cyberpoint

CyberV@R

A Cyber Security Model for Value at Risk

Dr. Mark Raugas, Dr. James Ulrich, Roberta Faux, Scott Finkelstein, and Charlie Cabot

January 7, 2013

CPI.WP.CVAR.002

Terms of Use: Cyber Point International, LLC is committed to benefiting the security community in its efforts to address cyber theft, cyber terrorism and other security issues. Information in this white paper is being provided to security professionals and other readers as a resource for their consideration with respect to analyzing and preventing cyber threats and security vulnerabilities, and not for any malicious or unauthorized use.

The data and information in this white paper are provided “as is,” and Cyber Point International, LLC does not represent or warrant that the information is reliable or error-free. Cyber Point International, LLC shall not be liable for any claims, expenses, costs, losses or damages (whether in contract, tort, or otherwise) arising out of or relating to the use of the information. No use of this information in contravention of applicable laws is permitted. Export or re-import of the information to any country may require export license.

Abstract

We present a model, based conceptually on computational techniques employed in the financial realm, for quantifying an institution's monetary value at risk (VaR) due to cyber threats to which that institution is exposed, and identifying the optimal countermeasures given a fixed maximum level of expenditure. We call the model "CyberPoint CyberV@R." The model accepts as input attack trees, constructed to model the exposure of asset classes of known monetary value to specific forms of cyber attack. Using Bayesian probabilistic graphical models, the trees are combined into a multivariate probability density function for overall loss exposure, parameterized by threat levels and countermeasure effectiveness. The trees may be modified to reflect the proposed adoption of specific countermeasures, yielding a change in the overall density function. By evaluating the corresponding reductions in loss exposure over a cost-constrained space of countermeasures, a locally optimal countermeasure posture is obtained. We describe the theoretical underpinnings of the model and provide its mathematical formulation, consider its dynamical aspects, and provide pseudo-code for key model algorithms, portions of which are implemented in the "libpgm" open-source Python package provided by CyberPoint Labs [libpgm]. Real-world issues attending commercial implementations are considered, and model extensions (including automated attack tree construction) are discussed.

Contents

CyberV@R: A Cybersecurity Model for Value at Risk

1. Introduction

The ever-expanding threat of cyber attack presents IT administrators and CIOs with the daunting challenge of safeguarding their institutions' cyber infrastructure from breaches that could lead to catastrophic economic loss [Brenner2011], [Clarke2010], [EOPOTUS]. Yet security resources remain finite, and their wise allocation requires a method of accurately quantifying potential economic loss due to cyber attack, and the comparative economic benefit of alternative scenarios for resource allocation, to implement attack countermeasures.

There is a long-standing practice among financial institutions of quantitatively assessing monetary risk, through a variety of models having as a common stated objective the calculation of the institutional "Value at Risk" (VaR) [JPM1996], [Hull2000]. The techniques enable a chief risk offer to issue statements of the form "we deem it $X\%$ likely that over the next N business days, our institution will lose no more than D dollars." (Hence if N were a time window of one business day, and X was set at 99%, then one would expect the next business day's losses to exceed amount D on no more than 2 or 3 business days a year.) Financial regulatory agencies issue guidelines on the time horizons and likelihood benchmarks that financial institutions must use when computing VaR [SEC], and hence the VaR models afford (in theory) a measure of uniformity in the management of, or at least in the statement of, risk levels recognized by the institutions [Jor2007].

At least one previous publication examines the transfer of formalized risk frameworks from finance to the cyber realm [Huw2008]. We present in this paper an adaption of VaR techniques, borrowed from and motivated by the financial industry, that will provide to chief information officers and information technology (IT) managers a means of performing uniform quantitative assessment of cyber risk. These officers are concerned not with the performance of investment portfolios under the vagaries of financial markets, but rather the exposure of key assets (particularly intellectual property) or liabilities (such as those involved in breaches of customer data) to the vagaries of the cyber environment in which a firm's computing infrastructure is operated. Hence, we describe a model to help these organizational representatives answer the question"within a certain level of confidence, what is the most amount of money we could lose in the next N days due to various forms of cyber attack to which we are exposed?" An important follow-on question is of course "and what can we do to decrease the monetary value at risk?" and we present mechanisms for answering this question as well. We stress that the model we present is *not* designed to assign a static "risk score" to an organization, as some commercially available products provide. These scores indeed may be harnessed as inputs to our model. The goal of our model, by contrast, is to present a time-dependent valuation of the assets at risk due to cyber attack, *given* an organization's current IT security posture. More precisely, our model will accept:

- the values of a set of intellectual property assets,
- the arrangement of the assets within an information technology infrastructure,
- the risks to which the infrastructure is exposed,
- the steps taken to mitigate the risk,
- a time horizon T,
- a probability P,

and compute a value V, such that the odds over the next T time units of losing more than V monetary units' worth of assets to cyber-related attacks on the infrastructure are less than $1 - P$.

To specify a coherent model, we must therefore provide mechanisms for describing an information technology structure, the nature and incidence rates of the cyber threats to which the infrastructure is exposed, and the nature and efficacy of the mitigations in place to lesson the severity of the threats, as well as the procedure for calculating the "cyber value at risk" itself (which we hereafter dub CyberV@R). We will also want to provide guidance on possible sources an organization might consult to establish the specific inputs for a given model instantiation, particularly with regard to threat incident rates and mitigation effectiveness. The bulk of this paper will be concerned with these issues in some detail. In the remaining subsections of this introduction, we provide a sketch of our approach and its motivations.

Note: in what follows, various 3rd-party open-source and commercial software, systems, and repositories are referenced. These references are made for illustrative purposes only and are not intended to imply any endorsement on the part of the 3rd-party entities.

1.1. **Conceptual motivation: borrowing from the world of finance.** As described in standard texts such as [Hull2000], the canonical value at risk model involves a portfolio of stocks; we provide here an example which for simplicity assumes a portfolio holding U.S. \$10,000 in shares of company A and U.S. \$20,000 in shares of company B. Assume that based on historical data, the daily volatility of A's stock price is 5%, and the daily volatility of B's price is 10%. One traditionally assumes that fluctuations in the price of a stock, over a fixed time horizon of T days, is modeled by a normal distribution having mean 0 and standard deviation $\sigma\sqrt{T}$, where σ is the daily volatility of the stock. So the T-day standard deviation for our A holding is given by:

$$\sigma_A = 10,000 \times 0.05 \times \sqrt{T}$$

and similarly the standard deviation for B is given by:

$$\sigma_B = 20,000 \times 0.10 \times \sqrt{T}.$$

Assume moreover that based on historical data, we know the coefficient of correlation ρ between changes in stock prices of the two companies. Then the T-day distribution for the change in value Δp of our portfolio p is given by a normal distribution having mean 0 and standard deviation:

$$\sigma_{AB} = \sqrt{\sigma_A^2 + \sigma_B^2 + 2\rho\sigma_A\sigma_B}.$$

Using this information, one can find (from a statistical table or spreadsheet program) the value X such that $P(\Delta p < X) = 0.02$, that is:

$$P(\Delta p \geq X) = \frac{1}{\sigma_{AB}\sqrt{2\pi}} \int_{x=X}^{x=\infty} e^{-x^2/2\sigma_{AB}} dx = 0.98. \tag{1}$$

We say that X is our 98% VaR. For $T = 10$ and $\rho = 0.75$, $X \approx -\$6382.00$.

In our CyberV@R model, we will want to perform similar calculations over distributions of possible losses of intellectual property (or incurring of liabilities) over time, due to various forms of cyber attack on computing infrastructure. But whereas in our simple stock portfolio example, the risk drivers are the volatilities of the stocks in the portfolio, in the case of CyberV@R, the risk drivers will be the various types and incidence rates of cyber threats, and the attack surfaces the infrastructure presents. Moreover, whereas in our simple portfolio example, a mitigation is implicitly present in the correlation coefficient ρ, in that anti-correlated stocks will buffer the effects of relative price movements, in the cyber case, the mitigations will take the form of explicit security measures. Finally, whereas in our stock example, the portfolio value is assumed to be normally distributed, in the cyber case, the distributions will be more complex, and we will model them using dynamic Bayesian networks [Kol2009], which are formed from unions of attack trees [Schneier99], [Roy2010], [Salter98], representing the interactions of threats, infrastructure, defenses, and defended assets.

There is some precedent for employing Bayesian networks to perform risk analysis and quantify countermeasure effectiveness [Pol2012]. However, closed form analytic solutions to computation will generally not be possible (or convenient to compute, in any case), and instead we will resort to Monte Carlo-based simulation techniques such as those described in [Glas2004]. We next sketch our methodology for characterizing cyber threats and their mitigations.

1.2. **Characterization of threat types.** Many good sources of documentation exist from which one may obtain a sense of the general flora and fauna of cyber threats, including materials provided by the American National Standards Insttitue [ANSI2010], [ANSI2012], the U.S. Department of Homeland Security [DHS2011] [DSH2009], the U.S. Department of Energy [DOE2012] , private insurers [LLOYDS2011], and private authors [Jakob2008]. More specific information regarding incidence rates for certain known threat types may be found in publications by federal entities such as the National Institute of Standard and Technology [NIST2011], [NIST2011B], commercial anti-virus vendors [Sym2009], [Sym2011], research groups [CWG2011], and in repositories such as those provided by the National Vulnerabilities Database [NVD], or the vulnerability taxonomy volume published by MITRE [CWE], or even by experimental means [Raft2011]. Ultimately, to maintain maximal flexibility and applicability, it is necessary for model users to have the ability to define their own threat vectors, and to associate to these vectors incidence rates based on historical organizational data or the results of penetration tests and security audits. CyberV@R will therefore model a threat vector (or thread of execution) abstractly, as as a sequence of threat stages, each of which may target certain types of infrastructure, with a certain base rate of occurrence, and with certain propensities to overcome, or be thwarted by, certain types of mitigating security actions. At the model level, these stages will be represented as nodes in an attack tree representing the threat's interaction with the infrastructure; each threat node will be decorated with a probability distribution providing the associated threat stage's incidence rate. A formal definition is provided in the sections below. Any implementation of the model will of course ideally present users with a pre-built set of threat vectors and threat stages, with which users can model the specific types of threats to which their organizations may be subject.

1.3. **Characterization of countermeasures.** The SANS Institute's "Twenty Critical Security Controls for Effective Cyber Defense: Consensus Audit Guidelines" [Sans2011] are an oft-cited source for a taxonomy of security measures IT managers may take, to protect organizational technology infrastructure from cyber threats. The specifications of the controls include descriptions of the threats the controls seek to mitigate, and metrics for gauging the efficacy of the controls. There are also various commercially available products on the market that measure cyber risk in some form. McAfee's RiskAdvisor [McAfee] allows IT managers to track patch levels of IT assets and assess the exposure of those assets to various forms of threat (essentially malware) as provided by the McAfee signature-based MITS automated threat feed. For each asset and threat, a risk score is developed, taking discrete variables indicating the value of the asset, the severity of the threat, the exposure of the IT asset to the threat (e.g. patched vs. not patched), and the presence of countermeasures (e.g. AV software vs. no AV software). These data may be aggregated in reporting tools. Rapid7 provides a similar risk scoring and reporting framework, built on top of its Rapid7 Nexpose and Metasploit tools [Rapid7]. ArcSight EMS [ArcSight] also provides a risk assessment framework, based on correlation of network log data. These types of tools may be of potential use when constructing the organization-specific attack trees that form part of the inputs to the CyberV@R model. But again, to allow for flexibility of application, mitigations (countermeasures) in the CyberV@R model will be represented abstractly, as nodes in an attack tree, each decorated with a probability distribution giving the likelihood that the associated mitigation will defeat any one incident of a threat stage execution attempt, for a predefined set of threat types and stages.

1.4. **Characterization of infrastructure.** Modern networked information technology infrastructure involves a plethora of physical and virtual entities: servers, workstations, routers, switches, wireless devices, mass storage devices, Virtual Private Networks, Local Area Networks, Wide Area Networks, and so forth, operating on a plethora of operating systems and versions thereof [Peter2012]. Modeling this infrastructure is a daunting task in and of itself. As with threats and mitigations above, we model elements of the infrastructure as nodes in an attack tree, in this case internal nodes for which threat stage and mitigation nodes are parents. In the following, we refer to such nodes generically as "access" nodes. Nodes may in theory represent individual devices, but by intent will model larger segments of the network infrastructure: DMZs, internal subnets, stand-alone subsystems, application servers, classes of remote devices, and so forth.

