INL/CON-05-00649
PREPRINT

Time-To-Compromise Model For Cyber Risk Reduction Estimation

Quality of Protection Workshop, ESORICS

Miles A. McQueen
Wayne F. Boyer
Mark A. Flynn
George A. Beitel

September 2005

The INL is a
U.S. Department of Energy
National Laboratory
operated by
Battelle Energy Alliance

Time-to-Compromise Model for Cyber Risk Reduction Estimation

Miles A. McQueen[1], Wayne F. Boyer[1], Mark A. Flynn[1], George A. Beitel[1],

[1] Idaho National Laboratory, 2525 N. Fremont,
Idaho Falls, Idaho, U.S.A. 83415
{miles.mcqueen, wayne.boyer, mark.flynn,
george.beitel}@inl.gov

Abstract. We propose a new model for estimating the time to compromise a system component that is visible to an attacker. The model provides an estimate of the expected value of the time-to-compromise as a function of known and visible vulnerabilities, and attacker skill level. The time-to-compromise random process model is a composite of three subprocesses associated with attacker actions aimed at the exploitation of vulnerabilities. In a case study, the model was used to aid in a risk reduction estimate between a baseline Supervisory Control and Data Acquisition (SCADA) system and the baseline system enhanced through a specific set of control system security remedial actions. For our case study, the total number of system vulnerabilities was reduced by 86% but the dominant attack path was through a component where the number of vulnerabilities was reduced by only 42% and the time-to-compromise of that component was increased by only 13% to 30% depending on attacker skill level.

1 Introduction

Control systems connected to public networks are at risk from cyber attack. Operators of these control systems need a measure of the risk associated with potential attacks to effectively manage their resources. Cyber security evaluations are traditionally qualitative in nature such that recommendations are given for remedial actions with no quantitative measure of how the recommended actions reduce the risk of a successful attack.

In April 2005 our risk analysis team was asked to perform a quantitative estimate of the risk reduction on a partial Supervisory Control and Data Acquisition (SCADA) system referred to as CS60. The baseline system had already undergone a security review, been modified to enhance security, and then been retested. For this analysis, we developed a methodology [13] for obtaining a quantitative risk reduction estimation. The methodology applies a graph theoretical approach. The methodology is briefly described by the following steps:

Step 1. Establish the system configuration.
Step 2. Identify applicable portions of the quantitative risk model.

Step 3. Identify and prioritize the security requirements of the primary target(s).
Step 4. Identify system component vulnerabilities.
Step 5. Categorize vulnerabilities on each component by compromise type.
Step 6. Estimate time-to-compromise each component.
Step 7. Generate compromise graph(s) and attack paths.
Step 8. Estimate dominant attack path(s).
Step 9. Do Steps 3–8 for baseline and enhanced system.
Step 10. Estimate risk reduction.

One could argue that all vulnerabilities should be fixed, e.g. by applying patches, thus the enhanced system should have no known vulnerabilities. We assert that a system with no known vulnerabilities continues to be at risk because of vulnerabilities that exist but are currently unknown, and we would like to measure that risk. Also, many real world systems operate with known vulnerabilities even after security upgrades. The crux of our methodology is the estimation of the time-to-compromise for each component in the system. Time-to-compromise is a measure of the effort expended by an attacker for a successful attack assuming effort is expended uniformly. We believe that as the time-to-compromise is increased, the likelihood of successful attack, and therefore risk, tends to decrease. The rest of this paper discusses the specific methods we used for step 6 of the methodology, estimating the time-to-compromise.

The estimation of time-to-compromise is particularly difficult because of the lack of reliable data. We recognize that some of the assumptions associated with our model have not been validated but we have attempted to provide justification with real data when data is available. We have used expert elicitation or have made simple assumptions when data is unavailable.

2 Related work

Researchers are testing the viability of different approaches for dealing with control system cyber security. Carlson et al. [8] describes a novel approach for applying Hidden Markov Models to an attack/defend scenario on an infrastructure system. The approach, based on sound statistical models, is flexible, but requires both detailed information about the system and significant set-up time. Madan et al. [11] apply a stochastic model to computer network system. It is used to determine steady-state availability of QoS attributes and also mean times to security failures based on probabilities of failure due to violations of different security attributes. The theory used is classic statistical stochastic modeling. Employing this type of model requires knowledge of the system in detail. Furthermore, Haimes [10] applied Hierarchical Holographic Models, event trees, and fault trees to a variety of applications, both models require specific details, are not dynamic, and rely on expert opinion.

Taylor et al. [15] provide an interesting cyber security assessment process that combines techniques from Survivability System Analysis and Probability Risk Assessment. The proposed process has some significant advantages, but seems more

suitable to complete and operational systems so that costs, attack scenarios, and critical system objectives may be fully explored. Further, the process is dependent on multiple iterations of expert elicitation, which are not available in many situations.

