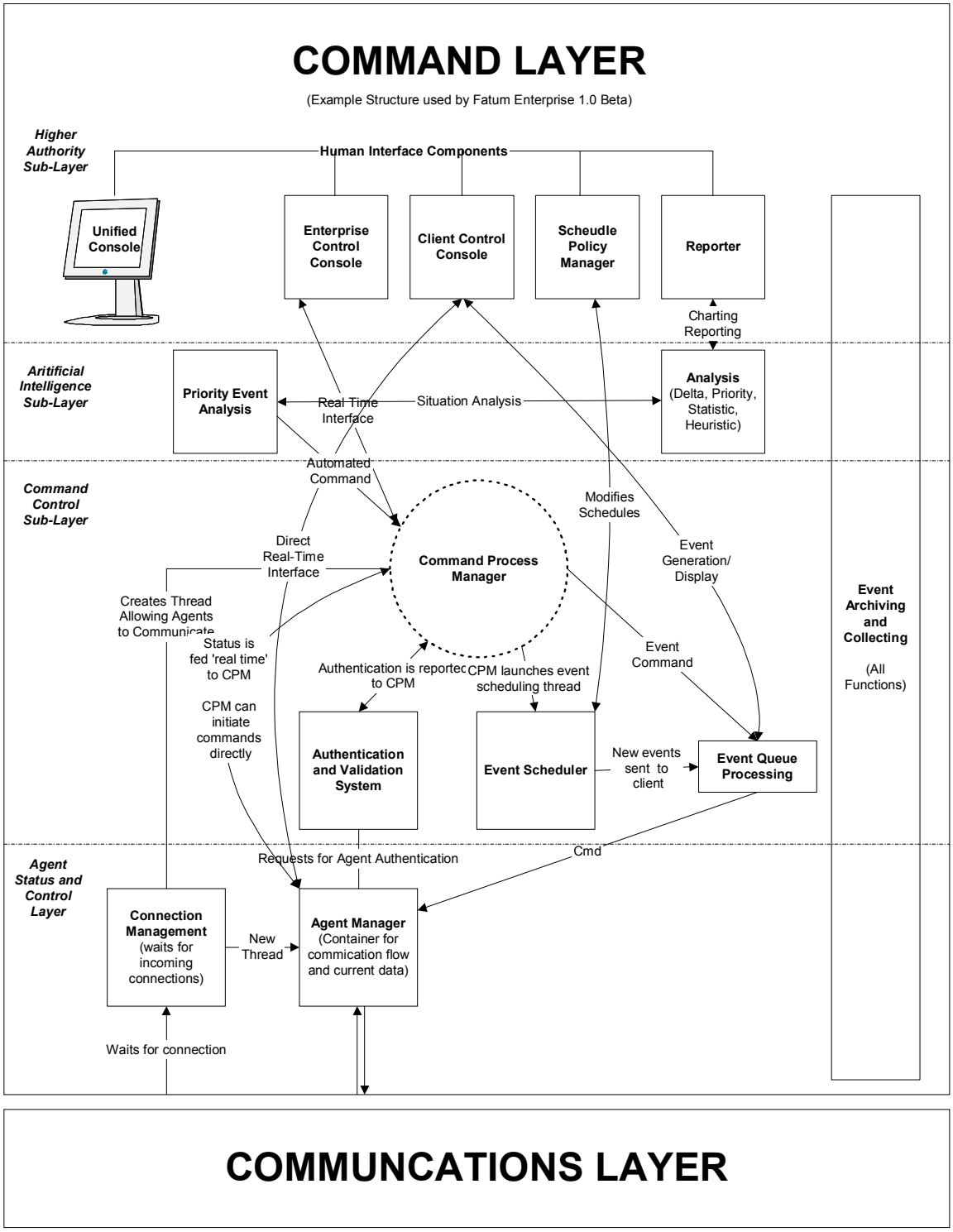
The current requirement for this technology (and probably will be a requirement for the next 30 years) will be to have a master console that gives a human the ability to 'influence' the events transpiring. The master console ideally should allow direct control over all the layers of the model, and this is done through robustness of control. Ideally, as mentioned before in the "ideals" of the Informational Warfare model, all control should be omni directional and information should be added, not removed, from the flow between Layer 6 up to Layer 1.

Likewise, although not demonstrated within the example, there needs to be a communication path to higher level controls. This is done by creating a Communications Layer above the Command Layer, and then creating another Command Layer. This series of additional layers is conceptually unnecessary, but required by present technology.

The structure of command-to-command layers is done for the following reasons:

- Reduce workload
- Create specialization
- Expanding beyond environment limitations
- Adding additional survivability to the security network

In this way, it's natural to think of the security network as being like military command, having lieutenants, captains, colonels, majors, generals, and a commander-in-chief. The goal, of course, is that each Command element has an adequate workload to perform its mission, and idealistically, that the overall security network structure has proper workload balance so that bottlenecks are not created on overloaded command systems when there are latent command posts able to perform the same functions.


COMMAND LAYER
(Example Structure used by Fatum Enterprise 1.0 Beta)
Higher Authority Sub-Layer
Human Interface Components
Unified Console
Enterprise Control Console
Client Control Console
Scheudle Policy Manager
Reporter
Charting Reporting
Aritificial Intelligence Sub-Layer
Priority Event Analysis
Real Time Interface
Situation Analysis
Analysis (Delta, Priority, Statistic, Heuristic)
Automated Command
Command Control Sub-Layer
Modifies Schedules
Direct Real-Time Interface
Command Process Manager
Event Generation/ Display
Creates Thread Allowing Agents to Communicate
Status is fed 'real time' to CPM
Authentication is reported to CPM
CPM launches event scheduling thread
Event Command
CPM can initiate commands directly
Authentication and Validation System
Event Scheduler
New events sent to client
Event Queue Processing
Event Archiving and Collecting
(All Functions)
Agent Status and Control Layer
Requests for Agent Authentication
Cmd
Connection Management (waits for incoming connections)
New Thread
Agent Manager (Container for commication flow and current data)
Waits for connection
COMMUNCATIONS LAYER

Agent Layer Construction

To build a perfect agent has about the same likelihood as building a perfect solider, however the basic capabilities need to be addressed foremost. Many of the skills required for agent combat are described in later chapters, but this chapter will focus on the construct and demonstration of the agent software.

The Agent Layer, like the Command Layer, has several natural sub-levels that are formed based on its construction. The layers can be crossed in case of special events, but the primary information flow will start with the layer closest to the Communications Layer, and then traverse down to the Functional Layer, then return through each Sub-Layer back to the Communications Layer.

A sample 'basic model' construct is included at the end of this chapter that comes from the design of the "Fatum Security Toolkit 1.1 Beta". This tool is an early version of a true Informational Warfare Model constructed defense agent.

The natural layers identified by the construct of this particular agent are:

- Security Network Sub-Layer
- Artificial Intelligence Sub-Layer
- Data Processing Sub-Layer
- Function Control Sub-Layer

The actual number of sub-layers, and components that reside in natural sub-layers are subject to technology and management advances. For example, the "Command Processor" module performs some intelligence functioning, but may be expanded to its own sub-layer as required by future technology.

Author's Note: The Agent Layer has the reference "XML", which stands for "Extensible Mark-Up Language", an information transport for record structures. Used in the context of this diagram, XML is not referring to the files but to

a record structure that is passed in memory that is XML-similar.

Security Network

The Security Network Sub-Layer is responsible for communicating with the security network Communications Layer. It needs to be able to receive command and report information back to the Communications Layer. Likewise, the information about the credentials of the Agent need to be presented, as well as any checks to demonstrate to the Agent the security of the Communications Layer and Command Layer.

Artificial Intelligence

The Artificial Intelligence Sub-Layer is responsible for the logical thinking processes, issuing commands, following orders properly, and generating analysis of the information it collects. The presented model adds human interaction components for overriding commands and performing analysis on the host.

The primary reason for a human interface is configuration of the tool, and after configuration is established the Agent should be able to run without the need to ever use the console. In the event of a Command Layer failure, the Agents can be restarted, reconfigured, controlled, and analyzed by the command interface.

Data Processing

The Data Processing Sub-Layer takes information presented from the sensors in the Function Control Layer and processes them into a form understandable by the Agent. Filters may be added between the parsing function and the data collection function in order to provide 'implied details' when the functional tool doesn't provide enough information about the security problem it detects.

For example, an anti-virus package may return an alert such as "Worm.reallybad located on File c:\program.exe" but omit the time stamp and its implied that this is an event of a "high" security nature. The filter will then add the information to the results of the functional tool's output and use that for later analysis.

Function Control

The Function Control Sub-Layer exists to handle the input/output of the functional tools the Agent is called upon to use. Because the desired result is that each functional tool An interface needs to exist to activate any functional tool. Likewise, if the tool fails to start or no longer exists, it needs to immediately notify the command processor to let it know something is seriously wrong.

The sensors for the functional tools perform the task of collecting the information. Four types of sensors have been identified for this publication.

- Log File
- Streaming
- Boolean
- Result

Log File Sensors

Log files sensors will attempt to collect information from a log file, this method is not in real-time. If a system is written that collects information from a log file as its being written, for example a named pipe in *NIX operating systems, the modification can be considered a streaming sensor. Authentications systems usually would require a log file sensor.

Streaming Sensors

A streaming sensor is an open communications path that reports to the sensor system the moment that a problem is detected. Streaming sensors are 'real time' and are considered ideal for Agents. Intrusion detection systems and firewalls are often used with streaming sensors.

Boolean Sensors

A Boolean sensor produces a "yes or no" value. The sensor will have to add more detail about what produced the result and what it means. "Presence of vulnerability" checks usually result in boolean values.

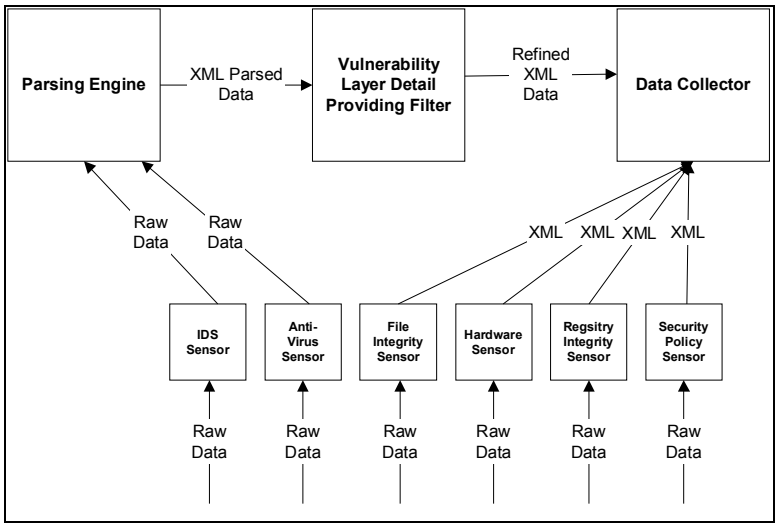
Result Sensors

The result of the function is a list, line, or record. The information is sent as a quick transfer through a stream and the stream is closed. Network security scanners and password crackers are examples of functional tools that would typically use result sensors.