1.5. **Characterization of assets.** In [Gor2004], a one-period model is presented for modeling the economic benefit of security measures undertaken to mitigate risks to infrastructure, involving computation of the expected decrease in economic loss L due to such measures. We will be concerned with analogous computations, when we discuss the application of CyberV@R to optimization problems. In our model, losses will be attributed to successful attacks directed against access nodes housing assets of monetary value V. The value may represent either an expected gain due to the asset – e.g., market value if it is intellectual property – or an expected liability should the asset be unintentionally divulged – e.g., if it is personally identifying information (PII). Asset nodes will be modeled as attack tree leaf nodes, having access nodes as parents, and labelled with a loss likelihood function, conditioned on a successful attack directed against the parent access node. Since our model is time-evolving, the function will also be conditioned on the incidence of a prior successful attack (a manifestation of the dynamism of the attack tree network).

1.6. **Computing loss likelihoods.** As with the financial VaR model described above, we ultimately seek to derive a dollar figure representing value at risk, by constructing and summing over a probability distribution that models our loss likelihood, over a given time horizon. As noted above, the distribution will take the form of a dynamic Bayesian network, constructed from a set of input attack trees, representing possible combinations of threats, countermeasures, target infrastructure, and defended assets. The network will represent the joint distribution of losses conditioned over all modeled forms of attack. VaR computations will be performed by running Monte Carlo simulations according to the distributions, to obtain a sample loss distribution, from which a loss threshold for a given confidence interval may be calculated, by analogy with equation (1).

We now turn to the formal definition of the model (after a motivating example).

2. Computing CyberV@R

In this section we will present a (more) formal definition of our model for computing CyberV@R. To provide concreteness, we will first however consider a simplest application of the model. From this example we will extract certain rules for constructing attack trees, that will guide our specification of the model proper.

2.1. **A simplest motivating example.** We posit a very simple world in which there is only one category of cyber risk - that associated with SANS Institute Critical Security Controls number 1 and 2 [Sans2011]. The risk is (roughly) "unauthorized network access," which we treat as a one-stage threat. We further posit the existence of just one of the security measures specified by the controls, which are intended (in a somewhat simplified interpretation) to thwart attacks related to unauthorized network access. Since our example is a very simple one, it calls for a very simple network, which we will depict as having a single access node (one either breaches the network, or not). Associated to that node will be an asset node representing a document repository of intellectual property exposed to the threat of theft via unauthorized access.

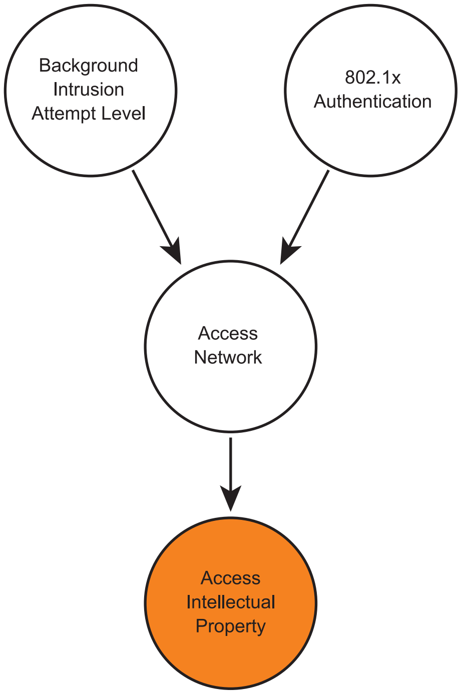

Figure 1. Overly simplified example of a Bayesian network for Critical Controls number 1 and 2, for a hypothetical organization.

Our model inputs call for a specification of threat incidence rates and mitigation effectiveness rates. We first consider sources for the prevalent background rate of attacks related to unauthorized network access. The following is meant to be illustrative only; the model definition does not specify any particular source of input data. With this caveat, we note that the MITRE Corporation maintains a hierarchical classification of cyber attack mechanisms known at the Common Attack Pattern Enumeration and Classification (CAPEC) repository [CAPEC]. Types within this hierarchy (identified by numbers known as CAPEC-IDs) are correlated to MITRE-maintained Common Weakness Enumeration (CWE) repository entries [CWE], which themselves are assigned identifier numbers (CWE-IDs). These in turn may be used to search for instances of vulnerabilities recorded in the DHS/NIST National Vulnerability Database of Common Vulnerabilities and Exposures (CVEs) [NVD]; these too are assigned identifier numbers (CVE-IDs). One of the top-level CAPEC attack mechanisms is “Exploitation of Authentication” (CAPEC-ID 225), which corresponds reasonably well to our posited risk category. This category of attack covers a variety of attack subtypes, to which (at the time of writing) the following list of CWEs were correlated (the given numbers are the CWE identifiers, and the *s are explained below):

6 J2EE Misconfiguration: Insufficient Session-ID Length *

20 Improper Input Validation

113 Improper Neutralization of CRLF Sequences in HTTP Headers (’HTTP Response Splitting’)

200 Information Exposure *

285 Improper Authorization

287 Improper Authentication *

290 Authentication Bypass by Spoofing

294 Authentication Bypass by Capture-replay *

301 Reflection Attack in an Authentication Protocol

302 Authentication Bypass by Assumed-Immutable Data *

303 Incorrect Implementation of Authentication Algorithm

306 Missing Authentication for Critical Function

311 Missing Encryption of Sensitive Data

315 Plaintext Storage in a Cookie

319 Cleartext Transmission of Sensitive Information

328 Reversible One-Way Hash *

346 Origin Validation Error

352 Cross-Site Request Forgery (CSRF)

359 Privacy Violation

361 Time and State

384 Session Fixation *

472 External Control of Assumed-Immutable Web Parameter

488 Exposure of Data Element to Wrong Session

522 Insufficiently Protected Credentials *

523 Unprotected Transport of Credentials *

539 Information Exposure Through Persistent Cookies

565 Reliance on Cookies without Validation and Integrity Checking

592 Authentication Bypass Issues

602 Client-Side Enforcement of Server-Side Security

614 Sensitive Cookie in HTTPS Session Without 'Secure' Attribute *

642 External Control of Critical State Data

664 Improper Control of a Resource Through its Lifetime

693 Protection Mechanism Failure

732 Incorrect Permission Assignment for Critical Resource.

Searching by CWE-IDs in the NVD produces vulnerability instances (CVEs) labelled with the date of discovery. The instances are for a variety of distinct software and hardware platforms and severity levels. Nonetheless, taken in aggregate they provide some information as to year-over-year changes in the rate of discovery of vulnerabilities related to certain types of attacks. One may take the change in rate of discovery for a given class of vulnerabilities as a leading indicator for the change in rate of attack instances exploiting the vulnerability class (on the theory that what *can* be exploited *will* be exploited). One may then use the discovery rate delta to adjust forward-looking risk rates, relative to some established benchmark, say one established by an individual organization through internal inspection of its own historical record of relevant cyber incidents. For our example attack pattern "Exploitation of Authentication," searching on the associated CWE IDs within the NVD yields 5 related CVEs for 2009 and 13 related CVEs for 2010. From this we conclude (absent other information) that the rate of authorization/authentication-related attacks was growing at 160 percent annually, at the beginning of 2011.

Now we proceed with our example risk valuation. Consider an organization with a 2010 daily benchmark rate θ for authentication/authorization attacks, based on internally reported or detected unauthorized access-related incidents. It wishes to compute the associated CyberV@R - the value at risk of loss due to such attacks, over the first ten days of 2011. To compute the CyberV@R, its chief risk officer (CRO) constructs the Bayesian network depicted in figure (1) (see [Kol2009] for a detailed discussion of Bayesian networks). The graph depicts one type of attack risk, represented by the graph node labelled "Background Intrusion Attempt Level". It also posits exactly one type of risk mitigation, "802.1X Authentication," the only network access control (NAC) technology in use by the firm. Unauthorized access to any of the organization's computing assets is represented by a single node, "Access Network." The directed edges to that node from the previous two nodes indicate that the occurrence of unauthorized access is dependent on both the background intrusion attempt rate, and the efficacy of the mitigating technology. Actual theft of intellectual property is represented by the "Access Intellectual Property" node. Such theft is directly dependent only on the occurrence of successful unauthorized access attempts.

To compute the CyberV@R, the CRO must construct a joint probability distribution that conforms to the graph. Each node represents a random variable, described by a probability distribution, conditioned on the variables corresponding to the parent nodes (if any exist). The joint distribution will model the dependencies between the variables, as given by the graph. In particular, let B denote the random variable associated with "Background Intrusion Attempt Level", P_B its (discrete) probability distribution, and let $P_B(B = b)$, or $P_B(b)$ for short, denote the probability of B taking a particular value b in its possible range of values. Similarly, let Ξ, A and L denote the random variables associated to "802.1X Authentication," "Access Network," and "Access Intellectual Property" respectively. The nature of the distributions and value ranges will be discussed below. Even without this information, the graph tells us that $P(B = b, \Xi = \xi, A = a, L = l)$, the probability of B, Ξ, A, L taking a given tuple of values (b, ξ, a, l), factors as:

$$(2) \qquad P(B = b, \Xi = \xi, A = a, L = l) = P_B(b)P_\Xi(\xi)P_A(a|B = b, \Xi = \xi)P_L(l|A = a),$$

where $P_I(i|J = j)$ denotes the probability of variable I taking value i, given that variable J took value j.

So, let us turn to the definition of the node-specific variables. Given the information above on the growth rate of authentication-related CWEs, the corresponding forward-looking unauthorized access attempt rate for the organization, for the d-th day of 2011 (counting from 0), might accordingly be estimated as $\lambda_d = 1.6\frac{d}{364}\theta + \theta$ (the estimated "close of day" rate), absent any other relevant information. The probability distribution associated to such attack attempts, and hence to the

"Background Intrusion Attempt Level" node of figure (1), might therefore be modeled by a Poisson distribution, in which the probability $P_B(n,d)$ of n attack attempts on day d of 2011 is given by $P_B(n,d) = \frac{\lambda_d^n}{n!}e^{-\lambda_d}$ [Bart1996]. Let's assume there is some upper bound M on the conceivable number of attacks that could be attempted on a given day. We accordingly modify $P_B(n,d)$ so that it returns 0 if $n > M$. Note also the introduction of the time parameter d in the arguments to the distribution for B. We may think of $B(d)$ as a random variable that gives a number of attacks occurring on day d; it gives a specific value n with frequency $P_B(n,d)$. Some of the other variables corresponding to our graph nodes are time-dependent as well, in the scenario we're presenting. As we'll see, to compute the CyberV@R, we will deal not in computations over a single graph, but rather over a time-indexed family of graphs and associated joint distributions.

We've assumed that the only mitigation available for the example risk is the use of network access control technology based on the IEEE 802.1x standard [IEEE2010]. The standard is credentials-based and hence, we assume, defeatable by exploits of weaknesses involving the theft of credentials or the bypassing of credential requirements. Obviously specific implementations of the standard will vary in robustness (e.g. certificate-based schemes vs. password-based schemes). Nonetheless, arguably 29 percent of the CWEs listed above (those marked with a *) involve theft of credentials, or bypassing of authentication requirements, that could conceivably be executed in some (though certainly not all) 802.1X environments. Hence in the absence of information about the organization's specific implementation of the technology, we place the technology's effectiveness rate at 71 percent, and so the random variable Ξ associated to the "802.1X" node of figure (1), takes the value $\xi = 0.71$ with certainty.

In our example graph, we've collapsed our organization's IT network infrastructure into a single logical device, which either is or is not accessed in any given unauthorized access attempt. Accessing the network requires circumvention of the network's 802.1X-based authentication technology. Hence whether or not the network is accessed is conditioned on two independent variables - the background attack level $B(d)$, and the efficacy of our authentication technology Ξ. As noted, we represent this state of affairs via the "Access Device" node of figure 1, representing the random variable $A(d)$, taking values 1 (at least one successful attack on day d), and 0 (no successful attacks on day d). The odds $P_A(d)$ of there being at least one successful attack on day d is given by:

$$P_{A(d)}(A(d) = 1 | B(d) = n, \Xi = \xi) = \frac{\lambda_d^n}{n!}e^{-\lambda_d}(1 - \xi^n).$$

Observe that either a decrease in the background attack rate, or an increase in the efficacy of our authentication technology to thwart any given attack, will decrease $P_A(d)$, as one would expect.

Let's assume for simplicity that the network contains documents detailing all the organization's intellectual property (IP), having some time-dependent total value $V(d)$ (measured in dollars), and that if the network is accessed by unauthorized means one or more times on any given day d, then $\frac{1}{r}V(d)$ of the value is removed on that day (regardless of the specific number of unauthorized accesses), where r is a fixed parameter.[1] Hence to the "Access IP" node of figure (1) we assign the random variable $L(d)$ taking values $l = \frac{1}{r}V(d)$ and 0; its conditional distribution $P_L(d)(l|A(d) = a)$ returns $v = \frac{1}{r}V(d)$ with certainty if $a = 1$, and 0 with certainty if $a = 0$.

