Information Assurance through Kolmogorov Complexity

Scott Evans, Stephen F. Bush, and John Hershey
GE Corporate Research and Development
evans@crd.ge.com

Abstract

The problem of Information Assurance is approached from the point of view of Kolmogorov Complexity and Minimum Message Length criteria. Several theoretical results are obtained, possible applications are discussed and a new metric for measuring complexity is introduced. Utilization of Kolmogorov Complexity like metrics as conserved parameters to detect abnormal system behavior is explored. Data and process vulnerabilities are put forward as two different dimensions of vulnerability that can be discussed in terms of Kolmogorov Complexity. Finally, these results are utilized to conduct complexity-based vulnerability analysis.

1. Introduction

Information security (or lack thereof) is too often dealt with after security has been lost. Back doors are opened, Trojan horses are placed, passwords are guessed and firewalls are broken down – in general, security is lost as barriers to hostile attackers are breached and one is put in the undesirable position of detecting and patching holes. In fact many holes go undetected. Breaches in other complex systems that people care about are not handled in such an inept manner. Thermodynamic systems, for example, can be assured of their integrity by the pressure, heat or mass the system contains. Hydrostatic tests can be performed to ensure that there are no "holes", and the general health of the system can be ascertained by measuring certain parameters. One doesn't wait, for example, for all the water to drain out of a heat exchanger and a rat to come inside to announce that there is a problem. A problem is identified as soon as the temperature or pressure drops and immediately one can take action to both correct the problem and to isolate other areas of the system from harm. But does one perform a hydrostatic test of an information system? What conserved parameters exist to measure the health or vulnerability of the system? How can one couple the daunting task of providing a system where vulnerabilities are readily measurable with the required need for simplicity of use for authorized users? This paper explores these issues and proposes that only through monitoring objective quantities inherently related to information itself can the science of information assurance move beyond patching holes.

Kolmogorov Complexity is proposed as a fundamental property of information that has properties of conservation that may be exploited to provide information assurance. In this paper, Kolmogorov Complexity is reviewed and current work in this area explored for possible applications in providing information assurance. The concept of Minimum Message Length is explored and applied to information assurance, yielding examples of possible benefits for system optimization as well as security that can be achieved through the use of Kolmogorov Complexity based ideas. Finally, complexity based vulnerability analysis is demonstrated through simulation.

2. Background

Currently information security is achieved through the use of multiple techniques to prevent unauthorized use. Encryption, authentication/password protection, and policies all provide some level of security against unauthorized use. But other than simply relying on these secure barriers, how does one measure the health of their security system. If a password or encryption key is compromised, what indication will be available? The degree to which a system is compromised is difficult to ascertain. For example, if one password has been guessed, or two encryption keys determined, how secure is the information system? Are all detectable security issues equal, or are some more important than others? These difficulties reflect the fact that there is no objective, fundamental set of parameters that can be evaluated to determine if security is maintained. Insecurity may not be detected until an absurd result (rat in a tank) discloses the presence of an attacker. An inherent property of information itself is desired that can be monitored to ensure the security of an information system. The descriptive complexity of the information itself – the Kolmogorov complexity is a strong candidate for this purpose. Kolmogorov Complexity is reviewed in the following section.

This effort is funded by DARPA ISO contract number F33615-00-C-1629. Accepted for publication at the DARPA Information Survivability Conference and Exposition II (DISCEX-II 2001) to be held 12-14 June 2001 in Anaheim, California.

2.1. Kolmogorov Complexity

Kolmogorov Complexity is a measure of descriptive complexity contained in an object. It refers to the minimum length of a program such that a universal computer can generate a specific sequence. A good introduction to Kolmogorov Complexity is contained in [3] with a solid treatment in [4]. Kolmogorov Complexity is related to Shannon entropy, in that the expected value of K(x) for a random sequence is approximately the entropy of the source distribution for the process generating the sequence [3]. However, Kolmogorov Complexity differs from entropy in that it relates to the specific string being considered rather than the source distribution. Kolmogorov Complexity can be described as follows, where φ represents a universal computer, p represents a program, and x represents a string:

$$K_{\varphi}(x) = \left\{ \min_{\varphi(p)=x} l(p) \right\}.$$

Random strings have rather high Kolmogorov Complexity – on the order of their length, as patterns cannot be discerned to reduce the size of a program generating such a string. On the other hand, strings with a large amount of structure have fairly low complexity. Universal computers can be equated through programs of constant length, thus a mapping can be made between universal computers of different types, and the Kolmogorov Complexity of a given string on two computers differs by known or determinable constants. The Kolmogorov Complexity K(y|x) of a string y given string x as input is described by the equation below:

$$K_{\varphi}(y \mid x) = \left\{ \begin{array}{c} \min\limits_{\varphi(p,x)=y} l(p) \\ \infty, \textit{ if there is no } p \textit{ such that } \varphi(p,x) = y \end{array} \right\},$$

where l(p) represents program length p and φ is a particular universal computer under consideration. Thus, knowledge or input of a string x may reduce the complexity or program size necessary to produce a new string y.