Dacier et al. [9] suggested the use of 'privilege graphs' to analyze security. Privilege graphs require modeling of vulnerabilities at a very low level, and, for a nontrivial sized system, would involve a graph of unmanageable size. Privilege graphs are transformed into Markov chains. But the assumptions underlying Markov chains are not necessarily applicable to an intelligent adversary.

Sheyner et al. [14] describe an automated technique for generating and analyzing attack graphs. They use a model checker as the core engine to comprehensively generate every attack path sequence that could lead to an undesired system state. There is a question of scalability in using a model checker to generate the attack paths, and the level of attack and vulnerability abstraction may be at a lower level than optimal for a quick estimate of risk reduction.

Byres et al. [7] describe how the attack tree methodology may be applied to the common SCADA protocol MODBUS/TCP with the goal of identifying security vulnerabilities inherent in the specification and in typical deployments. Attack trees are a promising technology but no method is provided for weighting the attack paths.

While a number of the above methods and techniques seem promising and merit future research, none could provide a quantitative measure of time-to-compromise that was necessary for our risk reduction estimation case study.

3. Estimate time-to-compromise

The time-to-compromise ($\mathbf{T_{pi}}$) is defined as the time needed for an attacker to gain some level of privilege p on some system component i. $\mathbf{T_{pi}}$ depends on the nature of the vulnerabilities and the attacker skill level. $\mathbf{T_{pi}}$ is modeled as a random process composed of the following three attacker subprocesses:

- **Process 1** is for the case where at least one vulnerability is known on component i that would achieve privilege level p, and the attacker has at least one exploit readily available that can be successfully used against one of the known vulnerabilities.
- **Process 2** is for the case where at least one vulnerability is known on component i that would achieve privilege level p, but the attacker does not have an exploit readily available that can be successfully used against any of the known vulnerabilities.
- **Process 3** is the identification of new vulnerabilities and exploits. Process 3 is a parallel process to processes 1 and 2, and is constantly running in the background. The attacker of a particular system may use the results of process 3 or may be an active participant in process 3. That is, the attacker may wait for new vulnerabilities/exploits to be identified and announced, or probe for new ones.

Each of the above processes has a different failure probability distribution. Process 1 and 2 are mutually exclusive and Process 3 is ongoing and in parallel with the other two processes.

3.1. Process 1 model

The Process 1 activities are shown in flow chart form in Figure 1. Notice that Process 1 always has a successful completion. The Process 1 model has two parts: 1) the probability estimate that the attacker has an exploit readily available to use against one of the component's vulnerabilities, that is, the probability the attacker is in Process 1, and 2) the time estimation for Process 1.

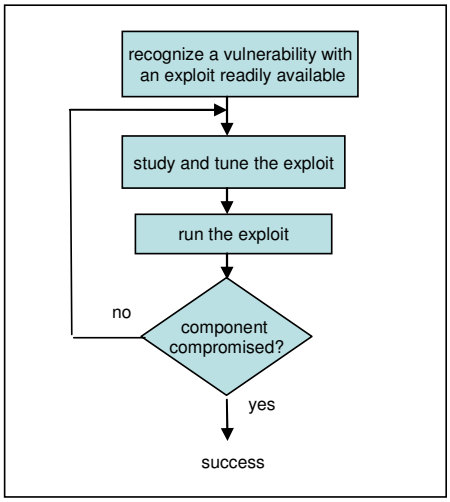

Fig. 1. Process 1

3.1.1. Probability the attacker is in process 1

The probability that the attacker is in Process 1 is calculated by using search theory in a similar fashion as has been applied to physical security systems by Major [12]. The following equation makes use of the simplifying assumption that the available exploits are uniformly distributed over all vulnerabilities:

$$P_1 = 1 - e^{-vm/k}. \quad (1)$$

where P_1 is probability that the attacker has an exploit readily available that will compromise the component, v is number of vulnerabilities on the component of interest, m is the number of exploits readily available to the attacker, and k is the total number of vulnerabilities. The value of k is 9447 and is defined to be the total number of nonduplicate-known vulnerabilities found in the ICAT database.

Table 1. Model parameters—number of readily available exploits by skill level.

Skill Level	*m (number of readily available exploits)*
novice	50
beginner	150
intermediate	250
expert	450

The value of m is a function of the attacker skill level. The novice skill level is defined as m = 50 because there is a Web site (metasploit) that has 50 exploits that are

trivial to use. The higher skill levels are defined by increasing the value of m for each increase in skill level as shown in Table 1. The specific choices in Table 1 are based on a postulated exponential growth in readily available exploits as a function of skill level.

3.1.2. Time estimation for process 1

The probability density function (PDF) for Process 1 is expected to be zero at time zero, rise rapidly then decrease to zero for times greater than some maximum time value. The shape of the PDF for this process is anticipated to look something like the beta distribution [3] shown in Figure 2.