[1] Use of a fixed loss rate is not without precedent. In some pricing models for credit default swaps, the recovery rate for the defaulted bond is fixed by assumption; c.f. [Lando2004] p. 120.

So far, the above ingredients provide a method for computing the expected loss $E[L, d]$ on day d, given by:

$$E[L, d] = \sum_{n=1}^{n=M} \frac{\lambda_d^n}{n!} e^{-\lambda_d} (1 - (\xi)^n) \frac{1}{r} V(d).$$

But what we really want is the dollar value C, such that the probability of losing more than C over the next y days is less than or equal to some cutoff q, say $q = 0.02$, for the sake of concreteness. One approach, having the advantage of being conceptually if not computationally tractable for arbitrary joint distributions, is to run some number of simulations $S_i, i = 1, \cdots, K$, where for our example we take $K = 100$. Each simulation S_i involves y iterations $t_{i_0}, \cdots, t_{i_{y-1}}$. Effectively, we are computing over the graph depicted in figure (2). We track separate $V_i(d)$ for each simulation S_i; initially they are all initialized to $V(0)$. On each iteration t_{i_d}, a value b_{i_d} is drawn according to $P_B(d)$, and then a value a_{i_d} is drawn according to $P_A(d|B(d) = b_{i_d}, \xi)$, and then a value l_{i_d} is drawn according to $P_L(d)(L(d)|A(d) = a_{i_d})$, [2] and finally, for $d < y - 1$, we set $V_i(d + 1) = V_i(d) - l_{i_d}$. At the completion of the iterations for the simulation S_i, the values l_{i_d} are summed to produce a total loss l_i for the simulation S_i; that is, $l_i = \sum_{d=0}^{d=y-1} l_{i_d}$. We may rank in decreasing order the l_i produced by the simulations S_i. Let C be the value of third entry. Then only 2% of the values l_i are greater than or equal to C. Hence C is the desired value at risk.

There are numerous ways in which complexity can be added to this approach, to more accurately model real-world risk situations. Additional risk, mitigation, IT infrastructure, and intellectual property nodes may be added, as illustrated in figure (3). More complex dependencies may be modeled, using more sophisticated conditional probabilities. New categories of nodes may be posited. We might introduce stochastic evolution of various background rates. And finally, graphs created for different risk areas (say, as given by other controls within the “20 Critical Controls” document), may be merged together to form networks of graphs - all evolving over time.

Moreover, one can envision that a full cyber risk valuation system, based on the methodology sketched in this section, will present the user with a set of Bayesian network templates, based on current real-world risks, risk mitigations, and common IT infrastructure configurations, visually manipulable and customizable by end users, and capable of being combined like building-blocks in a variety of ways, to form complex, organization-specific risk valuation models. The parameters for these models may come in part from real-time data feeds ingested by the system from external entities.

To facilitate progress towards a general conceptual framework supporting such risk valuation models, we now present more formally the CyberV@R methodology for computing value at risk of cyber attack. The next section summarizes the mathematical background, as found in [Kol2009], [Press07], requisite for the formal specification of the CyberV@R methodology.

2.2. **Bayesian Networks, Dynamic Bayesian Networks, and Markov Chain Monte Carlo (MCMC) Simulations.** Following [Kol2009], a *Bayesian Network* is a directed acyclic graph in which each vertex v represents a random variable with probability distribution P_v, such that if v has parents $p_1, p_2, \cdots, p_n$, then P_v is conditioned on the p_i. Hence the graph represents both a joint probability distribution P on all the vertices v_i, and a factorization of the distribution into (possibly) conditional distributions P_s which are independent. For example, the attack tree described in section 2.1 is a Bayesian network having a joint distribution with factorization given

[2] General methods for drawing samples from a given distribution may be found in texts such as [Robert2004].

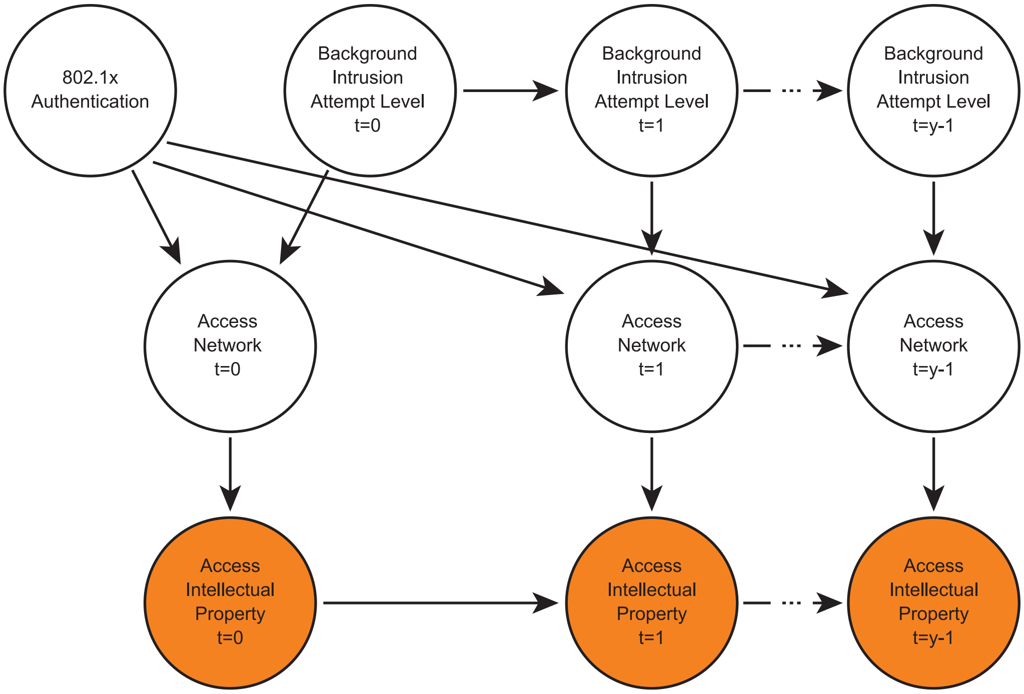
802.1x Authentication
Background Intrusion Attempt Level t=0
Background Intrusion Attempt Level t=1
Background Intrusion Attempt Level t=y-1
Access Network t=0
Access Network t=1
Access Network t=y-1
Access Intellectual Property t=0
Access Intellectual Property t=1
Access Intellectual Property t=y-1

FIGURE 2. Time evolutiuon of the Bayesian network for Critical Controls number 1 and 2.

by equation (2), where two factors have no conditions, one factors is conditioned on one random variable, and one factor is conditioned on two random variables.

A *two time-slice* dynamical Bayesian network is a pair $\langle \mathcal{B}_0, \mathcal{B}_\rightarrow \rangle$, where $\mathcal{B}_0$ is an initial Bayesian network, and $\mathcal{B}_\rightarrow$ is a process for obtaining (1) a Bayesian network B_t from a Bayesian network B_{t-1}, where the vertex sets $V(B_t)$ of B_t and $V(B_{t-1})$ of B_{t-1} are identical, and the edge set $E(B_t)$ of B_t is a subset of the edge set $E(B_{t-1})$ and (2) a Bayesian network with vertex set $V(B_t) \bigcup V(B_{t-1})$ and an edge set containing $E(B_t)$ and $E(B_{t-1})$, at least one edge from a vertex of B_{t-1} to a vertex of B_t, and no edges of any other form. In other words, $\langle \mathcal{B}_0, \mathcal{B}_\rightarrow \rangle$ gives a discrete time evolution of a Bayesian network B_0, in which no new intra-time dependencies are introduced, but at least one inter-time dependency is introduced from B_{t-1} to B_t. Note the time-evolution of $\langle \mathcal{B}_0, \mathcal{B}_\rightarrow \rangle$ over any fixed number of time steps is itself a Bayesian network. Figure (2) depicts a Bayesian network obtained from such a process of time evolution. In what follows, we shall be concerned with dynamic Bayesian networks in which each random variable, at each time step, takes values in a discrete set.

We observe that the two time-slice dynamical Bayesian network $\langle \mathcal{B}_0, \mathcal{B}_\rightarrow \rangle$ defines a Markov chain, in that at each time B_t, we have a Bayesian network B_t with joint probability distribution P_t, which may be regarded as the distribution for a random variable $\mathbf{X}(t)$, whose possible outcomes $\{\mathbf{x}_i(t)\}_{i \in \mathcal{I}}$, for $\mathcal{I}$ a suitable index set, are the distinct tuples of outcomes of the individual nodes in the graph of B_t. Then by the definition of $\langle \mathcal{B}_0, \mathcal{B}_\rightarrow \rangle$, we have that $\mathbf{X}(t)$ is (externally) conditioned only on the value taken by $\mathbf{X}(t-1)$. Therefore we can sample from $\langle \mathcal{B}_0, \mathcal{B}_\rightarrow \rangle$ via MCMC sampling to obtain, over repeated invocations of MCMC up to a fixed time T, a sample distribution of values for $\mathbf{X}(T)$ at T, from which we can make statistical inferences. In the case of a dynamical Bayesian network in which each node is associated to a discrete conditional probability distribution taking a tabular form, the sampling is straightforward, as given by **Algorithm 1**.

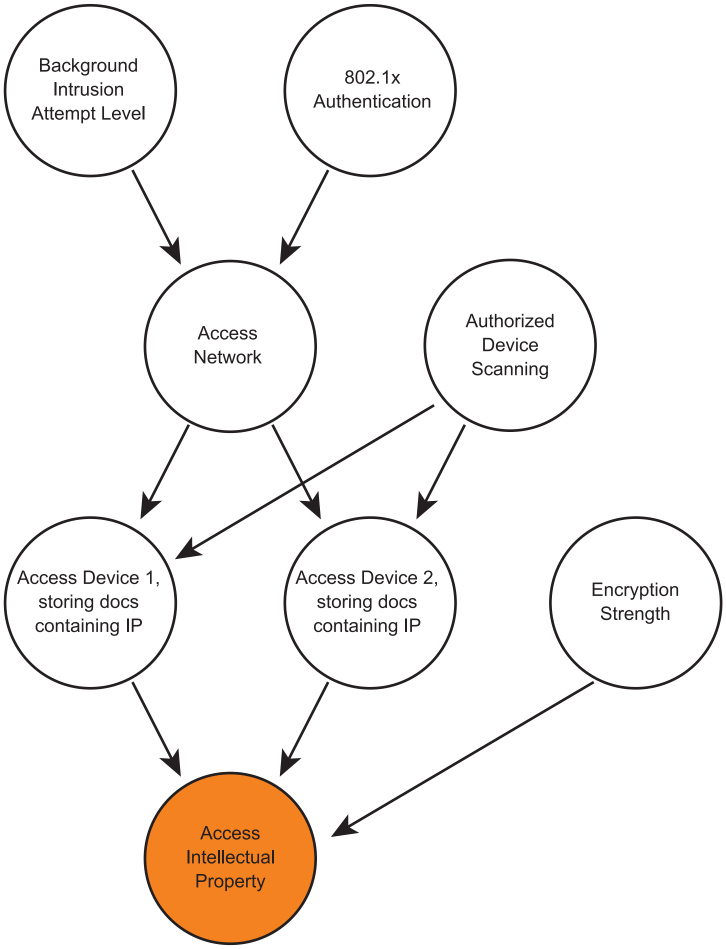
Background Intrusion Attempt Level
802.1x Authentication
Access Network
Authorized Device Scanning
Access Device 1, storing docs containing IP
Access Device 2, storing docs containing IP
Encryption Strength
Access Intellectual Property

Figure 3. Slightly more complicated Bayesian network for Critical Controls number 1 and 2.