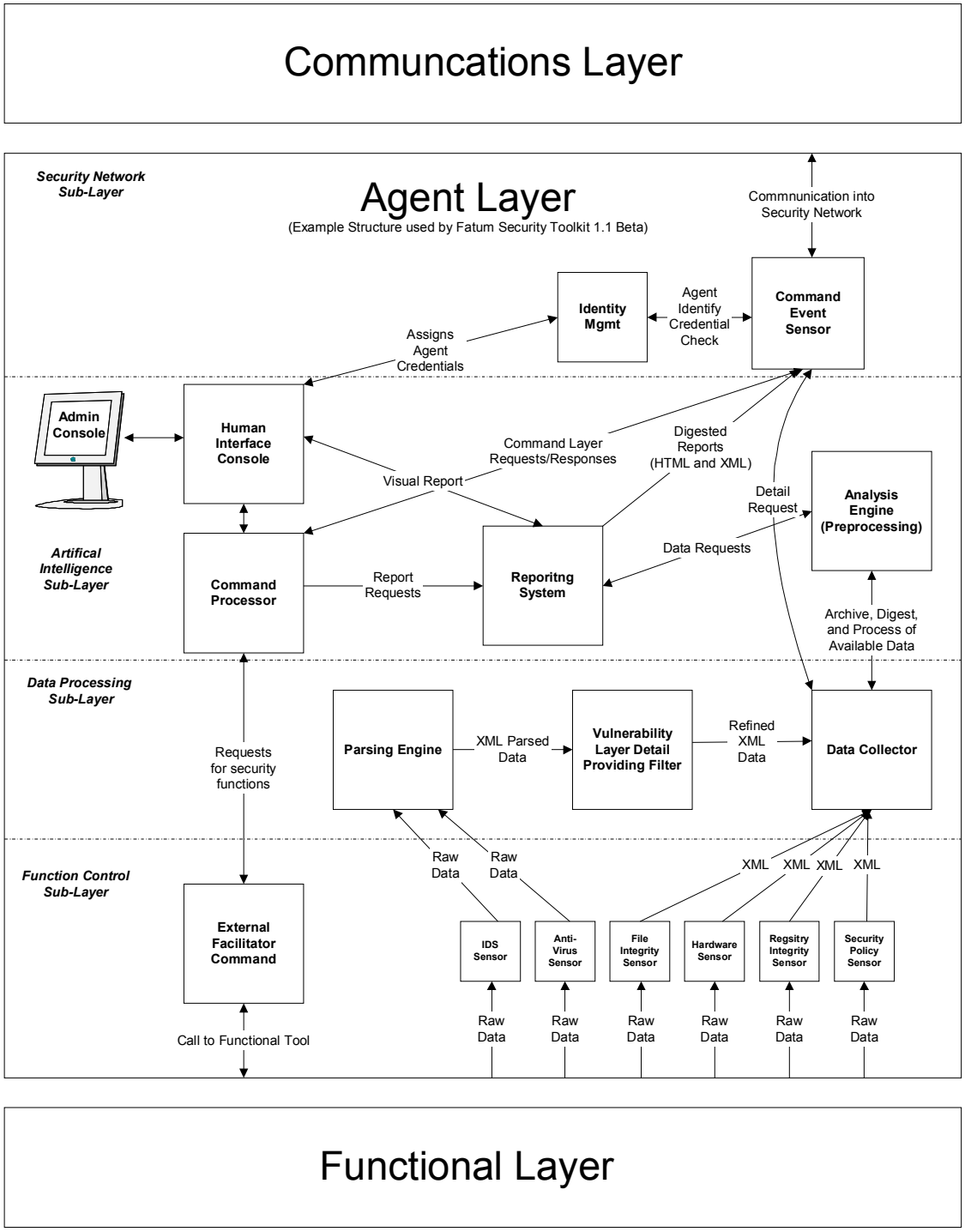
Functional Layer Standardization

From observations made in the displayed model, as can be observed from the following 'corner' of the graph at the end of the chapter, there are two paths for information to take after it is collected by a sensor.

The IDS and Anti-Virus tools were written by third-parties and don't naturally have integration with the Agent's communication structure. As a result, the information must pass through the parsing engine and then filtered to contain information required by the Agent's analysis system.



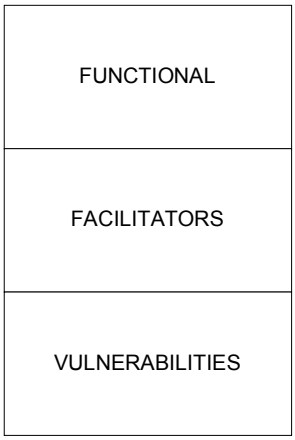
The other four sensors were written specifically for the Agent, and the information flows directly into the data collector. Standardization increases the performance of the agent considerably. The ideal for the Informational Warfare Model is to have all the functional tools produce compatible information for the Agent so that this sub-layer can have maximum performance.


Communcations Layer
Security Network Sub-Layer
Agent Layer
(Example Structure used by Fatum Security Toolkit 1.1 Beta)
Commnunication into Security Network
Identity Mgmt
Agent Identify Credential Check
Command Event Sensor
Assigns Agent Credentials
Admin Console
Human Interface Console
Command Layer Requests/Responses
Digested Reports (HTML and XML)
Visual Report
Detail Request
Analysis Engine (Preprocessing)
Artifical Intelligence Sub-Layer
Command Processor
Report Requests
Reporitng System
Data Requests
Archive, Digest, and Process of Available Data
Data Processing Sub-Layer
Parsing Engine
XML Parsed Data
Vulnerability Layer Detail Providing Filter
Refined XML Data
Data Collector
Requests for security functions
Raw Data
Raw Data
XML
XML
XML
XML
Function Control Sub-Layer
External Facilitator Command
IDS Sensor
Anti-Virus Sensor
File Integrity Sensor
Hardware Sensor
Regsitry Integrity Sensor
Security Policy Sensor
Raw Data
Raw Data
Raw Data
Raw Data
Raw Data
Raw Data
Call to Functional Tool
Functional Layer

Common Network Attack Strategies

This chapter outlines several network attack strategies as observed today, showing the relationships to the informational warfare layers. The effort in this chapter is to demonstrate the evolution of the effectiveness of informational warfare in relation to their model properties.

Hacker Attack

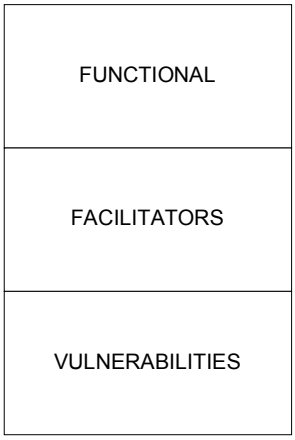


A hacker attack is a slow process where a human attempts to use level 4-6 tools in order to compromise the security of a computer. The advantage of such an attack is that creative analysis is at a maximum, but the speed of the attack is at a minimum.

Hacker attacks are the most often used in controlled penetration tests to prove the effectiveness of security. Due to the nature of security issues, the penetration test will likely be incomplete due to the time constraints and is absolutely going to be rendered obsolete quickly. However, gaping flaws that a computer was not programmed to detect will become apparent.

Attack Speed: Low
Stealth Factor: Low
Management Factor: Medium (human controlled)
Response Time: Low
Compromise Level: High

Viral Infestation

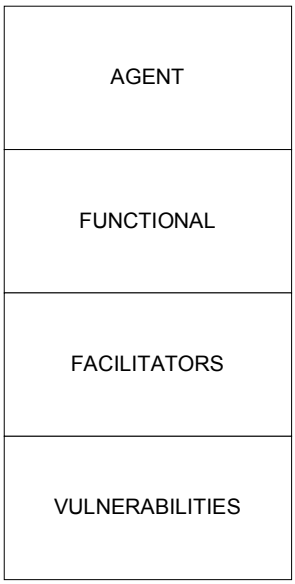


A viral infestation occurs when an attack omits layers 1-3 and simply propagates randomly. It substitutes pre-programming for human logic. It has no awareness of anything except its own components and the host environment its on. Its interaction with other tools is based on level 5-6 components that are included in its propagation (such as the vulnerability in most anti-virus packages of being easily deactivated or checking to see if the virus is already present.)

Due to the lack of calculation logic, viral infestations are the fastest of all informational warfare attacks.

Attack Speed: Fast
Stealth Factor: Moderate (small and hard to notice)
Management Factor: None
Response Time: None
Compromise Level: Low (only what its programmed to do)

Bee Swarm

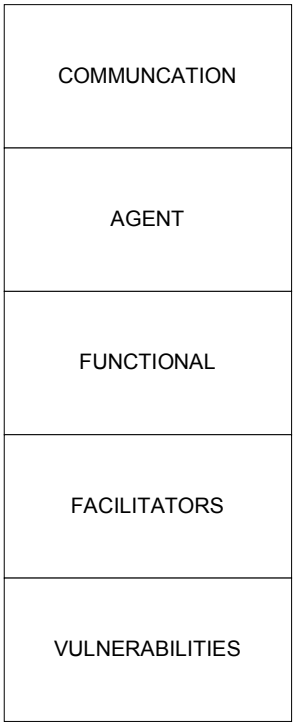


Similar in nature to a viral infestation, a bee swarm attack omits layer 1 and layer 2, and substitutes command and control with inherent programming in layer 3. Very purpose oriented, bee attacks are useful for reproducing, "finding honey" and "swarming a target". "Bee"-like attacks have been predicted to work well against military or business targets, and often search for words like "nuclear", "secret", or look for "16 digit" account numbers and put them in some sort of storage hive (usually a public message forum to make the origin hard to track.)

Because "bees" have pre-programmed rules, they aren't as fast as viral infestations. Although more intelligent than a virus, they have no ability to evolve past their programming.

Attack Speed: Fast
Stealth Factor: Low
Management Factor: None (pre-programmed)
Response Time: Fast
Compromise Level: Moderate (capable of pre-programmed attacks of a greater complexity against a target network)

Conscription



Many agent programs have been developed that are 'trojan horse' in nature. Once the Trojan horse is installed, it becomes a part of a communication network (Level 2) that allows it to work with other compromised (conscripted) computers. These computers are often called "Bots" (robots) and their purpose is to blindly follow commands.