The mean time for Process 1 was estimated as follows. Process 1 assumes that the attacker is familiar with at least one of the available vulnerabilities and has experience with at least one exploit to take advantage of the known vulnerabilities. Currently, the time it takes for an expert or novice to compromise a component under these conditions is considered to be roughly similar. Thus, the mean time-to-compromise for Process 1 is not modified based on skill or any other external factors. Cohen [2] states: "It takes a few days to program a few new attacks into systems, test them out, and prepare for a serious attack if you are already in the business of attacking other people." This suggests that the mean should be a few days. However, experiments conducted by Jonsson [4] suggest that a team of two nonprofessional attackers can execute a compromise in approximately 4 hours, on average. Based on the specification of time used by Jonsson, the 4 hours could represent the total time used for the attack or the time devoted by each team member, for a possible total of 8 hours. Somewhat arbitrarily, we decided to use 8 hours (one working day) as the mean time for a successful attack in Process 1, since it is at least marginally more in line with Cohen's comment.

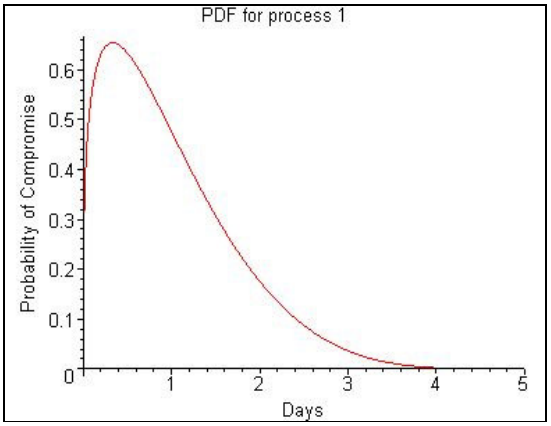

Fig. 2. PDF for Process 1. Time-to-compromise a component for the case where the attacker has an exploit readily available. This PDF is a 'beta' distribution with shape parameters == 1.3, 5.2; range == 0..5; mean = 1:

3.2. Process 2 Model

The Process 2 activities are shown in flow chart form in Figure 3. Notice Process 2 can have multiple tries and may end in failure or success. The Process 2 model has

two parts: 1) the probability estimate that the attacker is in Process 2, and 2) the time estimation for Process 2.

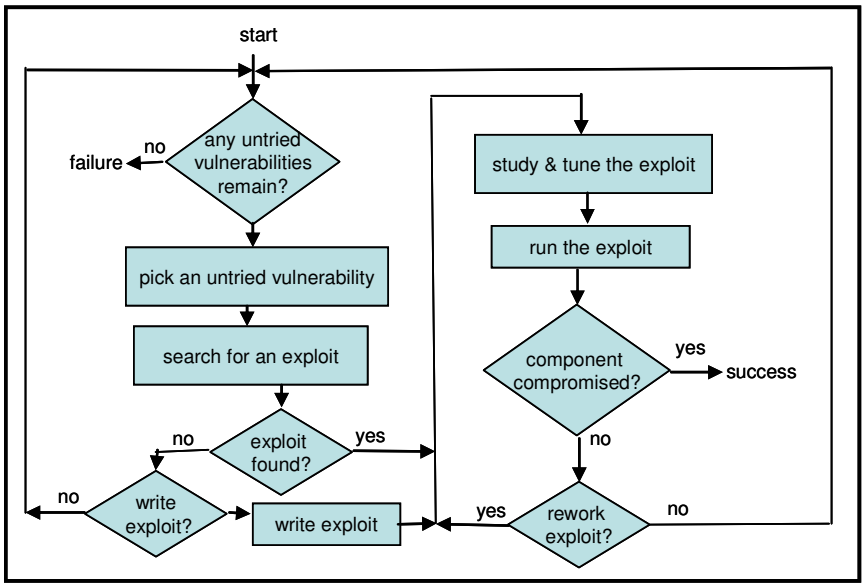

Fig. 3.. Process 2

3.2.1. Probability the attacker is in process 2

Since Process 1 and Process 2 are mutually exclusive, when $v > 0$ the probability that the attacker is in Process 2 is the complement of the probability that Process 1 applies. That is

$$P_2 = e^{-vm/k} = 1 - P_1. \quad (2)$$

where P_2 is the probability that the attacker does not have an exploit readily available that will compromise the component and P_1 is from Equation 1.

3.2.2. Time estimation for process 2

The PDF of Process 2 is expected to look similar to the gamma distribution [3] shown in Figure 4. A gamma distribution was chosen because the PDF is zero at time zero, and as time increases it peaks and then trends towards but never reaches zero. We chose a PDF that is non-zero at infinity because Process 2 has no guarantee of successful completion within any given time. The PDF in Figure 4 is a baseline PDF for process 2 and represents the case where the attacker is expected to find or write a usable exploit for the first vulnerability they try to exploit. As the expected number of

vulnerabilities that an attacker must try to exploit before being successful increases, the PDF will be modified according to the number of "tries" needed as explained below. The average value of the baseline PDF for Process 2 was chosen as the average time from vulnerability announcement to exploit code availability, which according to [6] is 5.8 days.