2.3. **CyberV@R via Dynamic Bayesian Networks: core rules.** The CyberV@R model is characterized formally as the specification of:

- a dynamic Bayesian network $\langle \mathcal{B}_0, \mathcal{B}_\rightarrow \rangle$
- a regime for sampling from $\langle \mathcal{B}_0, \mathcal{B}_\rightarrow \rangle$ up to a fixed time T
- statistical inferences made on the resulting sample distribution at time T

where each node in the Bayesian network B_t at each time t is one of four types:

(1) A risk (or "threat stage") node, denoted $R_{i,\tau}(t)$, where i is a label unique across threat stage nodes, τ is label unique across threat stage types, and t is the time index. Threat stage nodes represent specific stages of specific types of cyber attacks (or "threat threads"), and for a given threat thread, have a fixed ordering. Threat stage nodes accept no incoming directed edges other than a single edge from a single access node a (defined below) having time index t, and have outgoing edges to access nodes of time index t only. Risk nodes may be roots of the CyberV@R Bayesian network (CBN). If a threat stage node does not have a parent access node, then its corresponding probability distribution, $P_{R_{i,\tau}}(n|t)$, shall be a Poisson distribution with time-dependent parameter $\lambda(t)$, and shall give the odds that n attacks, associated to the threat stage type τ, will occur between time index t and $t+1$. If the threat stage node has a parent access node a, then a must have a parent threat stage

node of type that is the predecessor of (or is equal to) $P_{R_{i,\tau}}(n|t)$ in the threat thread's fixed ordering of stage types, and letting $n_a(t)$ denote the number of successful access of the parent at time t, the distribution for $R_{i,\tau}(t)$ is one of:

$$\text{Poisson, with } \lambda(t) \quad \neq 0 \quad \text{if and only if } n_a(t) > 0 \text{ , or} \tag{3}$$

$$n_a(t) \quad \text{with certainty, or} \tag{4}$$

$$m \in \mathbf{Z}^+ \quad \text{with certainty} \quad \text{if and only if } n_a(t) > 0 \tag{5}$$

depending on the threat stage node specification (see section 2.4). In the first equation, there may be an upper bound of $n(a)$ imposed on the possible outcomes, depending on the threat stage specification. Finally, we note here that a threat stage node having a parent access node may indicate (see the schema inputs in subsection 2.4) that its invocation should set the state of its parent access node to "not accessed" (the threat stage represents a transiting rather than propagating action).

(2) a Mitigation node, denoted $M_{j,\xi}(t)$, where j is a label unique across mitigation nodes, and ξ is a label unique across mitigation types, such that ξ is a recognized mitigation for τ, denoted $\mathcal{I}_\tau(\xi) = 1$, and again t is the time index. The corresponding probability distribution will be a Bernoulli variable $M_{j,\xi}$ (independent of time) giving the odds of the mitigation thwarting any given threat stage instance of type τ. Mitigation nodes have outgoing directed edges to access nodes only (see below). Mitigation nodes represent instances of attack countermeasures, and are roots of the CBN. As indicated in the input schema (see subsection 2.4), some mitigations may only (or additionally) serve, with a certain likelihood, to remove a threat once it has infected a child access node. In this case, the mitigation node $M_{j,\xi}(t)$ at time t will have an incoming edge from $A_k(t-1)$, where $A_k(t)$ is the child access node of $M_{j,\xi}(t)$. Mitigation nodes accept no other incoming edges.

(3) an Access node, denoted $A_k(t)$, where k is a label unique across access nodes, and t is the time index. An access node accepts an incoming directed edge from a single threat stage node $R_{i,\tau}(t)$, zero or more incoming edges from mitigation nodes $(M_{j_1,\xi_1}, \cdots, M_{j_N,\xi_N})$ where $\mathcal{I}_\tau(\xi_q) = 1$ for $q \in \{1 \cdots N\}$, and for $t > 1$, an incoming edge from $A_k(t-1)$. If the threat stage parent node $R_{i,\tau}(t)$ has a parent $A_q(t)$ access node, then that node must in turn have a parent threat stage node $R_{h,\tau'}(t)$, such that τ' must be equal to τ, or else be the immediate predecessor of τ in the ordered list of threat stage node types included in the containing threat thread. Since we presume the odds of n attempts all failing is given by $(1-p)^n$ where p are the odds of a single attempt succeeding, the associated conditional probability distribution for $A_k(t-1) = 0$ is given by:

$$P_{A_k}(A_{k,t} = 1|t) = \sum_{n=1}^{n=B_{k,t}} \frac{\lambda_t^n}{n!} e^{-\lambda_t} [1 - (1 - (1 - M_{j_1})(1 - M_{j_2}) \cdots (1 - M_{j_N}))^n] \ , \tag{6}$$

$$P_{A_k}(A_{k,t} = 0) = 1 - P_{A_k}(A_{k,t} = 1), \tag{7}$$

where $A_{k,t} = 1$ indicates the event "one or more attacks occur between time t and time $t+1$ on node $A_{k,t}$ and $A_{k,t} = 0$ represents the event "no attacks occur on node $A_{k,t}$ between time t and time $t+1$," and $B_{k,t}$ is a bound on the number of attacks associated to $A_{k,t}$ that can occur between time t and time $t+1$. For $A_k(t-1) = 1$, if $A_k(t)$ has no mitigation parent nodes with post-infection effectiveness rates, then we have $P_{A_k}(A_{k,t} = 1) = 1$. If $A_k(t)$

does have such parent mitigation nodes, and was accessed at $t-1$, then the conditional distribution takes the form:

$$P_{A_k}(A_{k,t}=1|t) = (1-M_{j_1})\dot{(}1-M_{j_2})\cdots(1-M_{j_N}), \tag{8}$$

where the M_{j_l} are those parent mitigations having post-infection effectiveness rates.

Access nodes may have outgoing edges to asset nodes (see below) or threat stage nodes. Access nodes represent the roles played elements of the IT infrastructure (anything from a port to an entire system) within an attack tree (so access node labels refer to distinct role/element pairs). Access nodes are neither roots nor leaves of the CBN.

(4) an Asset node, denoted $V_l(t)$ where l is a label unique across asset nodes, and t is the time index. An asset node represents the aspect of the organization (intellectual property, operational continuity, absence of legal liability) that is at risk due to cyber attack. Asset nodes accept incoming edges from one or more access nodes $A_{k_1}, \cdots, A_{k,N'}$. They have no outgoing nodes; they are the leaves of the CBN. The associated probability distribution $P_{V_l}(v|t, A_{k_1}(t)\cdots A_{k'_N}(t))$ returns 1 with certainty if $v=\nu(l,t)$ where $\nu(l,t)$ is a loss amount associated to $V_l(t)$, and at least one $A_{k_i}(t)=1$, and 0 otherwise. In practice, $\nu(0,l)$ is assigned a fixed asset value $v_l(0)$ and a loss percentage $\Delta\nu_l$ and $\nu_l(t+1)=\max(0,\nu_l(t)-\Delta\nu_l\cdot\nu_l(t))$ if a loss occurs at time t, and $\nu_l(t)$ otherwise. Hence the asset node $V_l(t)$ implicitly accepts an incoming edge from $V_l(t-1)$.

Essentially, the model is a dynamic Bayesian network which may be used to simulate the exposure of a IT infrastructure to one or more types of multi-stage cyber attack (threat thread). Note that at each time slice, we implicitly employ a separate Bayesian sub-network (or "attack tree") for each threat thread (this is because for simplicity, each access node may only accept an input from one threat stage, hence a copy of the access node must appear in the network for each threat stage that is applicable to it). As the mitigation nodes have no dependencies, the same node may participate in multiple Bayesian sub-networks. The threat thread-specific subnets combine at the asset leaf nodes to form a single network.

The input data describing these nodes, attack trees, and dynamic attack tree network shall take the form of a Java Simple Object Notation (JSON) text file [JSON], containing data representing the threat threads, threat stages, mitigation, IT infrastructure, and defended assets, and conforming to the above rules.

2.4. **CyberV@R via Dynamic Bayesian Networks: input formats.** As noted above, every node in the dynamic bayesian network via which the VaR is calculated will be either a risk node, a mitigation node, an access node, or an asset node. The risk nodes represent stages of an evolving cyber threat. The mitigation nodes represent the defenses an organization may deploy to counteract or reduce the impact of a threat stage. The access nodes represent the appropriate segments of a network topology (a DMZ, a financial, human resources, or engineering subnetwork, an application firewall, or perhaps even an individual air-gapped computer, where relevant). The asset nodes represent valued information, either due to commercial potential, as with intellectual property, or because of liabilities and organization would incur due to its loss, as with health or tax records or other personally identifying information (PII). Accordingly, users will want to specify model inputs in these terms, and leave to the hosting computer system the task of constructing the network of attack trees the inputs describe. Moreover, users may wish to treat certain threats or mitigations at a different level of detail from others (for example, to indicate that a certain mitigation applies only to a given device, or a particular strain of a threat). To this end, we now provide a specification of input structures that will representing the nodes or node sets, and carry user-supplied information that may be used to combine them into dynamic networks. The specification will be given in a loosely structured manner, following the proposed Javascript Simple Object Notation (JSON) schema notation described at [JSONSchema] and [JSONTutorial]. Corresponding to the four node types, the schema defines four types of input four entities: a *threat thread entity*, a *mitigation entity*, an *access entity*, and a *node entity*.

2.4.1. *JSON schema: threat thread type.* . The JSON schema for the threat thread input entity is as follows: {

```
    “description” : “ A set of risk nodes representing the stages of a cyber threat”,
    “type” : “object”,
    “properties” : {
        “threat_type” : {
            “title” : “specifies a threat type to which the risk belongs”,
            “type” : “string”,
            “required” : true,
        },
        “threat_name” : {
            “title” : “the UUID of the threat thread instance”,
            “type” : “string”,
            “required” : true,
        },
        “intent_type” : {
            “title” : “specifies the intent type name associated to the threat instance”,
            “type” : “string”,
            “required” : true,
        },
        “threat_stages” : {
            “title” : “stages of the threat (ordered sequentially)”,
            “type” : “array”,
            “required” : true,
            “items” : {
                “type” : “object”,
                “properties” : {
                    “stage_type” : {
                        “title” : “type of stage”,
                        “required” : true,
                        “type” : enum[“reconnaissance”, “initial”, “persistence”, “beaconing”, “propagation”, “exploitation”, “exfiltration”],
                    },
                    “stage_name” : {
                        “title” : “UUID for the threat thread stage node”,
                        “type” : “string”,
                        “required” : true,
                    },
                    “objective” : {
                        “title” : “description of the stage objective”,
                        “type” : “string”,
                    },
                    “child_prob_type” : {
```

```
    “title” : “type of probability when not a root node (lambda or parent accesses w/certainty or fixed m)”,
    “type” : enum: [ “parent_access”,“lamdba”, “int”]
},
“base_lambda” : {
    “title” : “initial lambda rate”,
    “type” : “float”,
    “minimum” : 0.0,
    “exclusiveMinimum” : true,
},
“growth” : {
    “title” : “constant for linear growth rate”,
    “type” : “float”,
},
“upper_bound” : {
    “title” : “maximum number of occurrences in any timestep - ‘parent’ indicates limit to number of successes of prior threat stage node”,
    “type” : enum[“int”, “parent”]
},
“repeats” : {
    “title” : “stage may repeat as required”,
    “type” : “true’,
},
“transits” : {
    “title” : “upon invocation, stage sets status of parent node to ‘not accessed’ ”,
    “type” : “boolean”,
},
“skippable” : {
    “title” : “node may be skipped in sequence of stages along any given attack path”,
    “type” : “boolean”,
},
“mechanisms” : {
    “title” : “mechanisms of action via which threat stage may be applied”
    “required” : true,
    “type” : “array ”,
    “items” : {
        “type” : “object ”,
        “properties” : {
            “mechanism_type” : {
                “title” : “type of the mechanism”,
                “type” : “string”,
                “required” : true,
            },
            “connectivity_types” : {
                “title” : “connective types of the mechanism: required for propagation stages”,
                “type” : “array”, “items” : { “type” : “string” },
            },
            “defeats” : {
```

```
                            “title” : “ list of names of mitigations this threat stage defeats,”
                            “type” : “array”,
                            “items”: {
                                “type”: “object”,
                                “properties” : {
                                    “mitigation_type” : {
                                        “title” : “type of the mitigation”,
                                        “required” : true,
                                        “type” : “string”,
                                    },
                                    “mitigation_qualifiers” : {
                                        “title” : “mitigation subtype indicators”,
                                        “type” : “array”,
                                        “items”: {
                                            “type” : “string’,
                                        },
                                    },
                                },
                            },
                        }
                        “applies_to” : {
                            “title” : “ list of types of access nodes to which this threat stage applies,”
                            “type” : “array”,
                            ‘items”: {
                                “type”: “object”,
                                “properties” : {
                                    “access_type” : {
                                        “title” : “type of the access node”,
                                        “required” : true,
                                        “type” : “string”,
                                    },
                                    “access_qualifiers” : {
                                        “title” : “access subtype qualifier sets (each set is an array of attack surface designators)”,
                                        “type” : “array”,
                                        “items” : {
                                            “type” : “array” : “items”: { “type” : “string” },
                                    },
                                },
                            },
                        },
                    },
                },
            },
        },
    },
}
“default_rate” : {
    “title” : “default base lambda rate for all stages”,
            “type” : “float”,
            “required” : true,
            “minimum” : 0.0,
            “exclusiveMinimum” : true,
        },
        “default_growth” : {
            “title” : “default growth constant for all stages”,
            “type” : “float”
            “required” : true,
        },
        “targets” : {
            “title” : “ list of types of asset nodes to which this threat stage applies,”
            “type” : “array”,
            “items”: {
                “type”: “object”,
                “properties” : {
                    “asset_type” : {
                        “title” : “type of the access node”,
                        “required” : true,
                        “type” : “string”,
                            },
                },
            },
        },
}
```

Essentially, a threat thread consists of a threat type name, a threat instance name (UUID across all threat instances), an associated intent type name, and a set of one or more threat stages. Each threat stage consists of a stage type name, a instance name (UUID across all node types), an optional λ value defining the initial Poisson distribution associated to the stage instance (or a prior stage-dependent or fixed number, as in equations (3,4,5), and an optional linear growth rate constant c for the λ value. The stage may also specify that it is skippable and/or repeatable; these are an aid in automated attack tree construction (see section 2.8) . Each stage also specifies one or more mechanisms of action, which in turn may specify a list of connectivity types. Each mechanism of action specifies a list of access nodes types (with subtype qualifiers) for which the stage is applicable (this type and qualifier information serves to define the “attack surface” types recognized by the threat). One or more arrays of strings may be specified. For the threat stage node to apply to an access node, it must have one array of qualifiers such that all qualifiers in the array are associated with the access node. Qualifier strings are intended to represent operating system types (e.g. “Windows7” or “RedHat5”), application types (e.g. “Adobe” or “MS Word”), or specific vulnerabilities (e.g. “CVE-123”). Each mechanism may also specify a list of mitigation types (with subtype qualifiers) which the mechanism is assumed to defeat. Finally, the threat thread specifies a default (λ, c) pair for use in all stages where stage-specific values are not provided.