By using a simple command console or by a human communicating at the Communications Layer by using commands for IRC or other messaging constructs, the hosts are usually completely compromised and can perform very complicated and elaborate attacks. The speed of which these attacks take place are at the pace of the human controller. The ability for the human involved to increase the control process is present, although the ability to perform fast calculations is up to the programming in the agent.

Attack Speed: Slow
Stealth Factor: Low
Management Factor: Moderate
Response Time: Moderate
Compromise Level: High (human controlled)

Invasion

A full scale level 1 through level 6 controlled attack, invasions have not yet been observed in public, grand scale form. Such an attack would place an agent on a host, attempt to compromise and control all of the security components on the host, violate the trust the computer shares with the other network elements, restructure the security of the enterprise infected to hide the presence of the attack, and allow for all forms of information search and control.

Agents can be dynamically fed new tools (as required) for a situation by the command structure (Level 1), can change the nature of their goals dynamically (Level 1), and can request assistance from the command structure (Level 1).

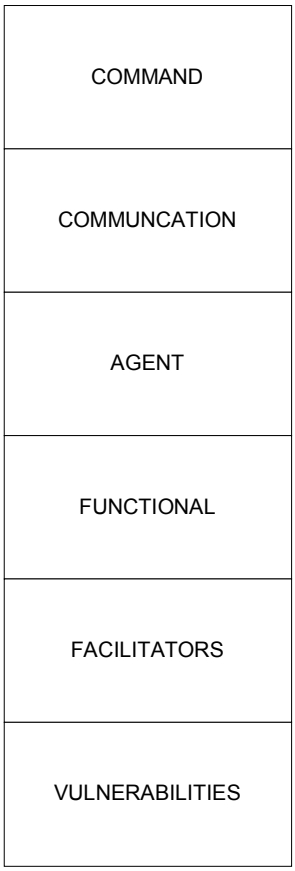
Invasion attacks are the slowest (at the agent vs agent level) of the fully automated approaches, but are also considerably faster than human controlled approaches and have the ability to handle multiple battles simultaneously.

Agents can be assigned tasks rapidly, given tools needed to perform their duties, and be ordered to remove tools once used so they can't be 'captured'.

Attack Speed: Moderate to Fast
Stealth Factor: High
Management Factor: High
Response Time: High
Compromise Level: High

Crawler

A crawler is a security network of linear design and limited size, designed especially for searches. The crawler will grow to the size of its environment and then choose to either divide or to relinquish an agent from the crawler structure.



This is a very, very intelligent form of a computer virus. This structure follows a 'keep it small and simple' (KISS) method but still has the advantage of a command and communications system.

Its mission can determine the size of the crawler. The purpose of each crawler segment is ideally a phase in the infiltration attempt.

- Command and Control segment, used to hold all the 'moving' information and repository. It moves slowly so it needs to already be fairly well established on the host before it arrives.
- Preparing Command and Control, a segment that will spend its time preparing to become the next Command and Control center after the crawler reaches its next growth cycle.
- Mission handler, accomplishes tasks associated with the mission.
- Infiltrator (stinger), this is a new agent that is in the process of compromising a host.

Once the infiltrator finishes its task, it waits for the Command and Control system to give the order to select a new target. Tools are then propagated up the crawler segments.

Crawlers forfeit the abilities of human command and have a pathetically low combined capabilities rating. However, they are very stealthy and very difficult to catch.

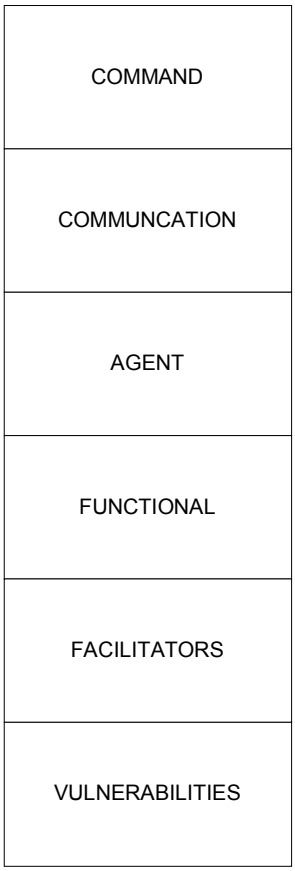
Attack Speed: Moderate
Stealth Factor: High
Management Factor: Low
Response Time: Moderate
Compromise Level: Moderate

Amoeba

An amoeba is a security network that has limited expansion capabilities. It probably has no abilities to expand its Command Layer structure. The result is the structure expands itself out to the limit of its potential and has to carefully pick and choose targets after that point.

Because of its growth limitations, will function much like its namesake as a single-celled organism, it has attacks and defenses, but division of the amoeba means two separate, limited sized structures.

Attack Speed: Fast
Stealth Factor: Moderate
Management Factor: Moderate to High
Response Time: High
Compromise Level: Moderate



Infiltration

The most likely type of attack, based on modern information warfare, will begin with basic 'espionage', 'camouflage', 'sabotage', and 'subterfuge.' This means that the attack will begin not with an outright assault, but a very sneaky single agent insertion that will gradually move about the security network.

The agent may, in fact, try to co-exist with other agents and try to be as invisible as possible. However, the absolute full power of the attacking security net is at its disposal whenever it needs additional capabilities.

Attack Speed: Slow to Moderate
Stealth Factor: High
Management Factor: High
Response Time: High
Compromise Level: High

Attack Method Comparison

Using "best case" scenarios, the following table shows the effectiveness of various strategies in relation to their use of layers. (Formula used: low = 1, medium = 2, high = 3, (all factors added) / 15), rounded to nearest integer)

Warfare Structure	Overall Effectiveness	Layers Involved
Hacker Attack		3
Viral Infestation		3
Bee Swarm		4
Conscription		5
Invasion		6
Crawler		6
Amoeba		6
Infiltration		6

The obvious conclusion from this table is that effectiveness of the attack increases with automation included in the higher layers of the informational warfare model.

In the present day, known attacks seem limited to layer 3-5, which seem to have roughly the same degree of overall effectiveness. Once they reach 6 layer capabilities, the office security gurus are likely to be overwhelmed by the magnitudes of speed and complexity of an opposing security network.

Agent vs Agent Warfare

When a host becomes compromised, the attacker will often attempt to remove any known tools that can stop it. Several virus programs that have been discovered will attempt to deactivate anti-virus packages that can detect it. However, in an environment with an agent program present, the deactivation of a tool would trigger an alarm and the security overlap "effect" would successfully identify the attack. The virus, which is not creative, would yield to the agent.

If the virus targeted the agent, it would find that a properly built agent would be considerably more difficult to remove because in the ideal warfare environment, it has all the characteristics of stealth and control allowed to it.

In the event that the attacker is another agent, the computer takes on a dual identity and becomes schizophrenic as the two agents struggle for domination of the computer system. In the simplistic form of informational warfare today, the actual battles are over quickly.

In an informational warfare environment with advanced agents, the attacker will be considerably more difficult to eliminate from a host but will also have a significantly more difficult time eliminating the opposing agent. Both agents will be programmed with an aversion to destroying the host computer, using annihilation as a last resort.

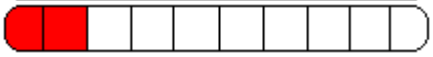
This chapter covers tactics that agents will use to battle agents and the opposing security network.

Agent Attacks

The agent goal is to have full control over its environment, in this way it can dispose of the attacker/defender in a number of different means.

Shutting down processes

If the opposition is identifiable by its running process on a host, it can be terminated allowing the victorious agent to attempt to clean up. This is a very effective attack, and can often end an agent vs. agent battle quickly.

Effectiveness:	10	
Speed:	10	
Complication:	2	

Promoting access level

The attacker will undoubtedly utilize a vulnerability to become the highest level of access possible. Compared to vulnerabilities that exist in a system from the outside the number of ways to promote access internally is many exponents greater. It can be assumed that any agent will have the resources to promote its access in at least 10-20 different ways.

Effectiveness:	8	
Speed:	8	
Complication:	5	

Seizure of Security Tools

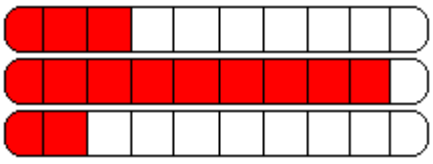
An anti-virus package can be just as useful to the attacker as the defender. If coded properly, its own signatures can be removed from the anti-virus database and the opposing agent's signatures can be added. Likewise, seizing of firewall functions can be used to disrupt the defender from requesting assistance. Almost all tools at Layer 4 of the model can be used both for and against agents on a host. The complication factor is high because the agents have to be programmed specifically to utilize the functions of the tools on the host. The attacker is at a serious disadvantage, but has an element of preparation.

Effectiveness:	8	
Speed:	3	
Complication:	10	

Creating New Services

The defending agent is typically outfitted with security tools based on the computer its on, but not have the security tools necessary to handle new services. If the attacker activates a service that has been configured by the attacking agent to include a path back into the computer, the new service becomes a liability. Although new services are quick to be identified, not all of them are immediate cause for alarm for an agent.

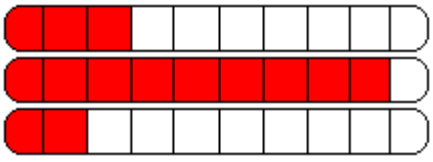
Effectiveness:	3
Speed:	9
Complication:	2



Downgrading

For systems that have dynamic patches, its possible for the improvements to be disabled which may allow for more vulnerabilities in the system that the defending agent will not become aware of or be able to respond to quickly.

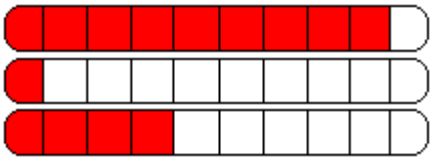
Effectiveness:	3
Speed:	9
Complication:	2



Removing the opposition

If the host is compromised and the process is not running, but the software is present and installed to run at a later time (presumably the next reboot), the agent can attempt to remove the offending program. This is often performed with anti-virus packages.

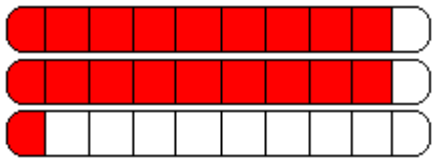
Effectiveness:	9
Speed:	1
Complication:	4



Disrupting communication

If the host has a firewall capability, and the opposing agent functions with a great degree of dependency on a security network than the agent performing this attack, a significant advantage can be gained in the course of the battle.