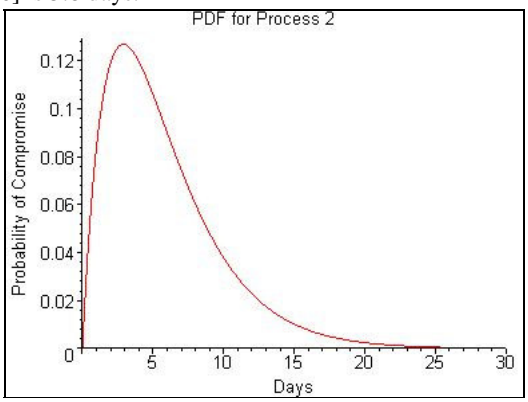

Fig. 4. Baseline PDF for Process 2. Time-to-compromise for the case of at least one vulnerability but the attacker has no exploit readily available. The PDF is a gamma distribution. Shape parameters == 2, 2.9; Mean == 5.8.

The mean time estimation for Process 2 should be dependent on the number of known vulnerabilities and the probability the attacker will be able to find or write their own exploit code to take advantage of the weakness. This was modeled as a serial process in which the attacker randomly selects one of the known vulnerabilities and then tries to find or create an exploit for it. The average time it takes for each of these tries is considered constant and is the baseline mean of the hypothesized gamma PDF (5.8 days). The mean time of Process 2 is modeled as the expected value of the number of tries times 5.8 days; that is:

$$t_2 = 5.8\ ET. \tag{3}$$

where t_2 is expected value of Process 2 and *ET* is the expected number of tries.

The expected number of tries may be written as:

$$ET = \frac{AM}{V} * \left(1 + \sum_{tries=2}^{V-AM+1} \left[tries * \prod_{i=2}^{tries} \left(\frac{NM - i + 2}{V - i + 1} \right) \right] \right) \tag{4}$$

where AM is the average number of the vulnerabilities for which an exploit can be found or created by the attacker given their skill level, NM is the number of vulnerabilities that this skill level of attacker won't be able to use (V-AM), and V is the number of vulnerabilities on the component of interest. See appendix for derivation of equation 4.

Equation 4 was obtained by assuming each try is an independent sample from the list of vulnerabilities where the unusable vulnerabilities are randomly distributed among the list.

Table 2. Fraction of vulnerabilities exploitable by attacker as a function of skill level.

Skill Level	*Fraction of Vulnerabilities that are Exploitable (AM/V)*	*20 vulnerabilities from ICAT database judged by expert to be exploitable by this and lower levels.*
novice	.15	CAN-2003-0004 CAN-2001-1039 CAN-2002-1048
beginner	.30	CAN-2004-1306 CAN-1999-1457 CVE-2000-0359
intermediate	.55	CAN-2004-0893 CAN-2005-0416 CAN-2002-0053 CAN-2003-0345 CAN-2004-0206
expert	1.00	CAN-2003-0897 CAN-2004-0117 CAN-2004-0208 CAN-2004-0575 CAN-2003-0724 CAN-2004-0118 CAN-2004-0119 CAN-2004-0123 CAN-2004-0897

To determine the values of AM/V as a function of skill level, we sampled 20 vulnerabilities to assess the availability of corresponding exploits. See table 2 for list of vulnerabilities. If our team expert assessed the vulnerability as requiring no code, or the code was available and trivial, it was then assumed a novice attacker could make use of the vulnerability/exploit pair. This criteria was met by three (15%) of the 20 vulnerabilities that were assessed. If our team expert assessed the vulnerability as being available to the novice or that it required exploit code that was readily accessible (and appeared easy to understand), it was assumed a beginner attacker could execute the code and take advantage of the vulnerability. Three additional vulnerability/exploit pairs also met this criteria, bringing the total to six pairs (30%) available to the beginner attacker. If, in addition to the above six vulnerability/exploit pairs, our team expert found the exploit code to be

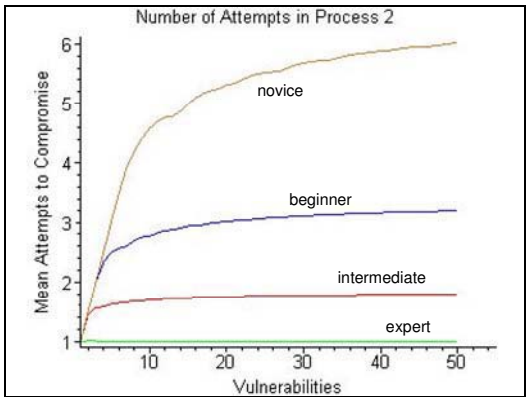

Fig. 5. Average number of attempts to compromise a component for Process 2 as a function of number of known component vulnerabilities and the attacker skill level. (Equation 4)

difficult to understand, found only example code conveying the essence of the exploit, or assessed from experience that it was not easy to get the type of required exploit to work properly, it was assumed it would require an intermediate level attacker to take advantage of the vulnerability/exploit pair. Five additional pairs met this criteria so that 11 pairs (55%) were available to the intermediate attacker. If, in addition to the 11 exploits above, no exploit code was readily available or what code there was required significant modification to adapt, we assumed that it would require an expert attacker. All of the remaining vulnerabilities fit into this category for a total of 20 vulnerability/exploit pairs (100%) available to the expert. The results of this exercise are summarized in Table 2 and were used as part of the time-to-compromise model.