The major difficulty with Kolmogorov Complexity is that you can't compute it. Any program that produces a given string is an upper bound on the Kolmogorov Complexity for this string, but you can't compute the lower bound [4]. A best estimate of Kolmogorov Complexity may be useful in determining and providing information assurance due to links between Kolmogorov Complexity and information security that will be discussed later. Various estimates have been considered, including compressibility, or pseudo-randomness, which measure the degree to which strings have patterns or structure. A new metric that is related to the power spectral density of the sequence auto-correlation is introduced in Section 4. However, all metrics are at best crude estimates. The inability to compute Kolmogorov Complexity persists as the major impediment to widespread utilization.

Despite the problems with measurement, Kolmogorov Complexity and information assurance are related in many ways. Cryptography, for example attempts to take strings that have structure and make them appear random. The quality of a cryptographic system is related to the systems ability to raise the apparent complexity of the string, an idea discussed in detail later, while keeping the actual complexity of the string relatively the same (within the bounds of the encryption algorithm). In other words, cryptography achieves its purpose by making a string appear to have a high Kolmogorov Complexity through the use of a difficult or impossible to guess algorithm or key.

Security vulnerabilities may also be analyzed from the viewpoint of Kolmogorov Complexity. One can even relate insecurity fundamentally to the incomputibility of Kolmogorov Complexity and show why security vulnerabilities exist in a network. Vulnerabilities can be thought of as the identification of methods to accomplish tasks on an information system that are easier than intended by the system designer. Essentially the designer intends for something to be hard for an unauthorized user and the attacker identifies an easier way of accomplishing this task. Measuring and keeping track of a metric for Kolmogorov Complexity in an information system provides a method to detect such short-circuiting of the intended process.

2.2. Minimum Message Length Principle

Since it is not computable, few applications exist for Kolmogorov Complexity. One growing application is a statistical technique with strong links to information theory known as Minimum Message Length (MML) coding [8]. MML coding encodes information as a hypothesis that identifies the presumptive distribution, from which data originated, appended with a string of data, coded in an optimal way. The length of an MML message is determined as follows:

$$\#M = \#H + \#D,$$

where #M is the message length, #H is the length of the specification of the hypothesis regarding the data, and #D is the length of the data, encoded in an optimal manner given hypothesis H. As discussed in [8], MML coding approaches the Kolmogorov Complexity or actual bound on the minimum length required for representing a string of data.

3. Conserved Variables

Conserved variables enable one to deduce parameters from the presence or absence of other parameters. The Law of Conservation of Matter and Energy [1] for example allows one to deduce how well a thermodynamic system is functioning without knowing every parameter in the system. Heat gain in one part of the system was either produced by some process or traveled from (and was lost from) another part of the system. One knows that if the thermal efficiency of a thermodynamic system falls below certain thresholds then there is problem. On the other hand, if more heat is produced by a system than expected, some unintended process is at work. A similar situation is desirable for information systems – the ability to detect lack of assurance by the presence of something unexpected, or the absence of something that is expected. This seems to be far from reach, given that information is easily created and destroyed with little residual evidence or impact.

One possible candidate for a conserved variable in an information system is Kolmogorov Complexity. Suppose you could easily know the exact Kolmogorov Complexity K(*S*) of a string of data *S*. You would essentially have a conserved parameter that could be used to detect, resolve or infer events that occur in the system, just as tracking heat in a thermodynamic system enables monitoring of that system. Operations that affect string S and cause it to gain or lose complexity can be accounted for, and an expected change in complexity should be resolvable with the known (secured) operations occurring in the information system to produce expected changes in complexity. Complexity changes that occur in a system that cannot be accounted for by known system operations are indications of unauthorized processes taking place. Thus, in the ideal case where Kolmogorov Complexity is known, a check and balance on an information system that enables assurance of proper operation and detection of unauthorized activity is possible. Unfortunately (as previously discussed) a precise measure of Kolmogorov Complexity is not computable. We can, however, bound the increase in Kolmogorov Complexity as shown in the theorems below.

3.1. Theorems of Conservation

Kolmogorov Complexity, K(x), can be thought of as a conserved parameter that changes through computational operations conducted upon strings. In order for K(x) to be a conserved parameter one must account for changes in K(x). Two theorems are presented below that enable bounds to be placed on the changes in K(x) that occur due to computational operations occurring in an information system. The two theorems below show bounds on the amount of complexity that can exist due to knowledge of other strings or computational operations.