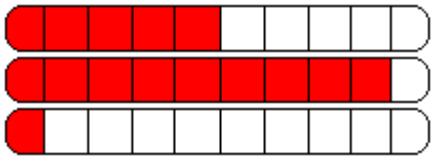
Effectiveness:	9
Speed:	9
Complication:	1



Backdoor

A backdoor (fast path around the security system already in place) is known as one of the earliest strategies. Creation of a backdoor (or multiple backdoors) can be a fast and simple process for the attacking computer. A well-equipped defending agent should have the ability to identify and disable backdoors, it might be able to remove the backdoor before a defeated agent re-enters the system.

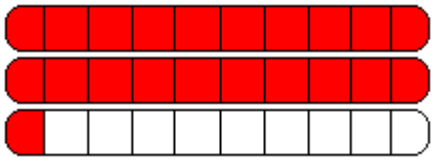
Effectiveness:	5
Speed:	9
Complication:	1



Highest Level Access

The agent starts with full control over the system, and attackers usually will have to start with a lower level of access and use vulnerabilities to promote itself before being able to contest the agent directly. Because of its effectiveness, speed, and complication, this method is the first one to be considered.

Effectiveness:	10
Speed:	10
Complication:	1



Binary Scan

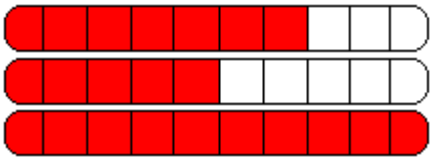
When the binary executable of an agent is discovered, it can be searched for function calls that can give away details about its characteristics. Although it won't reveal the extent of it's artificial intelligence or pre-programming, it can reveal if it can modify files, access the registry, access external programs (and presumably, which external programs), network awareness, and such. The information is used by the agent to assess the scope of attacker's threat.

Effectiveness:	5
Speed:	5
Complication:	5

Compromising the opposition

If the opposition is a trusted element in a network, then the agent to disrupt the attack can initiate a counter-attack. When this is possible, a significant collapse of the security on one side or the other could initiate.

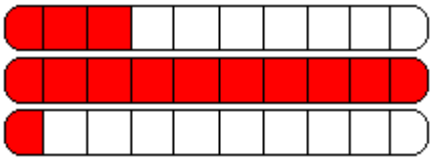
Effectiveness:	7
Speed:	5
Complication:	10



Call for help

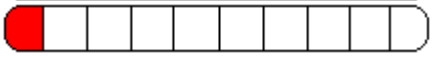
The agent can request help from a higher layer or human element to assist it in resolving the attack. In this way, new or alternative tools can be transferred to the host in assisting the removal of the attacker. Other agents in positions to disrupt the attacker may be given orders to do so.

Effectiveness:	3
Speed:	10
Complication:	1



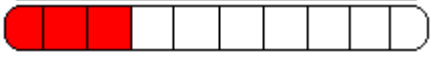
Ghosts

By creating a false image of a security element on the computer, the attacker may be fooled by its existence and begin attacking it instead.

Effectiveness:	10	
Speed:	9	
Complication:	1	

Analysis Disruption

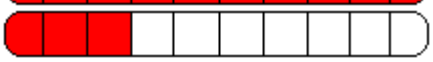
When the opposing agent is formulating a strategy, it occupies memory and depending on the A.I mechanism, will be planning a course of attack. Any interruptions to the calculation process will disrupt its 'thinking process'. In effect, it's like kicking a chess player from across the table. Doing this repetitively could cause an agent's ability quality decisions to be rendered completely ineffective.

Effectiveness:	?	VARIES
Speed:	10	
Complication:	3	

Sandbox Modification

Many modern security mechanisms are creating what is referred to as a "sandbox", it determines the limits on calling system commands on the computer. Commands that are not authorized for a particular program are immediately discarded and a security violation is created. Types of sandboxes are "stack overflow sandboxes", "program language environment sandboxes", "system calls allowed by a program sandboxes", and more sandboxes will be created in the future.

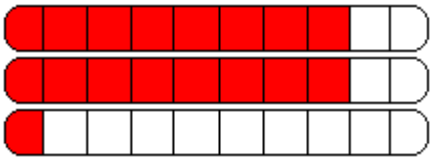
Modifying the security of a sandbox isn't easy to detect, and once a sandbox is compromised, a hoard of vulnerabilities can appear. The point of sandboxes is to allow for poorly written code, and trust in the sandbox to protect the system. However, the sandbox is a valid attack target for weakening the security of the host.

Effectiveness:	?	VARIES
Speed:	10	
Complication:	3	

Resource Starvation

An attempt at reducing the resources for an agent or security tool to the point they are unusable. This can be done by lowering the priority of a process down to the lowest possible levels, filling up drive space, or otherwise exhausting a critical resource on the system required for it to function.

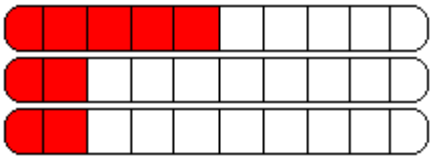
Effectiveness:	8
Speed:	8
Complication:	1



Overload

A security system can only be designed to handle so much information before it finally tanks. Attackers have significant advantage in that they usually don't care about keeping all the components in order until after the host is compromised. By giving all the components something to keep them occupied, the entire system may fail.

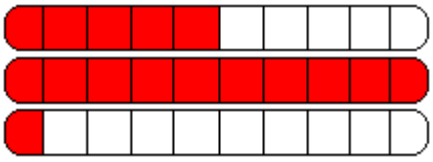
Effectiveness:	5
Speed:	2
Complication:	2



Rebooting

Immediately after an agent attempts to remove and seize control over the system, a quick reboot of the computer will force the system to reactive all the software, presumably under control of the agent that made the reboot command. There needs to be some element of certainty that the opposing agent will be cleared, but agents won't necessarily have the ability to made a reasonable judgment.

Effectiveness:	5
Speed:	10
Complication:	1

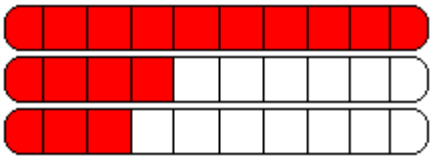


Agent Defenses

If attackers have ambush advantages, the agent has the home field advantage. A properly written agent is already configured to exist in the environment and can entrench itself deeply.

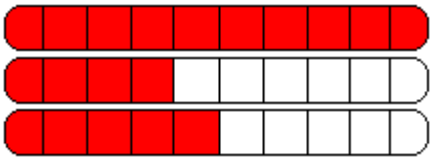
Deep Embedding

The agent can be launched into the system from any number of locations, and if the process is killed, it can be reactivated from 'secret' or 'hidden' locations on the computer. This is normally a viral or Trojan technique, but also works well for defending agents.

Effectiveness:	10	
Speed:	4	
Complication:	3	

Polymorphism

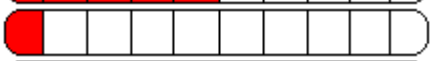
Another trick used by viral software, the program can recompile itself in a way to disguise its identity by using different version of the same code. For example, NOP is the operation "No Operation", it does nothing. By adding this into the code at random locations, it changes the length and sum totals of the software. Likewise, changing math equivalents such as "ADD 1" to "SUBTRACT –1" will remove code signatures as well.

Effectiveness:	10	
Speed:	4	
Complication:	5	

Advance Awareness

The agent can quickly reactivate compromised and deactivated tools. Configurations can be digested and stored on the computer at a location known only the agent. Typically, the defending computer has the advantage in terms of speed. Likewise, the attacking agent will gain advantage being prepared for tools on a host.

The information for 'advance awareness' can come from many sources for an attacker. The human influence has prepare the agent tool list in advance, or the tool list can be predicted by examining what currently exists on a different, compromised host on the network. The more information is collected during an invasion, the faster the speed of the invasion becomes.

Effectiveness:	5	
Speed:	1	
Complication:	5	

Agent Required for Use

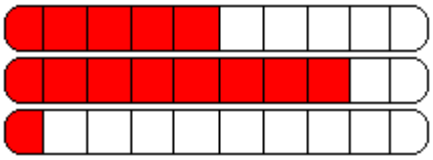
Tools, operating system functions, and such may be configured in a way that requires a specific agent to be 'in operation' in order for them to function. This will prevent the deactivation of the agent. This can be thought of like the new firearms that take biometric fingerprints of the person using it. If the fingerprints aren't authorized, the firearm is useless. The opposing agent will have to request and install its own version of the tool, if possible, in order to receive that function.

Effectiveness:	5	
Speed:	10	
Complication:	1	

Encrypted Binary Executable

The machine code form of the executable can be encrypted before placement and decrypted with a simple decryption routine. The decryption routine is usually extremely small (as little as 24 bytes of code) and can incredible polymorphism permutations as a result of its size and nature. Combined with polymorphism, any agent can be almost unrecognizable 'on disk'. In memory, the executable becomes decrypted and becomes susceptible to signature identification.

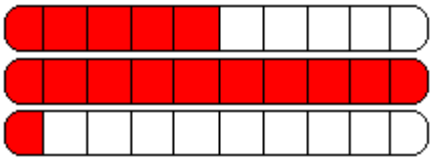
Effectiveness:	5
Speed:	8
Complication:	1



Quarantine

The agent can make the decision to tell the organization to no longer accept communication from the host and/or allow traffic to reach the host. This will undoubtedly disrupt the attacker and limit its abilities.

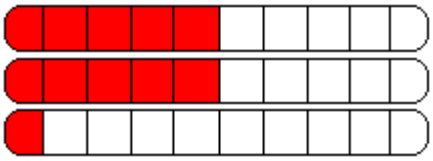
Effectiveness:	5
Speed:	10
Complication:	1



Scuttle

If the agent is aware of the information its protecting, the information can be transferred and removed, preventing the compromise of valuable information.

Effectiveness:	5
Speed:	5
Complication:	1



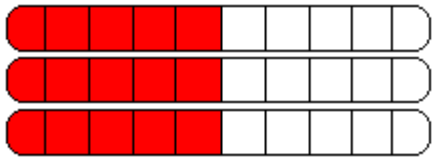
Hide valuables

If the host contains valuable information, or –may- have valuable information, the defending agent can attempt to hide the information by encryption. The process is often way too slow, but the effort at hiding might not be detectable or able to be stopped by the attacking agent even if the defending agent loses the battle. The largest problem is that computers don't have effective ways of determining what information on the host is valuable.

Effectiveness:	5	
Speed:	5	
Complication:	5	

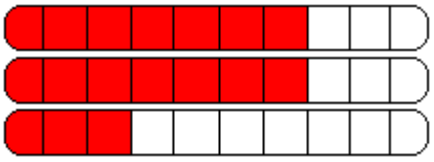
Honeypot

A deception created by the agent to fool the attacker. Creating a specific environment that the attacking program finds 'tempting' does this. The attacker focuses its attention away from the agent and the value properties of the computer. For honeypots that exist on a live computer, there are ways for the attacker to determine if it's trapped in a honeypot.