The average number of tries as described by Equation 4 is plotted in Figure 5. For an expert, the average number of tries is one, because an expert is expected to have access to an exploit for every vulnerability. As the skill level decreases, the average number of tries increases as expected.

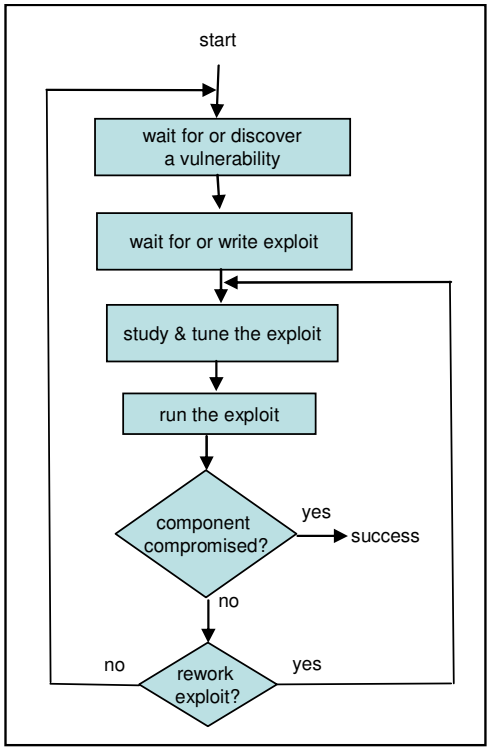

Fig. 6. Process 3

3.3. Time estimation for process 3

The Process 3 activities are shown in flow chart form in Figure 6. Notice Process 3 continues until "success". The time to the discovery of a new usable vulnerability is modeled as a constant rate of new vulnerabilities/exploits occurring on a component. This model is the same form as the classic exponential distribution for constant failure rates as shown in Figure 7.

This exponential distribution was chosen for its simplicity and because research by Rescorla [5] indicates that the hypothesis stating that the vulnerability discovery rate is constant over time could not be rejected for the operating systems he studied. Using data from the same source, it also appears that a reasonable estimate for the mean time between vulnerabilities (MTBV) is 30.42 days. In addition to a vulnerability, an exploit would be needed. The time between the announcement of a vulnerability and the release of a corresponding exploit is now approximately 5.8 days [6]. The vulnerability rate esti-

mate will be scaled by V/AM according to the portion of vulnerability/exploit pairs each attacker level can use (see Table 2). For example, the beginner attacker will on average require 1/(0.3) vulnerability/exploit pair occurrences before one becomes usable. To determine the expected time-to-compromise for Process 3, the MTBV was multiplied by the scaling factor then half the MTBV was subtracted because, on the average, the midpoint of the fault cycle is the start point, and the mean time to create an exploit (5.8 days) is added. Thus:

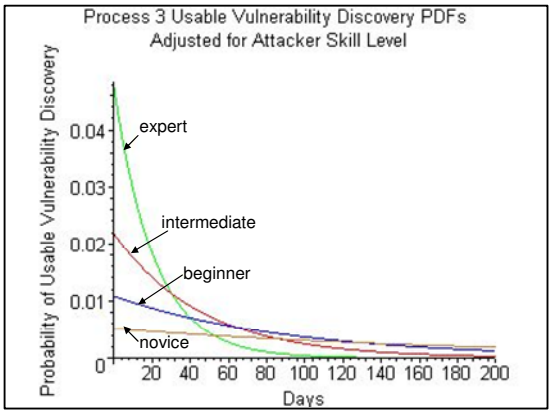

Fig. 7. PDF for probability of discovery of a new usable vulnerability. (Process 3).

$$t_3 = ((V/AM) - 0.5)\ 30.42 + 5.8 \quad (\mathbf{5})$$

where t_3 is the expected time-to-compromise of Process 3, and V/AM is the appropriate value from Table 2.

One might assume that development and release of patches might be effective in mitigating the window of opportunity for an attacker, but as indicated by [1] the hypothesis of 'poor system administration' seems to be confirmed. In other words, it takes quite a long time for administrators to actually apply patches, and although there are some indications that the time between release of a patch and its application may be decreasing in the IT domain, control systems may be slower.

3.4 Overall compromise time estimation

Given the three attacker processes for compromising a component, their probabilities, and their time-to-compromise probability distributions we can now generate an overall time-to-compromise probability distribution for the component. For now, the analysis only uses the expected value of the time-to-compromise. The expected value of the overall distribution is approximated as a weighted sum of the expected values of each of the three attacker processes, where each weight is the probability that the respective process is operative.