3.1.1. Theorem 1: Bound on Conditional Complexity

$$K_\varphi(y \mid x) \le K_\varphi(y)$$

Proof

Since K(y) is the minimal length program that produces string y with no input, input x can only reduce the length of the program required to produce y. At worst, x can be ignored completely, in which case K(y|x)=K(y). However, knowing x may reduce the program to produce y, depending on the extent that string x contributes towards generating string y or enables a more efficient generation of y. QED

3.1.2. Theorem 2: Bound on Complexity Increase Due to Computational Operation

$$K_\varphi(y \mid x, p) \le K_\varphi(x) + L(p)$$

Proof

Program p of length L(p) takes input string x to produce output string y. Proof by contradiction: consider a program p that could be run on input string x to produce string y. Assume that the complexity of y = K(y|x,p) > K(x) + L(p). But one could produce string y by first forming string x with program of length K(x), then running program p of length L(p), thus producing y with a program of length K(x) + L(p). But this violates the definition of Kolmogorov Complexity as being the minimum length program, since a program of smaller length has been found. Thus the assumption is false and K(y|x,p) must be <= K(x) + L(p). QED

3.3. Conservation of Complexity

As shown above, while not computable from below, upper bounds on the increase in Kolmogorov Complexity can be crudely known by keeping track of the size of programs that affect data. This bound may be incredibly loose, as it is quite possible to operate on a string and make it much less complex than the input. One would need a method to recognize this simplification. However, these results provide an intuitively attractive method for quantifying the "work" performed by a computational operation on information – the change in complexity introduced by the operation. A thorough treatment of bounds related to K(y|x) and the "Information Distance" between strings is contained in Bennett et al.[9].

4. A Measure for Binary String Complexity

As previously discussed, due to its non-computable nature, estimates of K(x) are difficult. Numerous

techniques for estimating K(x) are discussed in [4]. The task of estimating K(x) is related to the task of assessing string structure. A new primitive approach to this related issue is introduced based on the power spectral density of a string's auto-correlation. This approach highlights the ability to gain knowledge of K(x) without any higher knowledge about the system producing string x or the meaning of the information.

Recognizing that the complexity of a binary string may be defined in many ways. A useful complexity measure may be related to properties of the string's non-cyclic auto-correlation. Specifically, given an n-bit binary string, S, where:

$$S = \{s(i)\}, 0 \le i < n,$$

and

$$s(i) \ni \{\pm 1\} \ \forall i .$$

Define the non-cyclic auto-correlation, R, as:

$$R = \{r(i)\}, 0 \le i < n$$

where

$$r(i) = \sum_{j=0}^{n-i-1} s(j)s(i+j) .$$

From R, calculate the sequence's non-negative power spectral density, Φ_i, by multiplying the Fourier transform of R by its conjugate. The measure for binary string complexity that is formed is denoted by Ψ and is defined as

$$\Psi = \frac{1}{norm.\ factor} \sum_i \Phi_i \log \Phi_i .$$

The motivation to this approach is found in the rich and venerable field of synchronization sequence design. Sequences that have an auto-correlation who's side-lobes are of very low magnitude provide good defense against ambiguity in time localization. Such an auto-correlation function will approximate a "thumbtack" and its Fourier transform will approximate that of band-limited white noise.

The authors of this paper expect that Ψ will be of utility in assessing complexity as it relates to the compressibility of a binary string. To begin the testing of this hypothesis, strings are generated from the Markov process diagrammed in Figure 1. A series of binary sequences of 8000 bits were generated, each for different values of p. Ψ was computed for each of these strings and also packed into 1000-kilobyte files. These were subjected to the UNIX compress routine. The Inverse Compression Ratio (ICR) was computed which is the size of the compressed file normalized to its uncompressed size, 1000 kilobytes in these cases.

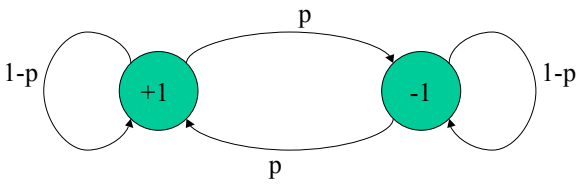

Figure 1. Markov model for string generation.

The hypothesis is that Ψ and the ICR should vary in a similar manner and that Ψ might be a useful measure of sequence compressibility and hence complexity. The graph in Figure 2 seems to endorse this hypothesis and further research is motivated.

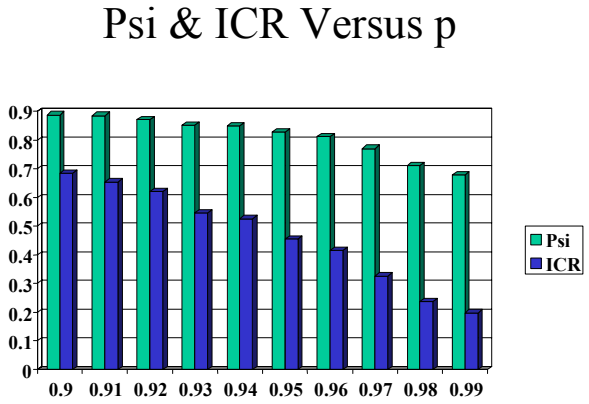

Figure 2. Variation of Psi and ICR with p.

The above results show that fundamental parameters such as power spectral density of sequence auto-correlation and compressibility are related and follow similar trends. These fundamental metrics are possible candidates for measuring trend of increase or decrease in K(x). However, also illustrated by these results (the unequal rate of change between the two metrics) are the loose bounds within which estimates of K(x) are related. Other methods of estimating K(x) are described in [4]. In the next section we introduce a method for attacking the issue of loose bounds in order to make complexity metrics useful for the purposes of assessing and providing information assurance.