Effectiveness:	5	
Speed:	5	
Complication:	5	

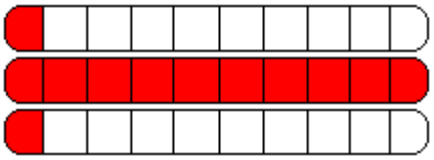
Replication

If one target is too simple, many targets can pose a greater problem. If two agents are on a host, each getting equal time, then 10 agents against 1 agent, each getting equal time, will tilt the contest in favor of the replicating agent.

Effectiveness:	7	
Speed:	7	
Complication:	3	

Mutually assured destruction

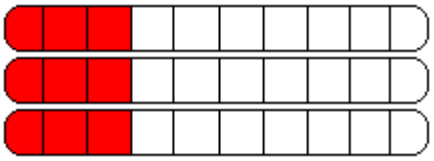
There comes a point where the battle becomes pointless and the risks are too high for the attacker to win. The agent will sacrifice the computer in order to 'capture' the attacker and prevent the information on the host from being stolen. This is as simple as initiating an immediate shutdown of the power on the host. This also decreases the overall strength of the security network, lowers the computer power of the enterprise, could disable necessary services, and overall, is a last resort.

Effectiveness:	1	
Speed:	10	
Complication:	1	

Forfeiture of Duties

If the host being compromised is unsure for success, but has critical responsibilities pertaining to the security network itself, it may forfeit its role and transfer responsibilities to another host. This is often useful in the case where it's the Level 1 and Level 2 components being assaulted directly. This technique keeps the integrity of the system in order to keep the security structure alive.

The effort of forfeiting duties can be hazardous for the server. The agent infecting the command and control layer will do everything in its power to prevent the order from being given, and instead attempt to seize control of the security network.

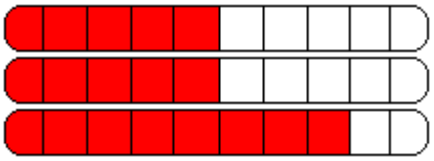
Effectiveness:	3	
Speed:	3	
Complication:	3	

Aftermath

Once the battle is over, and the computer is still running, the victorious agent has objectives to obtain in order to advance its mission.

Scavenging

The victorious agent, if programmed to do so, can search the 'dead' agent's configuration and attempt to collect more information about the losing agent. In theory, it can collect the agent's identity and attach itself to the opposing security network. This isn't necessarily a full-scale compromise of the other security network, but it could cause disruption depending on the sophistication of the victorious agent's security network capabilities.

Effectiveness:	5	
Speed:	5	
Complication:	8	

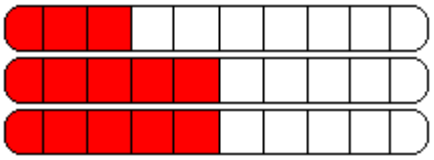
Searching for valuables

When the attacking computer succeeds in the attack, it can begin looking for valuable information. This can be identities, passwords, databases, documents, configuration information, network information, etc. In the case of the defender winning, collecting the exploits, tools, identity, and information relating to the compromise is a very high priority.

Effectiveness:	5	
Speed:	5	
Complication:	8	

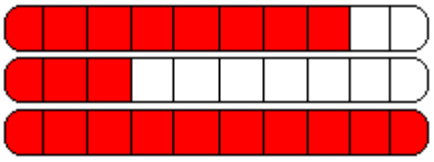
Cleaning the Logs

When the attacking computer succeeds in the attack, the log files can be wiped clean of the host to prevent analysis of why the attack was successful. When the defending computer wins, the logs need to be archived and removed from the host quickly so that analysis can be performed on the compromise. By winning the battle, the defender may have a follow-up attack aimed at destroying the computer so the logs are not reported accurately to the defense network.

Effectiveness:	3	
Speed:	5	
Complication:	5	

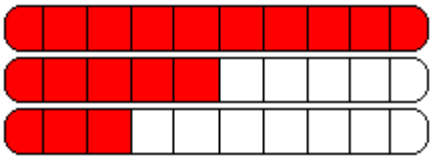
Customizing the environment

When the attacker wins, the host will become a formidable attack point and tools can be placed on the host to launch another attack. When the defender wins, if it can trace the attack to an origin, tools can be used to block another agent from entering the host. If the defender fails to secure the environment, another agent can enter and the battle will ensue again.

Effectiveness:	8	
Speed:	3	
Complication:	10	

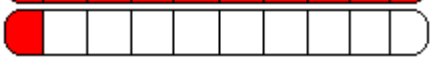
Selecting a new target

Once information is collected and analyzed about what assets have been acquired as a result of the battle, a new attack or counter-attack may be initiated by the agent, depending on its programming and as determined by its layer 1 commander.

Effectiveness:	10	
Speed:	5	
Complication:	3	

Reporting

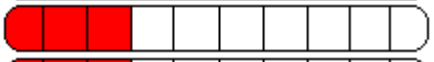
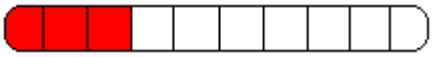
Once the battle is over, the result and aftermath needs to be reported to the security network. This report will contribute to the overall ability to create an optimized attack/defense against the opposing agent and provide information critical to the breach attempt.

Effectiveness:	10	
Speed:	10	
Complication:	1	

Promotion/demotion

This can happen anywhere in the network as deemed necessary. If the security network discovered that command and control is in jeopardy, agents can be transferred the layer 1 and layer 2 security components and establish a new leader. Likewise, if the layer 1 and layer 2 components were compromised, the selection of a new 'leader' is a necessity.

If the attacker is victorious, and the Layer 1/Layer 2 system is becoming overburdened due to tremendous campaign success, an agent can also be selected to start processing traffic to speed up the operations by receiving the Layer 1-2 components and being assigned agents.

Effectiveness:	10	
Speed:	3	
Complication:	3	

Fulfilling the Mission

Once a host is compromised, the attacking agent is victorious over the defending agent, it needs to advance or complete its mission. For more information on possible missions that an agent is designed for, see the chapter on Mission Goals.

Effectiveness:	?	VARIES
Speed:	?	VARIES
Complication:	?	VARIES

Event of Capture

When a battle ensues on a host, there are many ways to 'capture' an agent. Although its possible for another agent to capture, a very common situation will be that the user of the computer gets frustrated with its lousy performance and turns it off. In this case, all the tools that the attacking agent is using will be 'captured'.

When an agent is captured, its capabilities can be analyzed and any 'secret information' such as which vulnerabilities it was using to gain effectiveness against the defense network can be discovered and made public.

There are several techniques that agents can use to reduce the damage caused by a capture.

- Keeping sensitive tools in random access memory
- Deleting tools immediately after use
- Using emulation engines or polymorphic machine code
- Encrypting tools and data using keys stored in random access memory

Tools in Random Access Memory

Unless the tools are placed in non-volatile memory, the disruption of power or killing of the process will cause the tools being used to be irrecoverably lost.

Deletion After Execution

Secure file wiping tools can be used to erase information from the hard drive. If the tool is a single use only component, it should be eliminated securely after the tool's use is finished.

Emulation Engines and Polymorphic Machine Code

There are two forms of altering machine code so they are difficult to understand their purpose, the most common is Polymorphic code, first used by computer viruses and accredited to "Dark Avenger". Emulation Engines, created by the author of this book in 1993, are single use versions of the same technology.

Polymorphic Machine Code

Polymorphic code is created during the run-time execution of the software. The program attempts to alter the appearance but not the nature of its code. It will change simple instructions to equivalent functions, and insert commands that have no impact or value on

the final outcome of the code. Polymorphic programs have the difficulty of having limited flexibility on how much of the code it can alter because it will attempt to modify the program at the machine level, after the assembler constructs it. The code cannot expand past its size structure in order for branches and jumps to fit its pointer constraints.

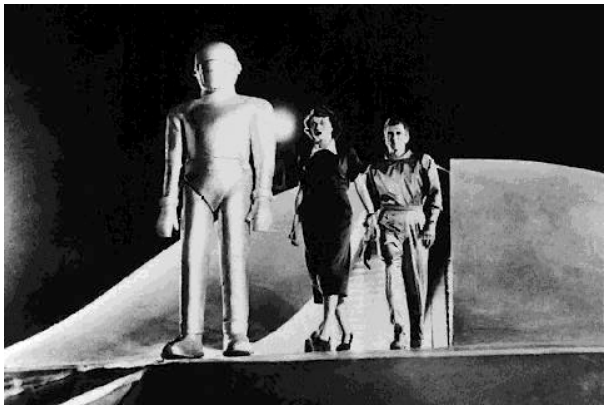
Emulation Engines

Emulation engines reside between the compiler and the assembler and perform the same task as the polymorhpic system. It has the advantage of being able to alter the code before assembly and therefore has no pointer constraints. Emulation engines require the presence of source code and an assembler, so they are likely to exist at Layer 1 and distributed "as required" by the Command Layer.

Encryption

If a file will be used often by an agent and needs to reside on the drive, the file can be encrypted with a key that is stored in memory. If the agent is compromised, the key is lost and the contents of the file are secured.

Human vs Agent



The argument that drives the creation of agent technology is that agents can defeat a human intruder better than they can defeat other agents. The proof of this exists in the damage being caused by viral infestations, deep embedding, and the use of 'root kits' that install numerous backdoors and Trojans on a computer – all functions of agent management that were originally meant to overwhelm the human entity by using the computer to perform the attacks on their own.

Because of the speed of the agent and the limited interface for a human attacker to make decisions, it's best for the human attacker to pre-program an agent to handle the attack. However, this particular essay is assuming a human has access to the host keyboard or through the network.

First off, lets list the defenses of Agents that a human will have absolutely no ability to influence on their own if the technique is present in the agent:

- Shutting down the opposition
- Disrupting communications
- Rebooting
- Deep Embedding
- Polymorphism
- Agent Required for Use
- Highest Level Access
- Encrypted Binary Executable

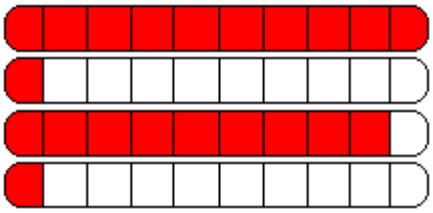
In effect, a human will have to assume the agent is too difficult to remove from the host. The most effective strategies of attack are then to separate the agent from its support structure, and to either bypass the agent or trick the agent into believing the attacker is an authorized user.

The value "Stealth Factor" is being added to demonstrate how likely other humans are for figuring out what exactly went on and how easy it is for them to track the hacker.

Physical Access

Because its doubtful that computer agents will be able to manipulate their physical environment in the conceivable future, an attacker can turn off the computer and move the equipment into an environment that prevents the agent from attacking.