2.4.2. *JSON schema: mitigation type.* . The JSON schema for the mitigation input entity is as follows: {

```
    “description” : “ cyber threat mitigation ”,
    “type” : “object”,
    “properties” : {
        “mitigation_type” : {
            “title” : “type name for the mitigation”,
            “type” : “string”,
            “required” : true,
        },
        “mitigation_name” : {
            “title” : “UUID for the mitigation node instance”,
            “type” : “string”,
            “required” : true,
        },
        “mitigates” : {
            “title” : “threat threads/stages mitigated”,
            “type” : “array”,
            “required” : true,
            “items” : {
                “type” : “object”,
                “properties” : {
                    “threat_type” : {
                        “title” : “threat type mitigated”,
                        “required” : true,
                        “type” : “string”,
                    },
                    “stage_type’ : {
                        “title” : “type of stage mitigated”,
                        “type” : “string”,
                    },
                    “action_type’ : {
                        “title” : “mechanism of action mitigated”,
                        “type” : “string”,
                    },
                    “effectiveness” : {
                        “title” : “effectiveness of mitigation for threat thread / stage (enter 0 if mitigation only applicable post-infection)”,
                        “type” : “float”,
                        “minimum” : 0.0,
                        “maximum” : 1.0,
                    },
                    “post_infect_effectiveness” : {
                        “title” : “effectiveness of mitigation for threat thread / stage, post-infection”,
                        “type” : “float’,
                        “minimum” : 0.0,
                        “maximum” : 1.0,
                    },
                },
            },
        }
        “applies_to” : {
            “title” : “ list of types of access nodes to which this mitigation applies,”
            “type” : “array”,
            ‘items”’: {
                “type”: “object”
                    “properties” : {
                        “access_type” : {
                            “title” : “type of the access node”,
                                “required” : true,
                                “type” : “string”,
                            },
                            “access_qualifiers” : {
                                “title” : “access subtype qualifiers”,
                                “type” : “array”,
                                “items: {
                                    “type” : “string’,
                                },
                            },
                        },
                    },
                },
            },
        },
        “default_effectiveness” : {
            “title” : “default effectiveness likelihood for all mitigated threats”,
            “type” : ”float”
            “required” : true,
            “minimum” : 0.0,
            “maximum” : 1.0,
        },
    },
}
```

A mitigation consists of a description, a mitigation type, a mitigation instance name and UUID, a list of (possibly qualified) threat nodes (with methods of action mitigated, and the mitigation effectiveness). If the mitigation provides a capability to remove a threat post-successful access, the effectiveness of this capability is also specified. A list of (possibly qualified) access nodes types for which the mitigation may be a parent node is also provided. Finally, a default effectiveness rate is specified (to be used for mitigated threats for which a threat-specific effectiveness is not indicated).

2.4.3. *JSON schema: access type.* . The JSON schema for the access node input entity is as follows: {

```
    “description” : “ cyber threat access node ”,
    “type” : “object”,
    “properties” : {
        “access_type” : {
            “title” : “type name for the access node”,
            “type” : “string”,
            “required” : true,
        },
        “access_name” : {
            “title” : “the UUID of the access node instance”,
            “type” : “string”,
            “required” : true,
        },
        “access_qualifiers” : {
            “title” : “access subtype qualifiers”,
            “type” : “array”,
            “items: {
                “type” : “string’,
            },
        },
    },
}
```

An access node entity consists of an access type name, an access instance name (UUID across all node types), and a list of access type qualifiers (which may be used to denote properties such as the hardware and software information applicable to the node; they serve to specify the attack surface type presented by the asset node in its role as target of the parent attack stage). If an access node A specifies a threat stage node T as a parent, then the following constraints must be satisfied:

(1) If A is of access node type t, then there must be an entry m in the “mechanisms” list of T, such that there is an entry e_m in the “applies_to” list of m, such that e_m specifies an “access_type” of t.

(2) If A specifies a qualifier set S, then the entry e_m must have a qualifier set Q_j in its “access_qualifiers” array such that $Q_j \subseteq S$ and $Q_j \neq \emptyset$.

Similarly, if A specifies a mitigation node ξ as a parent, then the following constraints must be specified:

(1) If A is of access node type t, then there must be an entry e' in the “applies_to” list of ξ for an access node of type t.

(2) If A specifies a qualifier set S, then the entry e' must have a qualifier set Q'_j in its “access qualifiers” array such that $Q'_j \subseteq S$ and $Q'_j \neq \emptyset$.

(3) If A has a parent threat stage node T, then the “mechanisms” entry m of T, satisfying the constraints on T, has a “mechanism_type” r such that ξ includes in its “mitigates” list an entry matching the pair (T, r).

(4) If A has a parent threat stage node T, then no “mechanisms” entry m of T, satisfying the constraints on T, may include an entry d in its “defeats” list, such that d specifies a “mitigation_type” matching the “mitigation_type” of ξ and has a non-empty set Q'' of mitigation qualifiers such that $Q'' \in Q_\xi$ where Q_ξ denotes the mitigation qualifiers of ξ.

2.4.4. *JSON schema: asset type.* . The JSON schema for the asset node input entity is as follows:

```
{
    “description” : “asset node (target of cyber threat) ”,
    “type” : “object”,
    “properties” : {
        “asset_type” : {
            “title” : “type name for the asset node”,
            “type” : “string”,
            “required” : true,
        },
        “asset_name” : {
            “title” : “UUID for the asset node”,
            “type” : “string”,
            “required” : true,
        },
        “drawdown_rate” : {
            “title” : “percentage by which single cyber incident decreases value”,
            “type” : “float’,
            “required” : true,
            “minimum” : 0.0,
            “maximum” : 1.0,
        },
        “initial_value” : {
            “title” : “initial value of the asset”,
            “type” : “float’,
            “required” : true,
            “minimum” : 0.0,
        },
    }
}
```

An asset node entity consists of an asset type name, an asset instance name (UUID across all node types), and a draw-down rate (the percentage by which successful cyber attack against the asset, at any given simulation time-step, reduces the asset’s value).

2.4.5. *JSON schema: attack tree.* The JSON schema for an attack tree instance is as follows: {

```
    “description” : “attack tree instance”,
    “type” : “object”,
    “properties” : {
        ’’tree_name” : {
            “title” : “the UUID for the attack tree”,
            “type” :“string”,
            “required” : true,
            },
        ’’threat_type” : {
            “title” : “the type of threat thread represented by the tree instance”,
            “type” :“string”,
            “required” : true,
            },
        ’’threat_UUID” : {
            “title” : “the name (UUID) of threat thread represented by the tree instance”,
            “type” ” “string”,
            “required” : true,
        },
        “title” : “threat stage vertex specification”,
        “type” : “array”,
        “required” : true,
        “items” : {
```

```
    “type” : “object”,
    “properties” : {
        “threat_stage_name” : {
            “title” : “UUID of threat stage”,
            “type” : “string”’ ,
            “required” : true,
        }
    }
},
“title” : “mitigation vertex specification”,
“type” : “array”,
“required” : true,
“items” : {
    “type” : “object”,
    “properties” : {
        “mitigation_name” : {
            “title” : “UUID of the mitigation”,
            “type” : “string”’ ,
            “required” : true,
        }
    }
},
“title” : “access vertex specification”,
“type” : “array”,
“required” : true,
“items” : {
    “type” : “object”,
    “properties” : {
        “access_name” : {
            “title” : “UUID of the access node ’,
            “type” : “string”’ ,
            “required” : true,
        }
    }
},
“title” : “asset vertex specification”,
“type” : “array”,
“required” : true,
“items” : {
    “type” : “object”,
    “properties” : {
        “asset_name” : {
            “title” : “UUID of the asset node ’,
            “type” : “string”’ ,
            “required” : true,
        }
    }
},
‘title” : “edge specification”,
“type” : “array”,
        “required” : true,
        “items” : {
            “type” : “object”,
            “properties” : {
                “source_node_type” : {
                    “title” : “type of the source node”,
                    “enum’ ’ : [“threat_stage”, “mitigation”, “access”] ,
                    “required” : true,
                },
                “source_node_name” : {
                    “title” : “UUID of the source node”,
                    “type” : “string”
                    “required” : true,
                },
                “dest_node_type” : {
                    “title” : “type of the destination node”,
                    “enum’ ’ : [“threat”, “asset”] ,
                    “required” : true,
                },
                “dest_node_name” : {
                    “title” : “UUID of the destination node”,
                    “type” : “string”
                    “required” : true,
                }
                “connectivity_types” : {
                    “title” : “connective types of edge (for access-access edges)”,
                    “type” : “array”, “items” : { “type” : “string” },
                },
            }
        }
    }
}
```

An attack tree is a tree name, threat thread, and type, together with list of the threat thread stages, mitigations, access nodes, and asset nodes that are applicable for the tree, and an edge set specification describing the connections of threat stage, mitigation, and/or access nodes to threat stage nodes and/or asset nodes (and connectivity types, for edges between access nodes). If different stages of a threat apply to the same element of an IT infrastructure, then the element may be represented by as many access nodes as there are stage nodes, with each access node carrying a distinct UUID (but the same name, if so desired). The graph must be directed and acyclic; the stages of a threat must form a path through the graph, and the ordering of the nodes within the past must be compatible with the ordering in the threat thread specification.

2.4.6. *JSON schema: cyber risk network.* The JSON schema for a cyber risk model is as follows: {

```
    “description” : “model instance”,
    “type” : “array”,
        “required” : true,
        “items” : {
            “title” : “attack tree UUID”,
            “type” : “string”,
            “required’ : true
        },
    }
}
```

Finally, the overall input to the CyberV@R model is a set of attack tree instances, as identified by their UUIDs, in which threat stage nodes (as identified by their UUIDs) are unique to an attack tree, and mitigation, access, and asset nodes (as identified by their UUIDs) may be shared across attack trees. These trees will be merged by the model (at the mitigation and asset node level) to represent the overall risk network of the organization for which the CyberV@R is to be computed. It is required that no two nodes are declared with the same UUID within a given attack tree instance.