Although Process 3 is a parallel process continually running in the background, we simplify the estimation of time-to-compromise and obtain a good first order approximation by assuming that Process 3 only applies if Processes 1 and 2 do not apply or are unsuccessful. This approximation is valid because the PDF for Process 3 is much more dispersed than the other processes, therefore its contribution to the composite PDF is small when Processes 1 or 2 are active. The following formula is valid under the assumption that all three processes are approximately mutually exclusive.

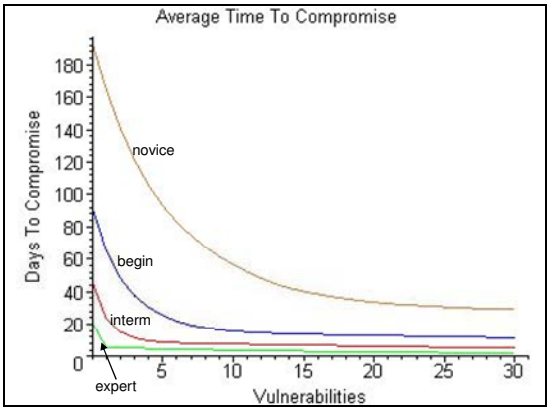

Fig. 8. Expected time-to-compromise a component for various numbers of vulnerabilities and attacker skill levels. (Equation 6 plotted)

$$T = t_1 P_1 + t_2 (1-P_1)(1-u) + t_3 u(1-P_1). \quad (6)$$

where

T is the expected value of time-to-compromise
t_1 is the expected value of Process 1 (1 day)
t_2 is the expected value of Process 2 (from Equation 3)
t_3 is the expected value of process 3 (from Equation 5)
$u = (1 - (AM/V))^v \equiv$ probability that Process 2 is unsuccessful (u=1 if V=0)
V is number of vulnerabilities, P_1 from Equation 1, AM/V from Table 2.

Equation 6 is plotted in Figure 8 where the number of known vulnerabilities for a component range from zero to 30 and attacker skill levels range from novice to expert. The time-to-compromise the component increases as the skill level decreases. Time-to-compromise decreases as the number of vulnerabilities increases, but for skilled attackers the time-to-compromise is not a strong function of the number of vulnerabilities. The shape of the curves shown in Figure 8 is consistent with intuition, although the numerical values are only approximations.

4. Case study

Our risk reduction methodology was applied to a small SCADA system (CS60) consisting of 8 generic component types connected to a local Ethernet LAN. The only potential attack target component identified was the RTU because it controls the physical state of equipment in the field. The system was tested as delivered from the

manufacturer and did not include a firewall. The only perimeter component for the CS60 system is the Ethernet switch that connects the system to the internet. For the purposes of testing, this perimeter component was assumed to be a simple switch that prevents locally addressed packets from external observation and prevents flooding of the local network from external sources.

Both the baseline and enhanced, more secure, versions of the CS60 system were tested with a variety of commercial and freeware scanning tools (including Nessus), password crackers and local tools. Tests of the baseline system revealed potential vulnerabilities in every component. The network scans found a total of 298 open ports, and 79 unknown services. Nessus vulnerability scans found 174 warnings and 154 holes. A 'hole' is a vulnerability that has the potential to allow an attacker to gain a foothold on the component. We were only concerned with holes that were noted by Nessus as high severity to increase confidence that they were significant vulnerabilities. High severity implies that the hole might allow one to run 'arbitrary code' on the component to gain user or root access. The password testing found weak passwords on virtually all of the componentss. Network scans of the enhanced system found 95 open ports and nine unknown services. Nessus vulnerability scans of the enhanced system found a total of 101 warnings and 21 holes. Some of the vulnerabilities identified by the tools were validated for the enhanced system. INL expertise was used for the identification and validation of additional vulnerabilities. The passwords in the enhanced system were found to be much stronger than for the baseline system. We identified additional potential vulnerabilities by searching vulnerability identification libraries.

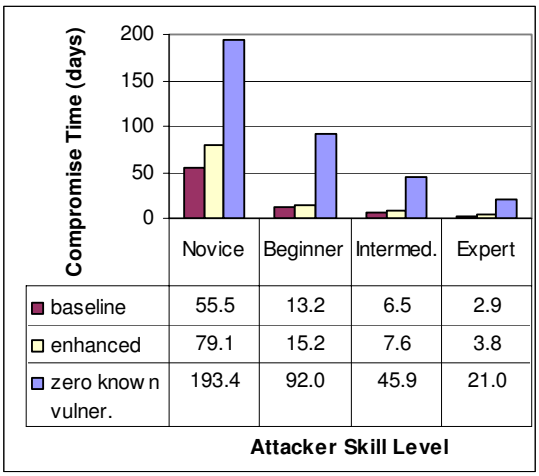

Fig. 9. Estimated compromise time of the most vulnerable component of the CS60 baseline system, enhanced system, and for hypothetical case of no known vulnerabilities.