5. Apparent Complexity

The results in Section 3 give an upper bound on complexity increase due to computational operations, but perhaps one can do better. In fact, the size of the shortest program one can find to produce a particular string is the best estimate for K(S). Since Kolmogorov Complexity is unknowable, the best that we can do is estimate well. This motivates the idea of apparent complexity: the best attainable estimate of K(S) given the limited information available to a particular party. The benefit or possible way to exploit the idea of apparent complexity is that a user generating a string should have the best idea of how hard it is to generate the string. There are many reasons why a user may not choose to generate a string using the minimal size program. Perhaps a longer program can execute faster, or perhaps the generator is unknowingly using an inefficient process. However, the generator of a string of data is presumed to have knowledge of the process used to generate that data. This may in fact make the non-computability of Kolmogorov Complexity an asset: a good candidate for use in providing information assurance, for the follow reason. The information system designer or an authorized user generating data should have better knowledge of the data process than an attacker. An attacker cannot simply compute the optimal process. Additionally, conservation of apparent complexity enables abnormalities to be tracked when the expected number of computational operations is not utilized in transforming string x into string y. Thus, even if one cannot know or compute the most efficient process for creating a string of data, one can at least gain benefit from ensuring through monitoring resources that the expected process is used. This type of assurance has in fact been used informally to detect network security problems for many years. Discrepancies in computer account charges have lead to detection of attack [10]. The idea of using Kolmogorov Complexity provides the possibility of using this type of technique on a more fundamental level where knowledge about the information content would not be required to determine unauthorized activity. The term apparent complexity will be used to reflect the best measurement of Kolmogorov complexity available to the party undertaking the measurement.

5.1 Process vs. Data Complexity

Apparent complexity can be applied to the problem of information assurance in two ways. As discussed above, conservation of apparent complexity may enable detecting and correcting abnormal behavior. Another method of using apparent complexity for information assurance is in the identification of weak areas or vulnerabilities in the system. Consider the postulate that the more apparently complex the data, the more difficult for an attacker to understand the data and exploit the system. Thus, the more apparently complex, the less vulnerable and vice versa. One proposed metric for vulnerability relates to evaluating the apparent complexity of the concatenated input and output K(X.Y). This relates to the joint complexity of the data input and output from a certain process (Black Box). The lower the complexity of K(X.Y) the easier the data is for an attacker to understand, thus we will regard K(X.Y) as a measure of data vulnerability. A competing metric is the relative complexity K(Y|X) of the process. This is the "work" done on the information by the process, or the complexity added to or removed from X to produce Y. Thus K(Y|X) is a measure of process vulnerability. The relationships between these respective complexity metrics and the black box process is shown in Figure 3.

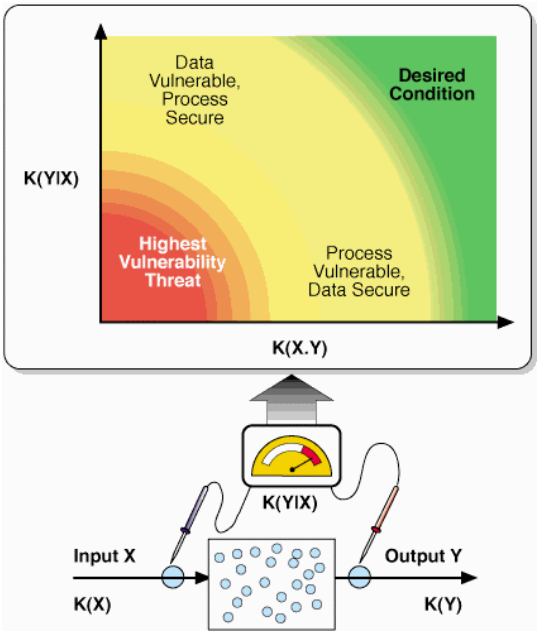

Figure 3. Process vs. data vulnerabilities.

Data vulnerability relates to how vulnerable a system is to an attacker knowing information. This type is perhaps best measured by K(X.Y), where the cumulative complexity of input and output data is observed to measure the difficulty an attacker would face in decrypting or identifying messages contained in input and output. For example, hopefully K(encrypted message) appears >> K(decrypted message) to the casual observer and is only recognized to be on the order of K(decrypted message) to an authorized user with the correct key after the decryption algorithm has been run.

Process vulnerability relates a system's susceptibility to an attacker understanding the processes that manipulate information. This vulnerability is best quantified by the

complexity injected or removed from the data by the process at work. For example, a copy or pass-through process adds little complexity, K(Y|X) is zero. But if encrypted data is sent through the copy process, K(X.Y) will be high. The attacker will be unable to discern the messages that are sent, but can learn to perhaps simulate this particular black box quite effectively. Whereas if plain text data is sent through the copy process K(X.Y) will be low, and in addition to understanding the process at work, and attacker may be able to know the particular messages that are sent. Both vulnerabilities are undesirable and represent two different dimensions of vulnerability to be avoided. To make systems secure one must maximize both process and data complexity to a non-authorized user while keeping the systems simple to authorized users. Proper accounting of K(Y|X) and K(X.Y) throughout the system will enable both identification of weak areas as well as identification of foul play through the conservation principles discussed earlier.