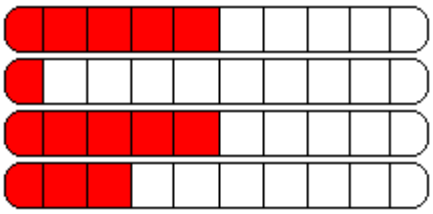
Effectiveness:	10
Speed:	1
Complication:	9
Stealth Factor:	1



Stolen Password/Identity

The attacker may be able to steal or use an identity from an individual that is authorized to work inside the security network. If this happens, for a period of time the user will be able to perform normal activities. Once the human attempts to change the security posture, then the agents will become aware of the trouble and "attack."

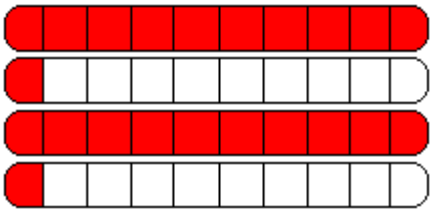
Effectiveness:	5
Speed:	1
Complication:	5
Stealth Factor	3



Insider Cooperation

What if the person not to be trusted has influence over the security network? The attack will undoubtedly be successful, but the culprit will be obvious. Although the effort at eliminating the security net is not difficult, convincing a person to risk that much is what raises the complexity rating.

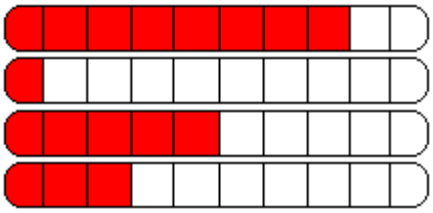
Effectiveness:	10
Speed:	1
Complication:	10
Stealth Factor:	1



Internal Access Point

This technique is definitely on the rise. Because of wireless communications and a lax attitude for the placement of "hot ports" to the network, outside attackers can enter into a network and place a computer inside the firewall.

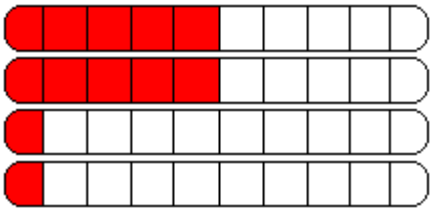
Effectiveness:	8
Speed:	1
Complication:	5
Stealth Factor:	3



Wiring Control

If the intent is to prevent the computer/agent being attacked from calling for help, requesting tools, and reporting incidents to the network while the hacker attacks the computer, this technique can be effective. The security network expects that computers will be turned off or can't communicate for limited periods of time depending on its function.

Effectiveness:	5
Speed:	5
Complication:	1
Stealth Factor:	1

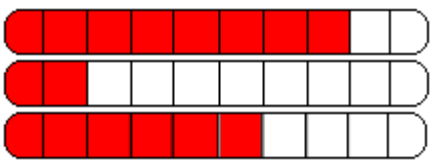


Human Effectiveness

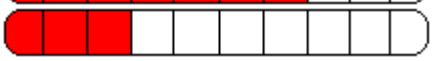
Without the assistance of an agent, human based attacks inside the security network and against other agents has a high degree of effectiveness but is slow and obvious. The hacker risks their physical well being to perform the most effective of attacks by placing themselves in close proximity to the computer system.

Getting an 'average' of the effectiveness of these attacks, it can seen that human influence is highly effective but has a high degree of complication and very low speed.

Effectiveness:	8
Speed:	2
Complication:	6



To compare this with the totals given from the ‘attacks’ of Agent vs Agent, the advantages for agents become clear:

Effectiveness:	7	
Speed:	7	
Complication:	3	

First of all, the effectiveness of the attacks caused from standard attacks isn’t much less than the overall effectiveness of a human acting independently. The speed is considerably faster, and the overall complication factor for the computer to solve the problems is much less.

In the environment of ‘cyberspace’, the agent has the clear advantage.

Mission Goals

In terms of warfare, the missions that can exist are infinite but they fall under general categories. For example, a soldier may have to fight in different environments but their goal -- lower resistance -- is the same. Once a soldier completes one mission, they are assigned another, and continue until there is no additional need.

Defense agents are given a very basic mission, and don't need to be overburdened with complicated objectives. They are in an environment where they need to fight fairly, but they also know what exists on a computer in advance and presumably have been given detailed instructions what to do.

There are four basic types of attack mission objectives – espionage, sabotage, camouflage, and subterfuge. A mission is one or a collection of specific objectives.

For example:

The mission is “to search and destroy company XYZ’s computer network.”

Objectives:

Espionage Related:

Steal all source code found on the computers and report to Command
Steal all financial records found on computers and report to Command
Discover and report all names and accounts

Sabotage Related:

Change all financial records with occasional, random numbers.
Change phone numbers
Delete lines from source code

Camouflage

Remove evidence of intrusion
Create false records pointing to internal employee mischief

Subterfuge

Launch ‘attacks’ from the hosts to outside companies
Change configurations of the network
Send ‘disinformation’ e-mails to all people in e-mail contact lists

Such a mission would be absolutely dreadful if it was successfully completed, but overall it’s a very simple mission. As more ‘mission coding’ is written, the most devastating a successful attack would be.

Espionage

The effort of committing an act of espionage is an attempt of obtaining information or other related 'power'. Usually this is in the form of information or access, and is considered 'significant' when a goal is reached.

- Credit cards
- Bank accounts
- Employee records
- Financial information
- Intellectual property
- Personal information
- Eavesdropping
- Acquiring "secrets"

Technically, "secrets" are the most dangerous catch-item on the list. If a married boss is having extra-marital relations and proof is discovered, the boss becomes easily and powerfully influenced by a blackmailer. There are a lot of other types of secrets that can be uncovered, but analysis of that information can be difficult for computers.

Sabotage

Denial of service is a common strategy, but more deceitful methods can prove more effective. Deliberately adding three spelling mistakes to a company advertising document can be thousands of times more devastating to the company reputation than crashing a computer for a few hours.

Effective sabotage involves human creativity, but some processes can be very easily automated.

From the history of destructive computer viruses, the ones that do small amounts of damage are more effective than the ones that cause a lot of damage quickly. They are harder to detect and harder for the extent of the damage to be determined.

Large-scale attacks are also possible, keeping areas of the Internet inaccessible for long periods of time. However, these attacks can often be shut down quickly – but not in all cases.

Common sabotage attacks currently are:

- Disabling the computer
- Disabling specific software
- Disrupting the network
- Removal of critical security mechanisms
- Removal of important business components
- Random file destruction

Sabotage is often performed as a malicious act, although it may be required to complete mission objectives, especially in the form of disabling security components. The most likely use for sabotage after an attack completes is to damage the mechanisms as much as possible so they cannot provide complete or accurate information about the intrusion, and thereby becomes highly questionable as to the extent of the attack.

Camouflage

The effort at disguising the nature of the attack, attacker, and that the attack is happening falls under camouflage. In some cases, it's completely unnecessary. However, those who value their freedom will do their best to hide their activities.

Agents can be programmed with all kinds of techniques used to hide their origins and mission objectives. A few of them are:

- Falsified attacks
- Misdirection
- Log elimination
- Log substitution
- Spoofing
- Masquerading
- Elimination of reporting entities
- Log overload
- Detection and Logging avoidance

Subterfuge

Stemming from the meaning of "to escape beneath", subterfuge is an attempt at performing actions secretly and escaping, in an effort to create favorable influence for the mission.

What is generally considered "underhanded" action, subterfuge often connects closely with "social engineering" and often creates serious trust violations between users.

Some examples of subterfuge:

- Sending forged e-mails
- Assuming identities
- Altering documents
- Placing information "in the wrong hands"
- Altering configurations (presumably for later access)

Subterfuge attacks are hard to generalize, and computers make poor conversationalists. Attacks of this nature are either extremely simple or pre-programmed specifically for a target.

Programming Evolutions Required for Missions

Later evolutions of espionage 'engines' can attempt to process conversations "Eliza" style, gathering factual statements based on English vocabulary. It will be able to pick out certain verbs and nouns that it can create connections with. For example, if the engine is 'reading' someone's e-mail box, it would try to recreate the conversation.

Email one: I would like to buy 10 copies of your product for $1000.

The computer would simply the statement through parsing and relating to the components in a simplified form:

I (want) (buy) (number) of (you-possessive form) product (for) (1000-monitary unit-USD)

Then amend some quick understandings:

(buy), association (I) is business client wishing to buy [to the computer: I (want) to buy]
(you), association (you-possessive) is selling something.
(product), association, product selling.
(1000-monitary unit-USD), is object being offered.
I is associated with the email address of the letter (joe@client)
(you-possessive) is associated with email address of the recipient (nancy@business)

The process of trying to parse a conversation and recording and tracking nouns, verbs, adjectives, etc. does cause a lot of information to be lost, but in the end, the conversation can be digested down to core elements:

Output:

> Email from: (joe@client), name "Joe Anyone"
> Email to: (nancy@business), name "Nancy Someone"
> Noun Reference to "product", 1. "air conditioner"

Verb References to “product”, 4. “Buy, use, test, understand”.
Noun References to “I”, 2 “business, accountant”
Verb References to “I”, 3 “looking, requesting, want”

The effort of this digested examination, or other forms output, can be run through additional filters in order to determine if this e-mail has information within it that relates to the mission objective.

(Editorial lighthearted comment- I hope they do this for spam-mail filters soon.)

At this level, it’s easy to produce a filter that can pick out key words that could trigger this. Likewise, subterfuge engines would collect the information from the filter and try and craft a response if required by its mission.

The difficulty in generating a response that’s meaningful isn’t beyond existing technology, and neither are efforts to adjust grammar usage to mimic a person’s style. For example:

Letter: 1000 words
Use of pronoun: “I” -- 50 times
Use of possessive: “my” – 20 times
Use of conjunction: “I’m” – 10 times

Then a 100-word response can include similar ratios of “I”, “my”, and “I’m”.

The effort at making computers ‘read’ is difficult, but programming evolutions can help identify information that is more critical for complicated mission completion. Computers can ‘read’ far faster than humans can, and providing digests is a significant advancement in informational warfare capabilities.

Agent Communication Structures

Ideally, the agents always have ready access to information in the security network. However, "in the heat of the battle" the communication links can be severed. The most obvious 'target' of a battle is the Command Layer.

Agents need a way to communicate and function with the absence of a Command Layer, even if it's only temporary, and also have a way of getting relevant security information in order to support their own decisions.