The time-to-compromise was calculated using Equation 6 and the number of vulnerabilities we identified that could be used to gain root access for each component in the CS60 system. Component APPS1 had the highest number of vulnerabilities in the baseline system (19) and in the enhanced system (11) for the type compromise that allows root access from a launch site, therefore the path through component APPS1 is considered to be the dominant attack path. The time to compromise APPS1 for various attacker skill levels is shown in Figure 9 for the baseline system, the enhanced system and for the hypothetical case of no known vulnerabilities. The total number of

CS60 system vulnerabilities was reduced by 86% but the number of vulnerabilities for the component APPS1 was reduced by only 42% and the time-to-compromise APPS1 was increased by only 13% to 30% depending on attacker skill level. For the hypothetical case of 100% reduction in known vulnerabilities, the time-to-compromise is estimated to increase by 240% to 624% depending on attacker skill level.

These estimates of time-to-compromise have not been validated but simply show how the model may be applied to a real system. The numbers can be interpreted as a measure of risk and therefore may be used to trade off the value of cyber security mitigation actions versus the cost.

5. Alternative simplistic time-to-compromise models/metrics

Consider some simplistic alternative time-to-compromise models/metrics. One such model is the binary open/closed door model in which any known vulnerability is considered an open door that a determined attacker will enter as easily as if there were many other open doors. The application of this model to the case study yields a time-to-compromise reduction of zero on the APPS1 component because there are known vulnerabilities (open doors) remaining that lead to a successful attack, even though many doors have been closed. This model has some merit, particularly if the attacker is highly skilled and is determined to attack that particular site, but is considered too pessimistic and too simplistic because it does not take into account the various types of potential attackers, and the difficulty associated with the attacker in exploiting different sets of vulnerabilities is not considered.

Another alternative time-to-compromise metric for components may be obtained by counting the reduction in number of vulnerabilities. This can be done in several ways. For example: the total number of vulnerabilities for each component (before and after system enhancements) may be counted. An alternative view of vulnerabilities is the number of open TCP services rather than CVE entries. For this case study, the total holes found by Nessus (http://www.nessus.org) was reduced from 154 to 21 (86%), the number of vulnerabilities on the most vulnerable component was reduced from 19 to 11 (42%), and the total number of open TCP services was reduced from 298 to 95 (68%). This model is also believed to be too simplistic and too optimistic because it implies a linear relationship between number of vulnerabilities and the time-to-compromise a component, and ignores other important considerations such as skill of attacker.

6. Conclusions

We proposed a model for estimating the time-to-compromise a system component that is visible to an attacker. The model can be used as part of a risk reduction estimation methodology for control systems. Time-to-compromise was modeled as a function of known vulnerabilities and attacker skill level and was applied to a specific SCADA system and for a specific set of control system security remedial measures.

The nature of the numerical results obtained show that time-to-compromise is related to system attributes in ways consistent with intuition, and reinforces the types of remedial actions that truly reduce risk. For example, the model emphasizes the dynamic nature of cyber security such that the time-to-compromise a component decreases over time, unless there is constant effort to install patches or disable services as soon as new vulnerabilities are discovered.

The model also suggests some new strategies for reducing the risk of cyber threats: the publication of false exploits or government restrictions on the publication of valid exploits could theoretically increase the time necessary for an attacker to compromise components. Software that spoofs vulnerability scanning tools could trick potential attackers into trying exploits that would not be successful, but would raise alarms.

The time-to-compromise model has the following drawbacks. The model does not currently take into account dependencies between vulnerabilities on different components. For example, if two components are not identical but have some of the same vulnerabilities, compromising them are not independent events. Whether the number of available exploits is representative of the skill level of an attacker and estimates of the number of exploits available to various skill levels were not validated. The assumption that exploits are uniformly distributed over vulnerabilities is incorrect. It is our hypothesis that certain exploits are far more likely to be in the hands of an attacker, since the vulnerability is found on many more systems. The PDFs were not validated for Processes 1 and 2.

The proposed model for estimating time-to-compromise provides a quantitative assessment mechanism that fits within an overall methodology of risk assessment. The level of abstraction is high enough to avoid detailed analysis of each known vulnerability but detailed enough to provide useful security and defensive information for guiding risk assessments and mitigation strategies.

7. Future Work

The time-to-compromise model needs to be validated through experiments and measurement where possible. We plan to run realistic tests to collect information about attacker processes. We would like to perform a sensitivity analysis to determine how sensitive the model is to changes in our underlying assumptions.

The kind of data needed to effectively estimate control system cyber security risk is currently lacking. For example: the industry needs a vulnerability library specific to control systems similar to the existing IT CVE vulnerability library. The existing CVE libraries do not always clearly identify where vulnerabilities apply, nor do they indicate how difficult it is to exploit a given vulnerability. Existing vulnerability scanning tools do not clearly identify which vulnerabilities are tested and which are not. We would like to run experiments that measure the statistics associated with Processes 1 and 2. Validated statistical models may allow for a measure of the error bounds associated with future time-to-compromise estimates.

When dealing with a system many components will have common vulnerabilities. A method should be developed to account for such dependencies. Also, many com-

ponents may be of equal use to an attacker. In such a case it may be more appropriate to aggregate the appropriate components into a higher level meta-component with the union of all its components vulnerabilities. This needs to be assessed.