6. Vulnerability Reduction by means of System Optimization

In this section issues related to system optimization that can be achieved through Kolmogorov Complexity and various related tradeoffs are discussed. Compression and security are strongly linked in that they are bounded optimally by the most random sequence that can be produced. But smallest program size is not the only or even most important performance metric. Execution time is and example of another metric that must be considered. The tradeoff is indicated in Figure 4, where the possibility of having programs with large K(X) and small execution time and vice versa are highlighted.

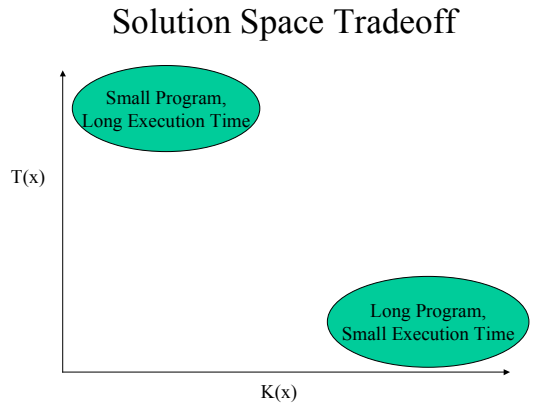

Figure 4. Program size vs. speed tradeoffs.

Through the use of active network techniques [11] the tradeoff indicated may be dynamically addressed using a concept called Active Packet Morphing for network optimization. As shown in Figure 5, changing the form of information from data to code as information flows through a system can optimize CPU resources and bandwidth resources.

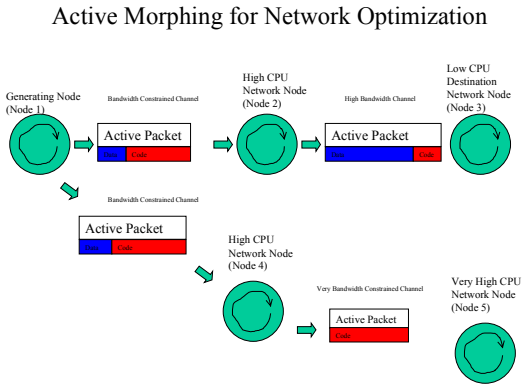

Figure 5. Active packet morphing for network optimization.

This idea can be extended to optimize or prevent adverse effects from critical resources in addition to bandwidth and CPU. Memory, time of execution or buffer space could be use to trade off forms of data representation to optimize certain system parameters. The ability of data to change form within a system opens up multiple optimization paths that were previously invariant in the system. Rigorous security quantification resulting from this work allows active packets to morph by adding the required security overhead along specific communication links such that the security of the link along with the security of the morphed packet yield the proper level of security required by a given policy. Thus, security overhead is minimized.

Another parameter that can optimize system resources is the knowledge of how a piece of data is used. MP3 audio is a good example of how leaving out information (specifically that which is undetectable by the human ear) can optimize data size. We introduce here the idea of "necessary" data to augment the idea of "sufficient" data or sufficient statistics that represent all information contained in the original data [3]. Sufficient representation of data contains all the information that the source data contains. Necessary data contains only the information that the source data contains that the destination instrument can effectively use. If one efficiently encapsulates all the information in source data in a statistical parameter one may have achieved a minimum sufficient statistic. If one further reduces this statistic such that one encapsulates only the information that is usable by the end node, one has obtained the minimal necessary sufficient statistic. Thus, Kolmogorov

Complexity related ideas have tremendous impact for system optimization as well as security.

7. Automated Discovery of Vulnerabilities without *a priori* Knowledge of Vulnerability Types

An information system can be designed in such a manner that the apparent complexity of the system under attack can be determined with respect to the attacker and that information used to maximize the distance in the apparent complexity between the attacker and defenders in an automatically reconstituted system. An Active Network [11] is an ideal environment in which to experiment with an implementation of automated system reconstitution because it provides extreme flexibility in fine-grain code movement and composition of code. Apparent complexity is used to reconstitute the system such that the complexity difference is maximized between legitimate users and attackers of the system. In this section, the discussion is limited to the automated hardening of a system based upon information about an attacker and a new form of vulnerability analysis, called complexity-based vulnerability analysis.

The motivation for complexity-based vulnerability analysis comes from the fact that vulnerability analysis tools today require types of vulnerabilities to be known *a priori*. This is unacceptable, but understandable given the challenge of finding all potential vulnerabilities in a system. Information assurance is a hard problem in part because it involves the application of the scientific method by a defender to determine a means of evaluating and thwarting the scientific method applied by an attacker. This self-reference of scientific methods would seem to imply a non-halting cycle of hypotheses and experimental validation being applied by both offensive and defensive entities. Information assurance depends upon the ability to discover the relationships governing this cycle and then quantifying and measuring the progress made by both an attacker and defender. This work attempts to lay the foundation for quantifying information assurance.