2.4.7. *Example input.* Following is an example input for a very simple 3-node network under attack from an instance of the Conficker virus [SRI2009], here imagined as carrying a data exfiltration payload. The initial threat stage is targeted (via exploit MS08-067) against an access node PC1, which employs a host-based intrusion detection system (HDS) that is 50% effective against the exploit. The threat then reconfigures a gateway access node (via Universal Plug and Play, or UPnP) to direct the MS08-067 against a second access node, “PC2”, which deploy an network-based intrusion detection system (NIDS). If the gateway disables UPnP, the attack is thwarted. The PC2 node is also defended against MS08-067 via an HDS. Once established on PC2, the threat downloads a (hypothetical) payload which will extract documents on PC2 containing intellectual property valued at $1,000,000. The downloaded may be thwarted by an NIDS device on PC2, which is deemed 50% effective against the threat. If the download succeeds, 5% of the available intellectual property will be lost (at a cost of $50,000). {

```
    'ThreatThread': {
        ‘threat_type’: ‘Conficker.A’,
        ‘thread_name’: ‘Conficker.A1’,
        ‘intent_type’: ‘Command and Control’,
        ‘threat_stages’: [
        {
            ‘stage_type’: ‘initial’,
            ‘stage_name’: ‘initial1’,
            ‘base_lambda’: 48,
            ‘upper_bound’: 100,
            ‘growth’: 1.05,
            ‘mechanisms’: [
                {‘type’: ‘MS08-067’,
                ‘defeats’: None,
                ‘applies_to’: [{‘access_type’: ‘Windows_PC’}],
                }
            ]
        },
        {
            ‘stage_type’: ‘gateway_configuration’,
            ‘stage_name’: ‘gateway_configuration1’,
            ‘base_lambda’: 48,
            ‘upper_bound’: 100,
            ‘growth’: 1,
            ‘mechanisms’: [
                {‘type’: ‘UPnP’,
                ‘defeats’: None,
                ‘applies_to’: [{‘access_type’: ‘Windows_gateway’}],
                }
            ]
        },
        {
            ‘stage_type’: ‘propagation’,
            ‘stage_name’: ‘propagation1’,
            ‘base_lambda’: 48,
            ‘upper_bound’: 100,
            ‘growth’: 1,
            ‘mechanisms’: [
                {‘type’: ‘MS08-067’,
                ‘defeats’: None,
                ‘applies_to’: [{‘access_type’: ‘Windows_PC’}],
                }
            ]
        },
        {
            ‘stage_type’: ‘exploitation’,
            ‘stage_name’: ‘exploitation1’,
            ‘base_lambda’: 48,
            ‘upper_bound’: 100,
            ‘growth’: 1,
            ‘mechanisms’: [
                {‘type’: ‘HTTP’,
                ‘defeats’: None,
                ‘applies_to’: [{‘access_type’: ‘Windows_PC_exploitable’}],
                }
            ]
        }
        ],
        ‘targets’: [{‘asset_type’: ‘IP’}],
        ‘default_rate’: 1,
        ‘default_growth’: 1
    },
    ‘Mitigations’: [
        {
            ‘mitigation_type’: ‘HDS’,
            ‘mitigation_name’: ‘HDS1’,
            ‘mitigates’: [{‘threat_type’: ‘MS08-067’, ‘effectiveness’: .5}],
            ‘applies_to’: [{‘access_type’: ‘Windows_PC’}],
            ‘default_effectiveness’: 0
        },
        {
            ‘mitigation_type’: ‘Disable_UPnP’,
            ‘mitigation_name’: ‘Disable_UPnP1’,
            ‘mitigates’: [{‘threat_type’: ‘UPnP’, ‘effectiveness’: 1.0}],
            ‘applies_to’: [{‘access_type’: ‘Windows_gateway’}],
            ‘default_effectiveness’: 0
        },
        {
            ‘mitigation_type’: ‘NIDS’,
            ‘mitigation_name’: ‘NIDS1’,
            ‘mitigates’: [{‘threat_type’: ‘HTTP’, ‘effectiveness’: .5}],
            ‘applies_to’: [{‘access_type’: ‘Windows_PC’}],
            ‘default_effectiveness’: 0
        }
    ],
    ‘AccessNodes’: [
        {
            ‘access_type’: ‘Windows_PC’,
            ‘access_name’: ‘PC1’
        },
        {
            ‘access_type’: ‘Windows_gateway’,
            ‘access_name’: ‘Gateway1’
        },
        {
            ‘access_type’: ‘Windows_PC’,
            ‘access_name’: ‘PC2’
        },
        {
            ‘access_type’: ‘Windows_PC’,
            ‘access_name’: ‘PC2_exploitable’
        }
    ],
    ‘AssetNodes’: [
        {
            ‘asset_type’: ‘IP’,
            ‘asset_name’: ‘SecretFormula’,
            ‘initial_value’: 1000000,
            ‘drawdown_rate’: .95
        }
    ],
    ‘AttackTree’: [
        {
            ‘tree_name’: ‘tree1’,
            ‘threat_type’: ‘Conficker.A’,
            ‘threat_UUID’: ‘Conficker.A1’,
            ‘threat_stage_vertex_spec’: [‘initial1’, ‘gateway_configuration1’,
                ‘propagation1’, ‘exploitation1’],
            ‘mitigation_vertex_spec’: [‘HDS1’, ‘Disable_UPnP1’, ‘NIDS1’],
            ‘access_vertex_spec’: [‘PC1’, ‘Gateway1’, ‘PC2’, ‘PC2_exploitable’],
            ‘asset_vertex_spec’: [‘SecretFormula’],
            ‘edge_spec’: [
            {
                ‘source_node_type’: ‘threat_stage’,
                ‘source_node_name’: ‘initial1’,
                ‘dest_node_type’: ‘access’,
                ‘dest_node_name’: ‘PC1’
```

```
},
{
    ‘source_node_type’: ‘mitigation’,
    ‘source_node_name’: ‘HDS1’,
    ‘dest_node_type’: ‘access’,
    ‘dest_node_name’: ‘PC1’
},
{
    ‘source_node_type’: ‘access’,
    ‘source_node_name’: ‘PC1’,
    ‘dest_node_type’: ‘threat_stage’,
    ‘dest_node_name’: ‘gateway_configuration1’
},
{
    ‘source_node_type’: ‘threat_stage’,
    ‘source_node_name’: ‘gateway_configuration1’,
    ‘dest_node_type’: ‘access’,
    ‘dest_node_name’: ‘Gateway1’
},
{
    ‘source_node_type’: ‘mitigation’,
    ‘source_node_name’: ‘Disable_UPnP1’,
    ‘dest_node_type’: ‘access’,
    ‘dest_node_name’: ‘Gateway1’
},
{
    ‘source_node_type’: ‘access’,
    ‘source_node_name’: ‘Gateway1’,
    ‘dest_node_type’: ‘threat_stage’,
    ‘dest_node_name’: ‘propagation1’
},
{
    ‘source_node_type’: ‘threat_stage’,
    ‘source_node_name’: ‘propagation1’,
    ‘dest_node_type’: ‘access’,
    ‘dest_node_name’: ‘PC2’
},
{
    ‘source_node_type’: ‘mitigation’,
    ‘source_node_name’: ‘HDS1’,
    ‘dest_node_type’: ‘access’,
    ‘dest_node_name’: ‘PC2’
},
{
    ‘source_node_type’: ‘access’,
    ‘source_node_name’: ‘PC2’,
    ‘dest_node_type’: ‘threat_stage’,
    ‘dest_node_name’: ‘exploitation1’
},
{
        ‘source_node_type’: ‘threat_stage’,
        ‘source_node_name’: ‘exploitation1’,
        ‘dest_node_type’: ‘access’,
        ‘dest_node_name’: ‘PC2_exploitable’
      },
      {
        ‘source_node_type’: ‘mitigation’,
        ‘source_node_name’: ‘NIDS1’,
        ‘dest_node_type’: ‘access’,
        ‘dest_node_name’: ‘PC2_exploitable’
      },
      {
        ‘source_node_type’: ‘access’,
        ‘source_node_name’: ‘PC2_exploitable’,
        ‘dest_node_type’: ‘asset’,
        ‘dest_node_name’: ‘SecretFormula’
      }
      ]
    }
  ],
  ‘RiskNetwork’: [‘tree1’]
}
```

2.5. **CyberV@R via Dynamic Bayesian Networks: Attack tree network construction.** As described above, the inputs to the model are a set of attack trees $\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_n$, consisting of distinct threat stage nodes, and possibly shared mitigation, access, and asset nodes. Conceptually, the model will first construct a separate Bayesian network $\mathcal{B}_0(\mathcal{T})$ for each distinct attack tree. It will then combine these these networks into a single Bayesian network $\mathcal{B}_0$ by performing the follow steps:

(1) First, if a threat stage, mitigation, or access node has a UUID y, and appears as y in K distinct attack trees having indices $k_1, k_2, \cdots, k_J$, each appearance will be relabeled as (y, k_i) where k_i is the index of the tree in which the appearance occurs.

(2) Next, $\mathcal{B}_0$ is formed by taking the union of the $\mathcal{B}_0(\mathcal{T})$.

(3) Next, all occurrences of asset node v, across all attack trees, are identified (combined into a single node, with an edge to each access node α such that there was an edge from α to v).

The Bayesian network B_0 is the initial network for the dynamic Bayesian network $\langle \mathcal{B}_0, \mathcal{B}_\rightarrow \rangle$ with transformation $\mathcal{B}_\rightarrow$ given by the conditioning of access nodes $A_k(t)$ on $A_k(t-1)$ and asset nodes $V_l(t)$ on $V_l(t-1)$. Then $\langle \mathcal{B}_0, \mathcal{B}_\rightarrow \rangle$ admits a sampling process by a which the CyberV@R algorithm may be viewed as a map $v : \langle \mathcal{B}_0, \mathcal{B}_\rightarrow \rangle \to \mathbb{R}$, as described in **Algorithm 1**. The algorithm utilizes a forward-sampling routine, an implementation of which may be found in the Python package provided by CyberPoint Labs at [libpgm].

2.6. **CyberV@R via Dynamic Bayesian Networks: Validation approach.** In finance, one would typically evaluate the performance of a VaR model by computing the number of instances, over some fixed period of time, in which the VaR was exceeded [Jor2007]. For example, one would

Algorithm 1 CyberV@R: Random Sampling Approximation for P-percent CyberV@R

```
procedure CyberV@R($N$ = number of trials,
                   $T$ = number of time steps,
                   $P$ = desired percent V@R,
                   $\mathcal{C} = \langle \mathcal{B}_0, \mathcal{B}_\rightarrow \rangle$ = union of all attack trees)
    for $n = 0 \ldots (N-1)$ do
        output = [ ]                                                ▷ an array
        evidence = {}
        for $t = 0 \ldots (T-1)$ do
            sample ← randomsample($\hat{\mathcal{B}}_t$, evidence)           ▷ a list of {key: value} pairs
            for accessnode in sample do
                if sample[accessnode] = ‘accessed’ then                     ▷ key indexing
                    if (parent threat node).transit = false then
                        evidence ← evidence ∪ {access node: ‘accessed’}
                    end if
                end if
            end for
            for assetnode in sample do
                if ∃ parent of assetnode such that sample[parent] = ‘accessed’ then
                    assetnode.value$_t$ ← assetnode.value$_{t-1}$ · assetnode.drawdown_rate
                end if
            end for
            for threatnode in sample do
                if ∄ parent of threatnode or sample[parent] = ‘accessed’ then
                    threatnode.lambda$_t$ ← threatnode.lambda$_{t-1}$ + threatnode.growth
                end if
            end for
        end for
        total_loss = $\sum_{\text{assetnodes}}$ (assetnode.value$_0$ − assetnode.value$_T$)
        output ← output ∪ total_loss
    end for
    output ← sorted output
    return output[$\lceil N(1 - \frac{P}{100}) \rceil$] ▷ the value $V$ s.t. odds of losing more than $V$ over $T$ are $\leq P$.
end procedure

function randomsample($\hat{\mathcal{B}}_t$, evidence)
    sample = {}
    for node in $\hat{\mathcal{B}}_t$ do
        if node is in evidence then
            sample = sample ∪ {node: evidence[node]}
        else
            $\rho$ ← probability density function for node, conditioned on outcomes of parents
            outcome ← pseudorandom draw from possible node outcomes, according to $\rho$
            sample = sample ∪ {node: outcome}
        end if
    end for
    return sample
end function
```

expect a daily 98-percent VaR, over the course of a 250-day business year, to be exceeded roughly five times. Occurrences significantly beyond this may call into question the assumptions of the VaR model. One may adopt Chi-square tests as described, for example in [Press07], to quantify the level of plausibility of the model.

For the realm of cyber security, however, unlike finance, it is not so easy to ascertain the actual number of times the CyberV@R is exceeded, for a fixed period of time. Losses may go undetected. Even if they are detected, in the cases of PII or IP theft, it may be years before the actual cost of the loss is known. In the credit realm, there is precedent for fixing the unknown future recovery rate of a defaulted bond, for the purposes of modeling the value of a credit default swap. The prices quoted by the models may then not be accurate in the absolute sense, but are still deemed accurate relative to each other, when calibrated to the same fixed recovery rate. A similar approach may work best for CyberV@R validation; assuming that any data breach entails a fixed cost per datum will allow two models to be compared for accuracy relative to each other, calibrated to those fixed loss rates.