References

1. Browne, H. K., McHugh, J., Arbaugh, W.A. and Fithen, W.L., "A trend Analysis of Exploitations," technical report CS-TR-4200, University of Maryland and Software Engineering Institute, November 2002.
2. Cohen, F., "Managing Network Security The Millisecond Fantasy," http://all.net/journal/netsec/1999–2003.html, 2003.
3. Evans, M., Hastings, N. and Peacock, B., "Statistical Distributions," Second Edition, 1993.
4. Jonsson, E., "A Quantitative Model of the Security Intrusion Process Based on Attacker Behavior," IEEE Transactions on Software Engineering, Vol 23 No 4, April 1997.
5. Rescorla, E., "Is Finding Security Holes a Good Idea," IEEE Security & Privacy, January-February 2005.
6. Turner, D., ed., "Symantec Internet Security Threat Report," Volume VI, September, 2004, http://enterprisesecurity.symantec.com/content.cfm?articleid=1539, 2004.
7. Byres, E. J., Franz, M. and Miller, D., "The Use of Attack Trees in Assessing Vulnerabilities in SCADA Systems", International Infrastructure Survivability Workshop (IISW '04), IEEE, Lisbon, Portugal, December 4, 2004
8. Carlson, R. E., Turnquist, M. A. and Nozick, L. K., Expected Losses, Insurability, and Benefits from Reducing Vulnerability to Attacks, SAND2004-0742, Sandia National Laboratories, Albuquerque, New Mexico, 2004.
9. Dacier, M., Deswarte, Y. and Kaaniche, M., "Quantitative Assessment of Operational Security: Models and Tools" Information Systems Security, ed. by S. K. Katsikas and D. Gritzalis, London, Chapman & Hall, p.179-86, 1996.
10. Haimes, Yacov Y., "Accident Precursors, Terrorist Attacks, and Systems Engineering," Presented at the NAE Workshop, 2003.
11. Madan, B. B., Goševa-Popstojavova, K., Vaidyanathan, K. and Trivedi, K. S., "Modeling and Quantification of Security Attributes of Software Systems," International Conference on Dependable Systems and Networks, Washington, DC,, 2002.
12. Major, J. A., "Advanced Techniques for Modeling Terrorism Risk," Journal of Risk Finance, Fall 2002.
13. McQueen, M. A., Boyer, W. F., Flynn, M. A. and Beitel, G. A., "Quantitative Cyber Risk Reduction Estimation for a SCADA Control System", INL/EXT-05-00319, Idaho National Laboratory, CSSC Report, prepared for U.S. Department of Homeland Security, May 17, 2005.
14. Sheyner, O., Haines, J., Jha, S., Lippmann, R. and Wing, J. M., "Automated Generation and Analysis of Attack Graphs," Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy, Berkeley, California, May 2002, 273–284.
15. Taylor C., Krings, A. and Alves-Foss, J., "Risk Analysis and Probabilistic Survivability Assessment (RAPSA): An Assessment Approach for Power Substation Hardening," Proc. ACM Workshop on Scientific Aspects of Cyber Terrorism, (SACT), Washington DC, November 21, 2002.

Appendix A

Derivation of Equation 4.

E(X) is expected value of **X** where **X** is a discrete random variable:

$$E(X) = \sum_{k=1}^{V} x_k * p_k$$

where x_k are the possible values of **X** (outcomes) and p_k is the probability of outcome k.

$$1 = \sum_{k=1}^{V} p_k$$

p_1 = Probability of matching an available exploit to the first vulnerability chosen.

p_1 = AM/V, because of uniform distribution of exploits over vulnerabilities. AM is number of usable exploits available, V is number of vulnerabilities.

p_2 = Probability of matching an available exploit to the 2nd vulnerability chosen.

p_2 = (probability of matching an available exploit to a vulnerability chosen from those remaining after first try)*(probability exploit not matched on the first try)

p_2 = (AM/(V-1))*((V-AM)/V)

p_2 = (AM/V) * (V-AM)/(V-1)

p_3 = (AM/(V-2))*(probability exploit not matched on the first two tries)

p_k = (AM/(V-k+1))*(probability exploit not matched on the first k-1 tries)

$$p_k = (AM/V) * \left(\prod_{i=2}^{k} \left[\frac{NM - i + 2}{V - i + 1} \right] \right) \; ; 2 \le k \le V\text{-}AM+1$$

$p_k = 0$; $k > V\text{-}AM+1$ (because there are AM usable exploits available, therefore there are no **untried** vulnerabilities with exploits available to the attacker for these cases. Attacker is successful for some previous value of k.)

$$ET = E(X) = \sum_{k=1}^{V-AM+1} k * p_k$$

$$ET = \frac{AM}{V} * \left(1 + \sum_{tries=2}^{V-AM+1} \left[tries * \prod_{i=2}^{tries} \left(\frac{NM - i + 2}{V - i + 1} \right) \right] \right)$$