Quantification is necessary because tools have been developed to measure and analyze security assuming rigorously defined security metrics exist. See [12] for an example of such a tool whose sample vulnerability chain output is shown in Figure 6. The numbers shown in Figure 6 are opportunities for an attacker to move across vulnerabilities. More precisely, tools such as these rely on an "insecurity flow" metric. However, a rigorously defined metric has not yet been derived. One focus of this paper is upon mathematically quantifying and refining insecurity flows. It is extremely important that a proper metric space is chosen, because the entire foundation of Information Assurance will rest upon this space. Particularly notable is the fact that relationships involving physics of information are being developed whose operations will be facilitated by the choice of metric.

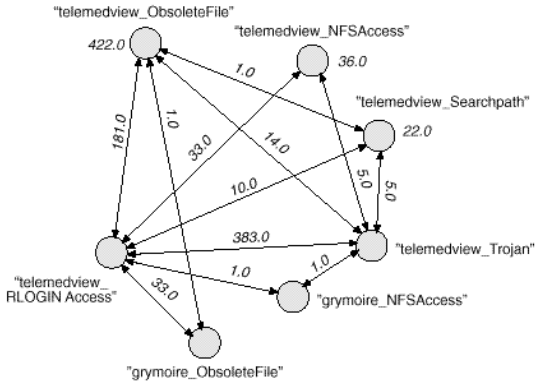

Figure 6. Vulnerability results from an analysis tool.

Any vulnerability analysis technique for information assurance must account for the innovation of an attacker. Such a metric was suggested about 700 years ago by William of Ockham [13]. Ockham's Razor has been the basis of much of this paper and the complexity-based vulnerability method to be presented. The salient point of Ockham's Razor and complexity-based vulnerability analysis is that the better one understands a phenomenon, the more concisely the phenomenon can be described. This is the essence of the goal of science: to develop theories that require a minimal amount of irrelevant information, all the knowledge required to describe a phenomenon should be algorithmically contained in formulae.

7.1 Mozart and Vulnerability Analysis

Science is art and art is science. One of the most mathematical of art forms is the composition of music. Music is compressed and transported over the Internet very frequently and most listeners of such music probably have little interest in the compression ratio of a particular piece of music. However, this piece of information can be very interesting and informative with regard to the complexity of a piece of music. One would expect an incompressible piece of music to be highly complex; perhaps bordering on random noise; while a highly compressible piece of music would have a very simple repetitive nature. Most people would probably prefer music that falls in a mid-level range of complexity; sounds that are not repetitious and boring yet not random and annoying, but follow an internal pattern in the listeners' minds. Music is a mathematical sequence that the composer is posing to the listener; the more easily the listener can extrapolate the sequence without being too challenged or too bored the more pleasing the music sounds. Carrying the music analogy forward in a more

explicit manner, consider the listener as an attacker and the composer as the designer of an information system. If a user has a preference for a given type of music, a sample of that music can be included as a hypothesis in an MML-based complexity analysis. The lower the complexity, the more appealing the music to that particular listener. The more easily the listener can extrapolate the musical sequence, the more vulnerable the system.

Imagine the composer who wishes his music to be enjoyed by only a specific group of listeners and no others. The composer is constrained from generating a completely invulnerable system, that is, totally random, because the composer wants the music to be meaningful to at least some potential group of listeners. Relating this analogy to the hydrostatic test that was mentioned in the introduction to this paper, vulnerability is the quantification of the potential leakage of music that is enjoyable to unintended listeners.

In a very quick experiment, the following three pieces of music were tested for complexity: Beethoven's Sonata Op. 27, No. 2 ("Moonlight"), Mozart's Sonata in A Major ("Alla Turka"), and Philip Glass's Opening to "Glassworks". The encoding explicitly represents notes, timing, dynamics, and phrasing. Beethoven's Moonlight Sonata has a complexity rating of 0.13, Mozart's Sonata has a complexity rating of 0.16, and Glass's Introduction to Glassworks has a complexity of 0.03 based upon then inverse compression ratio. Philip Glass makes extreme use of repetitious arpeggios in his work, thus the low complexity rating. One of the authors expected Beethoven to have a slightly higher complexity than Mozart by a small amount, but it was the reverse in this case. Note that one could decompose the overall complexity to determine the complexity of a composer's use of rhythm, note structure, phrasing, or other musical components. Once a composer's typical complexity band is benchmarked; this type of analysis could be used as an indicator to determine authenticity. Additionally, one could conjecture that "learning" to model Beethoven or Mozart might be more difficult than other composers.