2.7. **Optimizing CyberV@R.** Given a CyberV@R Bayesian network $\mathcal{C}$ as defined in section 2.5, one may consider a set of graph transformations $M_1, M_2, \cdots, M_n$, that map $\mathcal{C}$ to a new dynamical Bayesian network $\mathcal{C}_1 = M_1(\mathcal{C}), \mathcal{C}_2 = M_2(\mathcal{C}), \cdots, \mathcal{C}_n = M_n(\mathcal{C})$. These transformations might represent additional mitigations, a re-arrangement of the network topology, a re-arrangement of asset locations, or some combination of all three. Let v be the result of applying the P-percent **CyberV@R** algorithm to $\mathcal{C}$, for fixed P, N, and T, and let v_i be the result of applying **CyberV@R** to $\mathcal{C}_i$ for the same P, N, and T. Then for each i one can form the difference $\delta_i = v - v_i$. The M_i resulting in the maximal δ_i is then the optimal transformation in the set of transformations considered.

Or, one may plot the M_i, ordered by cost of implementation, against the δ_i to obtain a "rate of benefit" graph against expenditure levels.

Finally, given a fixed cost, an initial network topology $\mathcal{C}$ with mitigations, and a universe of possible threats and possible mitigations, with accompanying constraints as described in 2.5, once can consider using simulated annealing techniques such as those described in [Kirk1983] to find an optimal transformation M, with respect to ensuing the reduction in **CyberV@R**. A representative set of fundamental operations (to be randomly selected at each step of the annealing process) is given by:

(1) The replacement of a mitigation ξ_i with a mitigation ı_j.

(2) The addition or removal of a mitigation ξ_i with respect to a node n_i.

(3) The relocation of an asset node v_i from an access node a_j to an access node a_k.

(4) The insertion or removal of an access node a_k between two asset nodes, or an access and asset node.

(5) The introduction of an applicable threat node t_i to an access node a_j.

2.8. **Automated graph construction and maintenance.** Rather than requiring as input an explicit definition of each attack tree, it would be desirable for users to be able to specify separately the threat threads of interest on the one hand, and on the other, the IT infrastructure as a directed acyclic graph of access nodes, mitigations deployed to the nodes, and assets housed on the access nodes, and let the system generate the attack trees. This amounts to the user eliding, from the definition of an attack tree given in 2.4.5, any edges including threat stage nodes. The system may then complete the attack trees by introducing all admissible threat node edges, inserting copies of access nodes as required, if the same threat thread contains multiple threat stage nodes applicable to the access node, such that the edges follow the construction rules of 2.3.

We present below **Algorithm 2** for performing this automated construction. The algorithm assumes the user has provided a network topology $\mathbf{N}$, consisting of a vertex set V (including access and asset nodes) and one edge set E_c for each mode c of cyber connectivity (ethernet, USB, wireless, bluetooth, etc.) (yielding a restricted topology N_{E_c}). We further assume the user has provided a set of mitigations M, and a set of threat threads T. Each $T_i \in T$ consists of an ordered set of threat stages $s_j,\ j = 1, \cdots, k$. Recall each threat stage s_j employs one or more mechanisms of action m_{i_j}, each of which in turn specifies a method of connectivity and a set of applicable access nodes, with attack surface qualifiers.

2.9. **Attack Tree Construction: an example.** Consider a five-stage threat thread T_X of the following form:

s_1 : initial stage
s_2 : (first) persistence stage
s_3 : propagation stage (implies persistence; assume here repeatable and skippable)
s_4 : exploitation stage (against final target)
s_5 : exfiltration stage.

The algorithm starts by searching for a node $n \in \mathbf{N}$ to which s_1 applies. On any given attack path (there may be multiple within an attack tree) the next threat stage must be s_2, followed by any number of instances of s_3, and finally by s_4 and s_5. Each stage must attack an applicable node in $\mathbf{N}$, so given a threat stage s_i, with parent a_i, the algorithm attempts to find all attackable nodes $n \in \mathbf{N}$ for which s_i may be a parent, under the rules of 2.3 and 2.4. For initial, persistence, exploitation, or exfiltration stages, the sole attackable node is simply a new copy of a_i itself (with a distinct UUID). For propagation stages, attackable nodes must meet two conditions: they must be of the type that the threat stage applies to, and they must be in the 'neighborhood' of n. We define neighborhood as follows: for each of a threat stage s_j's connectivity types, a set we call $C(s_j)$, there exist a set of children of n (because we have different network topologies $N|_{E_c}$ for each connectivity type $c \in C(s_j)$). We can define a set $\nu(n, s_j)$, the neighborhood of n with respect to s_j, as follows:

$$\nu(n, s_j) = \bigcup_{c \in C(s_j)} \mathrm{Ch}_{N_{E_c}}(n)$$

where $\mathrm{Ch}_{N_i}(n)$ refers to the children of n on network $N|_{E_c}$. The algorithm searches this set for matches, and for each node $x \in \nu(n, s_j)$ that the threat stage applies to, the algorithm establishes a connection between s_i and x and continues recursively on s_j and x, and also on s_{j+1} and x, where s_{j+1} denotes the first applicable successor stage to s_j for x.

Let us say for our example that our entire threat thread operates with mechanisms that propagate via ethernet cable, and that our network topology $N|_{E_{\mathrm{ethernet}}}$ looks like this:

$$A \leftrightarrow B \leftrightarrow C \leftrightarrow D \leftrightarrow E$$

The algorithm iterates through the nodes of $\mathbf{N}$ and finds, say (in our example), that s_1 applies to A. Since s_2 is not a propagation stage, it then checks if s_2 applies to A. Then it checks the children of A in $N|_{E_{\mathrm{ethernet}}}$, namely $\{B\}$, to see if s_3 applies to any of them (in our example, we'll say it does apply to B). Since s_3 is a propagation stage, the threat now does two things: it attempts to run stage s_4 on B, and it propagates again via another stage of s_3 to C. Our attack tree splits with

Algorithm 2 Semi-automated attack tree construction

```
procedure ConstructAttackTree($\mathbf{N}, M, T$)
    result ← [ ]                                                ▷ init result tree array
    (global) counter ← 1                          ▷ init global to ensure UUIDs across trees
    for $T_i$ in $T$ do
        (global) tree ← $G(V = \{\}, E = \{\})$
        (global) $S_T$ ← threat stages of $T_i$
        (global) propagated ← [ ][ ]       ▷ $|S_T| \times |N|$ boolean array; reset for each tree
        for $i = 1$ to $|S_T|$ do
            for $n$ in $\mathbf{N}$ do
                propagated[$i$][$n$] ← false
            end for
        end for
        tree[$V$] ← tree[$V$] ∪ ⟨$S_T[1]$, counter⟩                            ▷ add root
        counter ← counter + 1
        for $n$ in $\mathbf{N}$ do                                      ▷ build tree recursively
            if $S_T[1]$ applies to $n$ then          ▷ for specification of 'applies to', see (2.4.3)
                AddEdge($S_T[1], n$)
                RunStage(stage + 1, $n$)
            end if
        end for
        valid ← ProcLeaves(tree[$V$], $\mathbf{N}$)                    ▷ ensure tree has a purpose
        if valid != true then
            break
        end if
        for $e(u, v)$ in tree[$E$] do                                         ▷ add mitigations
            if $u$ is threat stage then
                mit_set ← {}
                for mech in mechanisms of $u$ do
                    for mitigation in $M$ do
                        if mitigation applies to $v$ and mitigates mech then
                            mit_set[mech] ← mit_set[mech] ∪ mitigation
                        end if
                    end for
                end for
                $e$ ← effectiveness calculated by equation (9) on mit_set
                $m$ ← a mitigation node with effectiveness $e$
                tree[$V$] ← tree[$V$] ∪ ⟨$m$, counter⟩
                tree[$E$] ← tree[$E$] ∪ (⟨$m$, counter ⟩, $v$)
                counter ← counter + 1
            end if
        end for
        result = result ∪ tree
    end for
    return result                                                          ▷ final output
end procedure
```

```
function RunStage(stage, $n$)                                         ▷ run stage from $n$
    AddEdge($n, S_T$[stage])
    if $S_T$[stage].type = ‘propagate’ then
        for $m$ in $\nu(n$, stage) do
            if propagated[stage][$m$] = false and $S_T$[stage] applies to $m$ then
                propagated[stage][$m$] ← true                         ▷ prevent infinite propagation
                AddEdge($S_T$[stage], $m$)
                RunStage(stage, $m$)
            end if
        end for
        if stage < $|S_T|$ then                                       ▷ skip propagation stage on local node
            RunStage(stage + 1, $n$)
        end if
    else                                                              ▷ run non-prop stages locally
        if stage < $|S_T|$ and $S_T$[stage] applies to $n$ then
            AddEdge($S_T$[stage], $n$)
            RunStage(stage + 1, $n$)
        end if
    end if
end function

function AddEdge(source, destination)                                 ▷ add vertex and edge to vertex
    tree[$V$] ← tree[$V$] ∪ ⟨destination, counter⟩
    tree[$E$] ← tree[$E$] ∪ (⟨source, counter - 1⟩, ⟨destination, counter⟩)
    counter ← counter + 1
end function

function ProcLeaves(leaves, $\mathbf{N}$)                             ▷ ensure that an asset is attacked
    valid ← false
    for leaf in leaves do
        for  $n$ in asset nodes of $\mathbf{N}$ do
            if (leaf, $n$) $\in \mathbf{N[E]}$ and $n$.type = $T_i$.type then
                valid ← true
            end if
        end for
    end for
    return valid
end function
```

these two options, and two separate attack paths are formed. Once s_3 has propagated through all possible nodes and s_4 and s_5 have been attempted on each node reached by s_3, this process terminates.

Note that this process could find an exponential number of attack paths in $\mathbf{N}$ since multiple neighbor nodes may have an applicable node type, and propagation stages s_j may have multiple mechanisms and can thus have multiple applicable node types. Every time n propagation target nodes are found, there is an $n+1$-way split in the attack tree. Hence this step of the algorithm takes the form of a graph exploration, constructing all valid attack trees in runtime $O(|E|^{|E|})$, where $|E|$ is the number of edges in the graph. In our example, this process constructs the following tree (the indices denote the threat stage, if applicable, and the vertex number):

$$\begin{tikzcd}
A_{:4} \arrow[d] & s_{2:3} \arrow[l] & A_{:2} \arrow[l] & s_{1:1} \arrow[l] & & \\
s_{3:5} \arrow[r] \arrow[d] & A_{:6} \arrow[r] & s_{4:7} \arrow[r] & A_{:8} \arrow[r] & s_{5:9} \arrow[r] & A_{:10} \\
B_{:11} \arrow[r] \arrow[d] & s_{4:12} \arrow[r] & B_{:13} \arrow[r] & s_{5:14} \arrow[r] & B_{:15} & \\
s_{3:16} \arrow[d] & & & & & \\
C_{:17} \arrow[r] \arrow[d] & s_{4:18} \arrow[r] & C_{:19} \arrow[r] & s_{5:20} \arrow[r] & C_{:21} & \\
s_{3:22} \arrow[d] & & & & & \\
D_{:23} \arrow[r] \arrow[d] & s_{4:24} \arrow[r] & D_{:25} \arrow[r] & s_{5:26} \arrow[r] & D_{:27} & \\
s_{3:28} \arrow[d] & & & & & \\
E_{:29} \arrow[r] & s_{4:30} \arrow[r] & E_{:31} \arrow[r] & s_{5:32} \arrow[r] & E_{:33} &
\end{tikzcd}$$

2.9.1. *Guaranteeing the tree reaches an intended asset.* The algorithm checks that there exists an edge from one or more of the nodes in $\mathbf{N}$ that has been attacked by s_5 (the exfiltration stage) to an asset node of value. The algorithm further checks that these asset nodes have the type that the threat thread is designed to attack. If either of these conditions is not met, the attack tree is discarded. (So in our example, we've assumed that there is an edge from node E, say, to an asset node α, and that the type of α is the type that our threat thread targets.) The runtime of the validation step is $O(|V||\alpha|)$ where $|V|$ is the number of vertices in the attack tree and $|\alpha|$ is the maximum number of assets reached by an individual access node.

2.9.2. *Adding mitigations.* Recall each mitigation in the user input is stored with a list of access node types to which it applies, and a list of threat stage node types that it mitigates. The algorithm searches through the existing [threat node → asset node] graph edges in the attack tree. For each of these edges, there is a set of mechanisms $\{t_1, \ldots, t_n\}$ via which the threat node could attack the access node. If there exists an edge $e = (u, v)$ such that one or more mitigations applies to v and mitigates a subset of $\{t_1, \ldots, t_n\}$, then that set of mitigations is combined and appended to the graph, as a single mitigation vertex m, with effectiveness e calculated as:

$$e = \prod_{j \in \text{mech. of } u} \left[\begin{array}{cc} 1 - (1 - M_{j1}) \cdots (1 - M_{jn}) & \text{if j has mitigations } M_{j1}, \ldots, M_{jn} \\ 0 & \text{otherwise} \end{array} \right] \tag{9}$$

(Note that if a mechanism j has no mitigations, e is forced to 0). In addition, an edge is constructed from the mitigation node m to v. The runtime of this step is $O(|E||M|)$, where $|E|$ is the number of edges in the attack tree and $|M|$ is the number of mitigations in user's system.