The requirements for a solid agent communication structure (minimal) should include the following:

- Ability to request needed tools from other agents
- Ability to request support directly from agents (Quarantine me now! Block this traffic now! Take these logs now! Here's the binary analysis of the enemy! Etc.)
- Ability to decide leadership in case the Command Layer becomes inoperative
- Share time consuming tasks

The evolution of agent communication structures has taken very different directions for evolution for offensive security networks and defensive security networks, although the merger of the two is imminent. Defensive security networks function primarily on a 'client/server' relationship, where the server commands each agent directly. Offensive security networks usually connect to a chat room ('bot-net') and coordinate with the commander from there.

The known structures for communication between agents are:

- Designated Server
- Broadcast Channel
- Peer-to-Peer Network
- Relay
- Private (direct communication)

Each of these communications methods are discussed in greater detail in this chapter. In the best environment, more than one communication structure exists. Client/Server and Private communication are the fastest and most secure systems, but their abilities to survive attacks are limited. Communication Rooms and Peer-to-Peer provide slower, less secure communication but allow for greater security network survival.

Having direct communication, an open communication to all other agents, and private communication directly with other peers creates the "Three Channel Method", which gives the greatest flexibility and control for an agent.

Communications Room

This technique can exist in a large number of protocols and using many types of transports. The idea is simple – all agents in the security network (or, optimally, all agents in a confined area of a security network) communicate directly to each other.

Advantages of a communications room are that agents can communicate and request information, reports, events, tools, and other methods of cooperation from each other without a Command structure present.

A few identified methods of creating communications rooms are:

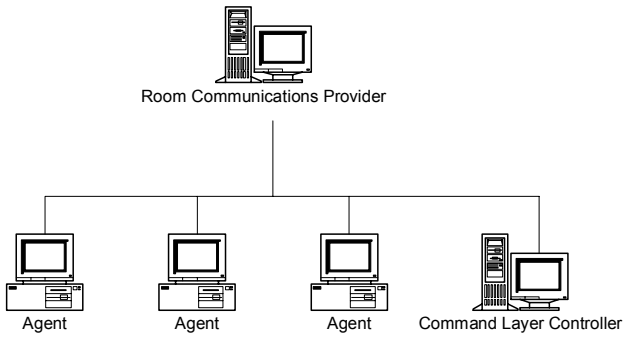
- Designated computer (e.g., one computer handles the communications room)
- Broadcast Protocol
- Relay (more than one computer is responsible for accepting connections into the communications room)

Designated Computer

Because the primary reason for having a communications room is to have a failsafe in the advent of Command Layer failure, the designated computer for controlling communications should not be on the same computer(s) as the Command Layer.

The Agents (Layer 3) and the Command Layer (Layer 1) utilize the designated service by connecting to the sponsoring host and then information is shared. This technique does

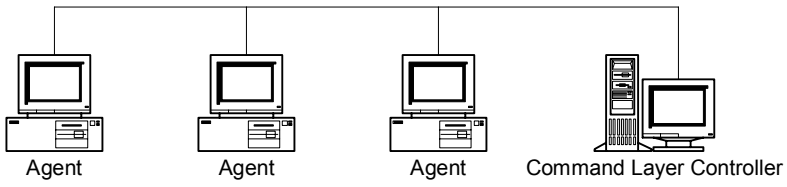
give authentication abilities and security elements for communication in the room. This is a secure communications method, but is subject to a single point of failure.



Broadcast Protocol

The broadcast protocol will send information across the network to all hosts that are listening. Some security precautions can be taken, such as encryption, but overall is the most insecure method. The broadcast protocol can be monitored, communicated with, and connected into by any computer in the range of the broadcast. Also, broadcast protocols are subject to errors and lost data. They are like a radio – if you drive under a bridge with a car, you can't request the radio station to re-transmit the parts of the broadcast you missed while driving under.

The advantage is that the broadcast protocol itself is not subject to direct attack, if the broadcast is rendered inoperative, so is the network itself. Another clear advantage is that this method is extremely fast, and operates at the highest possible speed of the network.



Peer-To-Peer

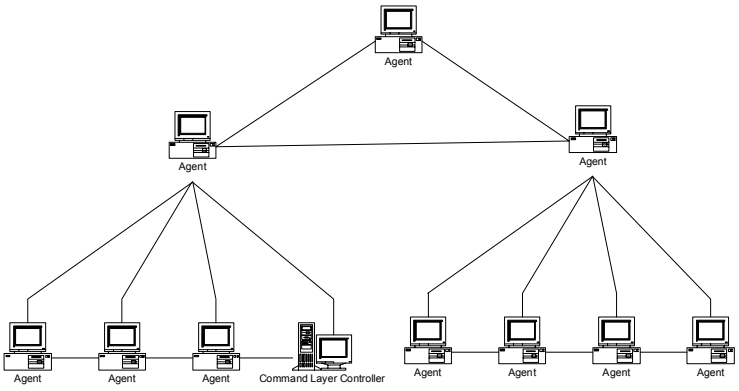
Peer-To-Peer networking broadcasts information to the computers in its designated network neighborhood, and then forwards up information to its 'super-peer' component that continues the traversal across all of the trees.

The advantages of peer-to-peer are that the structure is private and adaptable. It has no reliance on a centralized authority and is difficult to "break". Communications in a peer-to-peer group use error-checking protocols, increasing communication reliability. In the

case of aggressive use of a security network, peer-to-peer structure can hide the identities of the majority of the network addresses of the hostile peers giving a significant 'stealth' advantage. Peer-to-Peer should be considered the best choice for aggressive informational warfare.

The security problems that exist in peer-to-peer environments are numerous because all agents in the enterprise are required to maintain their own security information for authentication into the network for all other hosts. Although some security precautions can be done to prevent easy entry into a peer-to-peer network, typically it's an easy task to locate the necessary credentials on a compromised agent to enter an opposing peer network.

Another disadvantage of peer-to-peer networks is the speed of communication. Each host must communicate through another host in order to send information. This increases the bandwidth required to send information exponentially for each layer.



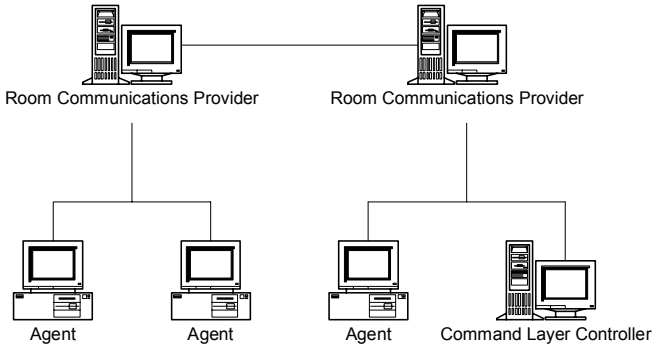
Relay

A relay is similar to the peer-to-peer network except it uses a combination of designated servers that communicate with one another.

The advantages of a relay communications system are that filters can be placed between relays, the system has the advantages of stronger authentication and use security, uses error checking protocols, and has redundancy.

If a relay fails, Agents can connect to a different relay server and reconnect themselves back to the network.

Because of the speed improvements, authentication strength, and reliability, the Relay approach should be considered the best method for a defensive security network.



Private Communication

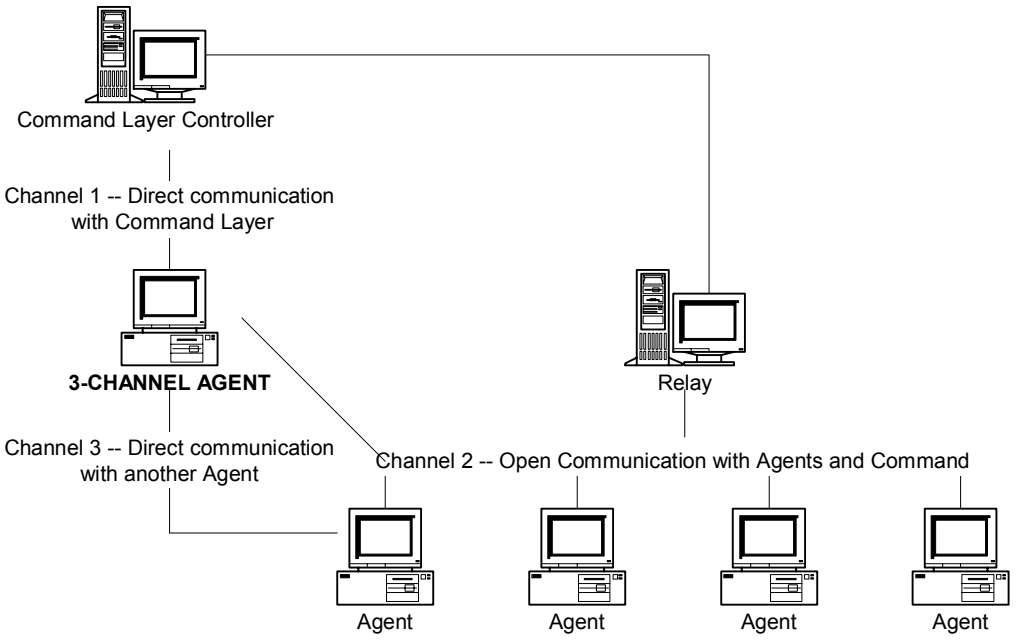
There are several events where agents are better left cooperating with each other than they are directly to the Command Layer or a communications room. For example, if a tool that an agent needs exists on another agent, it can request it from the agent, lessening the workload on the Command Layer.

Reasons for private communication between agents:

- Highly secure, short term communication
- Transfers of needed tools
- Transfers of sensitive information (from 'scuttling' information from hosts)
- Transfers of instructional information (when an agent is forced to relinquish duties)

Three Channel Method

In order to handle the maximum number of security problems an agent will encounter, having three methods of channel communication is the ideal solution. By having a secure channel to the Command Layer, the Agent will be able to receive coordination securely and quickly. In the advent of a Command Layer failure, it will be able to coordinate with other agents to revive the Command Layer by being able to communicate and cooperate directly with other agents in the broadcast channel. Having agent-to-agent direct communication allows for optimum communication and distributed responsibilities to work in the network. In the advent of a broadcast channel failure, the cooperating agents can attempt to revive the broadcast channel or coordinate for a controlled shutdown of the security network.



Security Network Warfare

Agent against agent warfare is only part of the battle because the entire security network is able to influence the outcome of an attack. Most of the war is not fought 'behind enemy lines', it's the *combined capabilities* of the security network that determines the amount of influence the security network has over the battle outcome.

If there are five hundred opposing agents on the aggressive force, and 20 agents on the defending force, but the battle began with one aggressive agent infiltrating the opposing security network, is this really a 1-on-20 battle, or a 500-on-20 battle, or even a series of 500-on-1 battles?

The combined capabilities of the opposing security network give the agent(s) advantages, such as the ability to share calculation efforts. If the attacking agent distributes complex calculations to its security network and lets each agent handle a small piece of the calculation, it will gain an environment advantage over the defending agent.