7.2 Demonstration of Complexity-based Vulnerability Analysis

Experimental validation of complexity-based vulnerability analysis has begun using a model information system that has been implemented in Mathematica. The goal is to determine the vulnerability, not only of the overall system, but also of system components. Vulnerability analysis should be done without any *a priori* knowledge about system operation or knowledge of particular types of vulnerabilities. Relating this mechanism to the hydrostatic testing mentioned in the introduction to this paper, one does not have to understand the motion and dynamics of every molecule in order to determine a leak. Expert systems and vulnerability analysis tools that rely upon rules identifying particular types of vulnerabilities, in other words, attempting to understand every intricate detail of a system and its vulnerabilities, are inherently brittle and, in fact, meaningless against an innovative attacker. Thus, the Mathematica information system model shown in Figure 7 purposely does **not** include component descriptions or explanations because the goal is to analyze the system as a black box with respect to vulnerability. The point is that vulnerability analysis can be done without having to know the details of the system. At the end of this section, the functions of the analyzed components are mentioned. It should then make intuitive sense that a particular component that was performing a simple operation had a lower complexity than one performing a more complex operation.

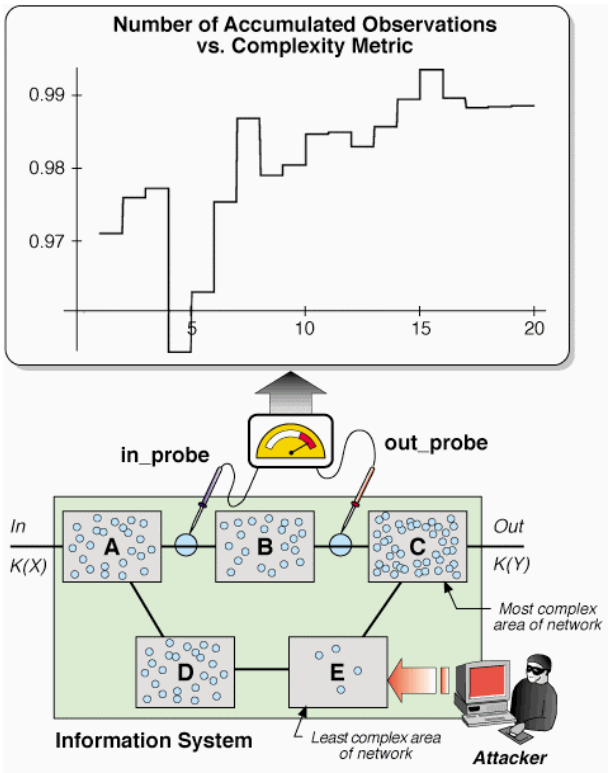

Figure 7. Mathematica System Model Components.

Each component of the Mathematica information system contains probe points through which bit level input and output can be collected. A complexity function based upon a simple inverse compression ratio is used as an estimate of complexity. The intent is to experiment with better complexity measures as this project continues. Figures 8, 9, and 10 show results from complexity measures taken of accumulated input and output of three separate components of the information system. The

graphs show the complexity of bit-level input and output strings concatenated together. That is, observe an input sequence at the bit-level and concatenate with an output sequence at the bit-level. This input/output concatenation is for either the entire system or for components of the system. If there is low complexity in the input/output observations, then it is likely to be easy for an attacker to understand that component of the system. Note that these graphs are showing estimates of Kolmogorov Complexity. If MML were used, the attacker's hypothesis would be used to determine the complexity relative to a particular attacker. In Figures 8, 9, and 10, the X-axis is the number of input and output observations concatenated to form a single string of bits. The particular complexity estimate used in this example is very poor; however, an ongoing area of research is to improve complexity estimates. Because of the inaccurate complexity metric, all the figures show a rising complexity with the number of accumulated observations. However, notice the rate at which the complexity rises in each of the figures. It would appear that Component E is most vulnerable due to its low rate of increase in complexity while Component B appears to be the least vulnerable due to its steeper rise in complexity. These results make intuitive sense because Component B is simply transmitting data without any form of protection while Component B is adding noise to the data. This vulnerability method does not take into account whether a component reduced or increased complexity; in other words whether the change was endothermic or exothermic complexity behavior. This project will extend this analysis with further work utilizing information distance as a vulnerability analysis technique.

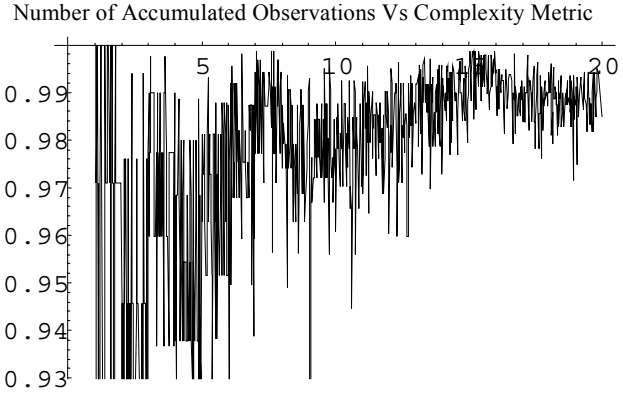

Figure 8. Complexity-Based vulnerability analysis of component C.