For example, if it captures an encrypted password file, but it will take 30 minutes to decypher it, chances are the agent will not be able to finish the calculations before it is removed. If the agent sends the password file back to the security network, each agent in its security network can work on a small piece of the password file and then return its results to the agent. The calculation would (in a uniform environment) be 500 times faster than if the infiltrating agent performed the calculations.

Combined Capabilities

The term "combined capabilities" represents the summation of the all of the contributing components in a security network.

Speed of Communication

The ability for agents to cooperate with each other is based on their network speed even more so than their calculation speed. The latency of the network will be one of the most time consuming and battle direction turning events. If it were compared to actual warfare, the 'weather of the battlefield' is the closest comparison. If the network is slow, the agents will be slow.

Combined Calculation

The ability for a security network to provide faster and more accurate analysis than the other security network is key to victory. Many elements need to be considered for combined calculation, such as:

- Fastest computer in the system
- Fastest specialized computer for a function (special encryption chips, etc.)
- Parallel computation speed
- Mission size
- Tool speed ratings

Combined calculations for a security network will ultimately need a well thought out rating system. For the sake of this publication, I'll assume it's a time related value with "1" being the first measurable speed of the first implemented informational warfare security network, when it becomes built and tested.

A security network with a combined calculation rating (say, "10") will probably find itself ripped to pieces by an extremely clever security network with a combined calculation rating of "1,000,000".

Robustness of Tools

If a security network has all the tools required to counter "tool for tool" the opposing security network, then the possibility exists for the security network to win the battle with very little opposition. If only some of the tools exist, it may still be possible for the opposition to be dispatched or held at bay by proper use and placement of tools that it has.

Artificial Intelligence

Having tools is important, but how they are used to win the battle is equally important. If an agent is 'behind enemy lines' and on a very mission critical computer, decisions need to be made, in accordance with the tools at its disposal, on the best way to end the battle. The fastest way might be to shut off the computer, but that would result in millions of dollars of business losses. Both attacking and defending security networks need to be able to have a fast, accurate, and dynamic A.I.

Combined Calculation Danger Rating

There is currently no 'magical formula' of combine capabilities that will ever truly guarantee a victory over an opposing security network, but a rating system can give an indication of the power.

The creation of a formula (as well as a way to deduce the strength of a security network by observing its combat characteristics) needs to be study of future research. However, such a rating system might appear like Fujita scale for rating tornado strength.

The value would be a combined rating of the power of all possible combined characteristics multiplied by each other multiplied by one another. A baseline of "1" could to be established for a recognized standard system of the day and adjusted accordingly each year.

Level	Intensity	Size
1	Mild	1-20 computers, below average ability, readily available network speed
2	Average	21-250 computers, known technology, fastest commercial network speed
3	Strong	251-2000 computers, using some advanced technology, several fast network channels.
4	Powerful	2000-50,000 computers, using very advanced technology, hundreds of fast network channels.
5	Overwhelming	50,000+ computers, remarkable technology, thousands of fast network channels.

In comparison to the present day, the "Slammer" worm might be a "5" because it archived overwhelming size, although its technology wasn't tremendously advanced.

The combined capabilities of "Slammer" were certainly overwhelming. Were it more intelligent and could select multiple targets or have a control interface, it would have utterly decimated the Internet.

Complexities of the Mission

The number of tasks and the time taken for each task determines the complication level of the mission. If the mission is to perform a single, quick task, the mission is simple and has a higher chance of success. The defending security network automatically is considered to have a complicated mission, with no time limit. The attacking security network has a relatively brief list of objectives to complete its mission, and therefore will be able to perform its functions faster. The more objectives created for an attacking security network will complicate its decision-making ability.

The attacking security network already has an implied objective of environment discovery of the defending subnet, which will require a significant amount of resources for the attacking subnet to eliminate.

Natural Warfare Advantages

As given in many previous examples, both attackers and defenders have advantages and disadvantages where it comes to informational warfare. This section describes various types of attacking and defending advantages.

Attacking

The proverb goes, "Its easier to destroy than it is to create." Attacking security networks have many powerful natural advantages over defending security networks. However, the advantages are never a guarantee of an outcome.

Ambush Advantage

When a security vulnerability is discovered that allows for an infiltration, it takes time for the patches and information about the problem to become known. It has been estimated that for every security vulnerability that is announced publicly, there are five more that aren't ever released to the public. Also, the attack doesn't have to begin with a known security hole, as the users of the computer network can be manipulated or have a desire to introducing a hostile agent.

Mission Advantage

The defending security network has a very broad mission of defense and protection of all the components its responsible for. Attacking security networks may have a limited and less calculation intensive mission and certainly will have less management responsibilities.

Deterioration Advantage

If the battle starts as a 500-on-20, and a host is compromised, the compromised host may be added to the opposition's capabilities, making it a 501-on-19 battle. Each computer removed from a security network lessens its capabilities.

Anonymity

The attacking security network will appear as if its coming from anywhere in the Internet, and the Command Layer will ideally be far and distant from the infiltrating agent. Due to the Legal concerns, counter-attacks are currently illegal and so the defending network is forced to take a beating.

If computer counter-attacks for self-defense is legalized (which may ultimately be the case in the future), the construct of an attacking security network will be built to confused, hide, and change its origins. The attacker might not be easily provable in a court of law.

Siege Advantage

If the "enemy is at the gates", in other words at the access point to the security network being attacked, the attacking security network has the ability to control, disrupt, and monitor the outside communications.

Defending

Although the attacking security network has natural protection and surprise, a properly implemented defending network should still seem like an impenetrable fortress. If done correctly, the entry way might quickly become a jail cell.

Preparation Advantage

The defending network begins with a complete awareness of its environment, including all the security privileges to defend itself. The attacking security network will need to gather the resources and privileges as its able to in order to gain control of the resources inside of the defending security network.

Likewise, an assessment should have already been made as to the tools necessary to protect the environment and are already present within the defensive security network.

Network Speed Advantage

The opposing security network will be through a slower network link than the defending network, and will not be able to gain a network speed advantage unless it successfully infiltrates and expands into the defending network.

Awareness Advantage

The defending network is completely aware of the 'battlefield', and presumably has already created numerous plans in case of attack. Valuable assets have already been identified and the agents have already been given orders on how to protect them.

Design Advantage

If the defending security network has redundancies, hidden communication paths, and so forth, the defending network will have options available to it that the attacking network will be unaware of. Internal firewalls, secondary network paths, information guards, and such all play strong roles in design advantage.

Cyber-Pandemonium

The basic building blocks have been created for attacking security networks against defending security networks, but there is a very large "what if?" a battle ensues against two or more aggressive security networks that are fighting for control of the open landscape of the Internet.

In the present day, the effect is seen with network worms, first created in 1988 by Robert Morris, Jr., worms have shown that incredible numbers of computers can be compromised. With an intelligent security network guiding their propagation, the damages can be amplified by several factors – the first of which is that the speed of the attack can be increased immensely, secondly the duration the computers are infected will increase, and third the control over the compromised computer will increase.

A properly crafted informational warfare-model security network could easily reach a size of 10,000,000+ computers, and in which case, its capabilities would be so immense that any target it desires would be annihilated except one – other aggressive security networks.

When two aggressive security networks collide, the event becomes cyber-pandemonium, and everyone on all public networks is going to feel the effects. Aggressive forces will collide, infiltrate, and vie for dominance over the entire Internet. Defending forces will protect their ground and do their best to confound the aggressors.

A security network of ten million captured computers would have severe logistics problems, constantly re-assessing its capability structure and dividing its command system into thousands of layers. Centralization would become problematic, and considerable lack of concern in the mission would exist for the conscripted computers security. The human controller(s) of the security network would have ample time to change missions, select targets, and introduce enhancements and improvements to alter the battle strategy.

The intelligence of the dominant security network may be able to thwart even a specially tailored counter-attack. The idea that a 'counter-worm' can be quickly developed and

spread across the Internet is also assuming that the dominant security doesn't also have the ability to realize its being attacking. With agent technology, it is very likely that repeated successful attacks, the A.I. structure and human influence that will be able to deduce a signature and produce a blockade.

For example, the dominant structure has vulnerability in its peer-to-peer mechanism that allows an agent to be shut down. A relationship develops between an outside connection to the peer connection port and the disappearance of node. Heuristic analysis for a human influence will quickly spot the connection, and a solution can be created and given to the Command Layer than quickly propagates through the security network.

It has been theorized that a worm can fight a worm, and many people have approached me in the past asking me if they could "write a virus that removes a virus". I've dismissed the idea because of legal concerns and randomness, but in the event of cyber-pandemonium, it may be the only way to restore order.

As would be the case in real warfare, each battle that ensues would be based on conditions that outreach single vulnerabilities or techniques, and 'the fog of war' would be introduced to what has previously been a cold and sterile computational environment. There would be too many human factors being influenced during the event of cyber-pandemonium to allow the raging wars to go unchecked.

On a global scale, cyber-pandemonium would be considered a true act of International War. The shock of the first time a broad scale event occurs, its initiator(s), its participants, and its victors is going to reshape international relations at all levels. If the American government, for example, released a counter-strike and successfully gained Internet Dominance, the only result would be resentment from countries and their populations that don't want to have their computers controlled by American security agents, and rightfully so because they were installed by force.

Computers and agents are very unaware and unable to sort all the Internet addresses from physical locations, and many proposals will be introduced to create some form of legalized barriers for computer influence. A possible solution might be to add a global positioning system (GPS) to all computers so that their location on the globe can be identified. Once again, this technique will not mean anything to rogue security networks, and other proposals for International firewalls will be created and so forth.

If the cost of Informational Warfare is currently approaching the $500 million/year damages level, as reported by the FBI, and the stock value of a single company can diminish $20 billion from an attack, then future attacks may reach the $100 billion/per attack levels in the event of cyber-pandemonium and cause incredible financial shifts capable of creating recessions or prosperity.

This is, unfortunately, the incentive for creating aggressive security networks and given the demonstration that the building of such networks is a low-cost warfare component, the prediction of cyber-pandemonium should be considered "inevitable."

Conclusion

What will become the ultimate frustration for administrators is that regardless if informational warfare is on a grand scale or a small person-to-person skirmish, the landscape and its problems are infinite and the technology to produce the tools of this form of warfare are inexpensive and readily available.

The battles that can ensure between security networks based on the Informational Warfare model are not guaranteed success for failure based on a single vulnerability or condition. People who are using products based on the informational warfare model for defense are capable of recovering or preventing even unknown attacks and therefore get significantly improved protection.

Of the tools described in this book, it is obvious that the majority of them already exist and the ones that have not yet been created are not conceptually difficult. Example code is readily available on the Internet, and public and global awareness of the effect of cyber-terrorism has completely ruined any chances of containment.