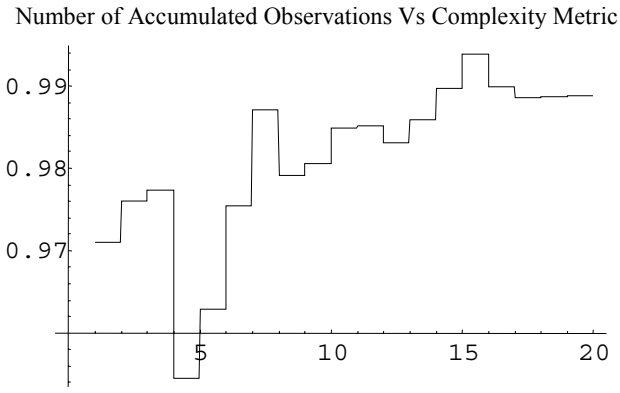

Figure 9. Complexity-Based Analysis of Component B.

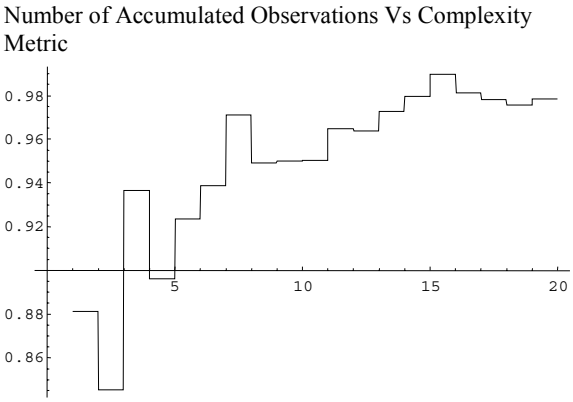

Figure 10. Complexity-Based Vulnerability Analysis of Component E.

These results show that vulnerabilities can be systemically discovered. When used in an MML approach to complexity measurement, apparent complexity, that is, complexity as seen by a particular attacker can be determined. Thus, this work has led towards automatic generation of vulnerabilities in graphical form similar to Figure , but without requiring expert knowledge of each type of vulnerability and in a more complete manner, depending upon the number of components analyzed. Note that all possible combination of components must be analyzed in this manner. Design of a general-purpose automated tool that can perform this type of vulnerability analysis is under construction.

8. Conclusions

Fundamental properties of information must be utilized to move the study of information assurance to a proactive versus reactive level. This work has identified the inherent property of Kolmogorov Complexity as a possible parameter for this task. Numerous opportunities exist for exploiting Kolmogorov Complexity to achieve security and network optimization. Two possible opportunities that were explored in this paper were utilization of Kolmogorov Complexity like metrics as conserved parameters to detect abnormal system behavior and Complexity-based vulnerability analysis. This paper has distinguished between data and process vulnerabilities, both of which can be systemically discovered without *a priori* knowledge of vulnerability types. When used in an MML approach to complexity measurement, apparent complexity, that is, complexity as seen by a particular attacker can be determined. This work leads towards automatic generation of vulnerabilities through multiple vulnerability chains without requiring expert knowledge of each type of vulnerability. Further research is necessary to develop suitable metrics and applications for these ideas to bear fruit.

9. Acknowledgements

This effort is funded by DARPA ISO contract number F33615-00-C-1629.

10. References

[1] Giancoli, Douglas C. General Physics, Prentice Hall, INC, Englewood Cliffs, NJ.
[2] Fraundorf, P. "Heat Capacity in Bits," April 28, 2000, Downloaded from URL http://www.umsl.edu/~fraundor/ifzx/cvinbits.html. An active revision of cond-mat/9711074 in the Los Alamos archives
[3] Cover, T. M. and Thomas, J. A. Elements of Information Theory. Wiley, NY, 1991.
[4] Li, Ming and Vitányi, Paul An Introduction to Kolmogorov Complexity and Its Applications, Springer, NY 1997
[5] Rolf Herken, ed. The Universal Turing Machine, A Half-Century Survey. Springer-Verlag, NY, 1995
[6] Denning, Elizabeth R, Cryptography and Data Security, Addison-Wesley, Mass, 1982.
[7] D. Eastlake, J. Schiller, S. Crocker "Randomness Requirements for Security", Internet draft, 30-Nov-00
[8] "Minimum Message Length and Kolmogorov Complexity," Wallace, C. S. and Dowe, D. L., The Computer Journal, Vol. 42, No 4. 1999.
[9] C. Bennett, P. Gacs, M. Li, P. Vitanyi, and W. Zurek, "Information Distance," IEEE Transactions on Information Theory, Vol 44, July, 1998.
[10] D. Denning, P. Denning, Internet Besieged, Addison Wesley, Mass, 1998.
[11] Bush, Stephen F. and Kulkarni, Amit B. Active Networks and Active Virtual Network Management Prediction: A Proactive Management Framework. ISBN 0-306-46560-4. Kluwer Academic/Plenum Publishers. Spring 2001.
[12] Bush, Stephen F. and Barnett, Bruce. *A Security Vulnerability Technique and Model*. GE Corporate Research and Development, January 1998. Technical Report 98CRD028.
[13] W. Kirchher, M. Li, and P. Vitanyi. *The Miraculous Universal Distribution*. The Mathematical Intelligencer, Springer-Verlag, New York, Vol. 19, No. 4, 1997.