3. Real-World Considerations

As described above in section 2.4, the CyberV@R model accepts as input a set of attack trees, with nodes representing threat thread stages, mitigations for the threat stages, access nodes that are the targets of the threat stages and the beneficiaries of the mitigations, and asset nodes to

Type / Example	Access Stage	Mechanisms	objective
SABOTAGE	initial	unauth, infected removable device	breach perimeter
Stuxnet	propagation	0-day Win O.S. vuln OR	infect internal subnets
		2-yr Step 7 app vul.	
	persistence	open TCP/IP ports	establish C&C
	exploitation	infected removable device	alter Siemens PLC
	evidence	redirect malware block r/w requests	hide presence
ESPIONAGE	initial	targeted malicious email	breach perimeter
Taidoor	persistence	MS Office/Adobe known vuln. OR	drop encrypted trojan
		zip'd exe/dll	
	exploit	injects in-core WinOS exe	establish C&C
	exfiltration	via encrypted HTTP msgs	steal files
IP THEFT	initial	targeted malicious email	breach perimiter
Nitro	exploit	launch exe attachment (Poison Ivy)	establish C&C
	exfiltration	via encrypted HTTP msgs	steal files
USURP CAPABILITY	initial	unknown MS-Windows RPC exploit	gain host access
Conficker.A	propagation	ibid.	infect peers
	persistence	deploy DLL started w/Windows	persist
	propagation	open backdoor through firewall	establish infection route
	exploitation	contact functionally-gen'd domain list	establish C&C

Table 1. Sample high-level attack segments for various threat types.

be defended. Ideally, real-world implementations of the model will not require all these data to be entered by hand for each model instantiation. Rather, sets of pre-defined threat thread inputs should be established for the user, with incidence rates provided and periodically updated by the model provider, perhaps based on data collected via vendor-deployed honeynets and other active and passive research mechanisms. Similarly, mitigation nodes and access nodes, with appropriate decorating qualifiers, should be populated automatically for the user based on the results of penetration tests, security audits, deployed risk monitoring systems, and related mechanisms. In what follows, we present some research directions to guide these automation processes.

3.1. **Templates for risk: the stages of attack.** In [Whit2006], a generic framework for characterizing attacks is defined, which we enumerate here as consisting (at a highest level) of the initial access, persistence (whether permanent or transitory), propagation, exploitation, exfiltration and evidence removal/tampering phases. For any given attack, not all of the stages will apply, not all will be relevant for modeling purposes, and moreover multi-stage attacks may encompass more than one instance of a stage. Any given access stage will have a named objective, and one or more mechanisms of realization (exploit). Each threat has an overall intent, which serves as the main characterizing mechanism (i.e. it is a threat type name for our modeling purposes). In table 1 we summarize the applicable stages for some of the more prominent threat types and instances thereof.

The first set of entries in our table covers SCADA attacks as exemplified by the Stuxnet virus. According to the proposed scenario given in [Sym2011], the attack initially enters the system via insertion of an infected USB, and spreads throughout a local area network (LAN) via (what were once) unknown O.S. vulnerabilities. Once resident in the LAN, it may establish command and control communications. Finally, from one of its persisted locations, it is carried again via USB to a stand-alone ICS system (Programmable Logic Controllers – PLCs – manufactured by Siemens are the specific target), which it can infect in such a manner as to sabotage the ICS-managed physical devices that are the ultimate objective of the exploit. Additional sample entries are based on information provided in [CWG2011], [SRI2009], [Sym2011], [Sym2012].

To model this attack (in simplified form), three access nodes are required; one representing an initial access point (denote it as A_0), a node that collectively represents all the possible LAN-based targets (A_1), and one that collectively identifies all stand-alone ICS computers (A_2). The risk to which A_0 and A_2 are exposed is essentially that of unauthorized device introduction, and the threat to which A_1 is exposed in essentially that of O.S.-level 0-day attacks. Hence an operational model incorporating this threat would require estimated incident rates for each of these attack types, for a SCADA-dedicated host system. The A_0 and A_2 nodes would each have a parent R_0 (respectively R_2) node representing this risk, and the A_1 node would have a risk R_1 representing the estimated 0-day O.S.-level vulnerabilities permitting unauthorized executable file propagation. Finally, the A_2 node would be attached to a V_2 asset node, representing the value of the controlled industrial system (or more accurately, the dollar value of the damage its destruction would cause).

An operational CyberV@R model for an organization deploying SCADA systems would have these node types available as part of a model-construction library, with incident rates maintained via threat provider feeds, or constants set according to historical data for the organization's industry (note it may be operationally beneficial to model only OS types, rather than specific versions, with the number of versions of the OS behind the current version reflected in the attendant threat incident rates). Ideally the nodes would be selected and wired together, in Lego-like fashion, with the help of an authoring tool. Of course an organization will be subject to more than one type of threat, and the model upon instantiation will combine the disparate specified threats into an overall Bayes network. A predetermined set of (access stage, {conditions}) pairs will allow threat threads and stages to be matched to a fixed set of relevant possible mitigations. Hence the library will also need to include relevant mitigation nodes; these are discussed in the next subsection.

3.1.1. *Recommendations for intrinsic threat stage support.* Based on the aforegoing observations, we suggest the following threat thread types be supported as part of any model implementation: *infrastructure sabotage , espionage, IP-targeted theft, PII-targeted theft , economic fraud, information fraud, denial of service, usurpation of capability.*

Moreover, we suggest the following standard threat stage types be supported as part of any model implementation:

(1) *Reconnaissance*: any exploration of a targeted IT infrastructure required to support a subsequent stage of attack (e.g. a port scan).

(2) *Initial access*: the initial point of entry of an attack into the IT infrastructure of an organization (e.g. delivery of shell code to an initially targeted workstation).

(3) *Persistence*: the establishment of a "beachhead" within the IT infrastructure (e.g. establishment of persistent malware on an initially targeted workstation).

(4) *Beaconing*: communication back to a command and control server from a comprised machine (e.g. to establish a command channel, or to receive updated exploits).

(5) *Propagation*: spread from an initially infected machine to other hosts within the network.

(6) *Transit*: transit from machine A to machine B, leaving A uncompromised.

(7) *Exploitation*: execution of attack against a destination target (e.g. access contents of a word document).

(8) *Exfiltration*: remove data from an IT infrastructure (e.g. copy contents of a word document back to a command and control server).

Each stage should be able to draw from the following universe of mechanisms of action, as applicable for the threat:

(1) Action via improperly configured or unauthorized device.

(2) Action via unauthorized or improperly configured software.

(3) Action via known software vulnerabilities.

(4) Action via unknown software vulnerabilities.

(5) Action via credential theft.

(6) Action via default or guessable credentials.

(7) Action via insufficient authentication.

(8) Action via solicitation.

(9) Action via unknown mechanism.

Ultimately, organizations should establish a hierarchy for mechanisms of action, so that, for example, if the mitigation "continuous software patching" applies to an access node, then the mitigation (given proper adjustments to the rules for applying mitigations to threats) will defeat all threat nodes acting via "known software vulnerability", and so in particular "known SMB vulnerability," and hence "MS08-067" (the Conficker.A exploit) [SRI2009].

Finally, the threat stages should apply to the following access node types: *router, web server, wireless device, application server, file server, database server, workstation*, and physical and virtual clusters (subnets), thereof, possibly qualified by operating system labels, vulnerabilities, and other attack surface identifiers.

3.2. **Incorporating 20 Critical Control scores - the conditional data.** As previously noted, the SANS Institute's "20 Controls" document, supplemented by the material in [NIST2011B], [NIST2007], offers sources for possible types of mitigations. We enumerate here for illustrative purposes a sample set of mitigations that a model implementation might support, with suggested methods for assessing the efficacy of the mitigation (expressed as a Bernoulli variable), and the threat stages to which the mitigations apply.

(1) *Automated Device Inventory Scans*: perform automated, regular scans of all devices connected to the organizational infrastructure. Mitigates: *action via improperly configured or unauthorized device.* Efficacy scoring: 0.98 for hourly scans, 0.8 for daily scans, 0.5 for weekly scans, 0.3 for monthly scans, 0 otherwise. Corresponds to SANS Critical Security Control 1.

(2) *Automated Software Inventory Scans*: perform automated, regular scans of all software deployed to the organizational infrastructure. Mitigates: *action via improperly configured or unauthorized software.* Efficacy scoring: 0.98 for hourly scans, 0.8 for daily scans, 0.5 for weekly scans, 0.3 for monthly scans, 0 otherwise. Corresponds to SANS Critical Security Control 2.

(3) *Secure Configurations for Hardware and Software*: Enforce deployment of hardware and software systems using standardized, hardened configuration settings. Mitigates: *action via improperly configured or unauthorized device, action via unauthorized or improperly*

configured software, action via known software vulnerabilities, action via default or guessable credentials. Efficacy scoring: percentage of devices on access node for which standardized configurations are deployed. Corresponds to SANS Critical Security Control 3.

(4) *Security Training*: Annually train users on hygienic computer use practices. Mitigates: *action via default or guessable credentials, action via solicitation* . Efficacy score: 0.98× percentage of users trained annually, of those who access the defended node. Corresponds to SANS Critical Security Control 9.

(5) *Continuous Vulnerability Remediation*: Use automated software patch tracking and deployment tools. Mitigates: *action via known software vulnerabilities.* Efficacy score: $\max(0.98 - (0.1 \times \text{average number of lag days between patch issuance and deployment}), 0)$. Corresponds to SANS Critical Security Control 4.

(6) *Boundary Defense*: Secure external face of organizational infrastructure from suspect traffic via network intrusion detection and prevention systems, packet logging and inspection regimes, and insertion of proxy servers between internal networks and external network. Mitigates: *action via unknown software vulnerabilities, action via solicitation.* Efficacy score: $0.249 \times N$ where N is the number of solutions used from the following list: *Intrusion Detection Systems, Intrusion Prevention Systems, Packet Logging, Proxy Servers.* Corresponds to SANS Critical Security Control 13.

(7) *Application Firewalls*: Front internal server with application-specific firewalls that block unauthorized / irrelevant traffic. Close unnecessary ports and disable unnecessary services. Perform scans to detect unauthorized traffic attempts. Mitigates: *action via unauthorized or improperly configured software, action via default or guessable credentials.* Efficacy score: $Max(D, 0)$ where D is calculated as $\max(0.98 - \frac{H}{24}, 0)$, where H is the average number of elapsed hours prior to detection of an unauthorized port or application access event. Corresponds to SANS Critical Security Control 11.

(8) *Access Control Lists*: Restrict access to data via access control list-based permission schemes. Mitigates: *action via insufficient authentication* . Efficacy score: $min(0.999, 0.3 * M)$ where M is the number of factors used in the authentication.

3.3. **Automated discovery of model parameters.** The input to the **CyberV@R** algorithm of section 2.5 consists of a set of attack trees, with nodes drawn from specifications of (1) cyber threats to which an organization is subject, (2) a network topology, (3) the security mitigations in place to defend the organization against the cyber threats, and (4) the location of defended intellectual property and other protected information within the network. Penetration tests and configuration assessments using tools such as [Metasploit],[Nessus] may conceivably be harnessed to automate the process of accumulating these specifications, including automated assessment of mitigation efficacy. Honeypots [Prov2008] may provide a means of objectively assessing the actual base incidence and growth rates for the threats that are modeled. Commercial tools such as those provided by [McAfee], [Rapid7], [SOURCEfire] may also provide opportunities to automate the process of mitigation discovery, and to maintain mitigation specifications as the organizational network topology changes. Security activity data streams from organizations such as US-CERT [US-CERT] and NIST [NVD] may be used to track vulnerability discovery against patch issuance rates for various threat types, for the purpose of gauging threat incidence and patch mitigation effectiveness rates. Ideally, software systems incorporating and instantiating the **CyberV@R** algorithm will incorporate these feeds within modules supporting the automated creation and manipulation of model specifications. Users will benefit further if such systems also provide means for graphical manipulation of model inputs, to facilitate "what-if" scenario evaluation.

4. Conclusion

The **CyberV@R** algorithm of section 2.5 presented in this document provides a means for quantitatively measuring the financial value at risk (VaR) to an organization, stemming from losses of PII, IP, and similar protected information, or from losses to system capability, due to cyber-related threats. Drawing on and adapting established modeling practices from quantitative finance, it supports the modeling of the dynamic behaviors of threats evolving over time. Its core purpose is to support the quantitative comparison of the *business value* of alternative approaches to strengthening an organizational IT infrastructure against cyber threats. Avenues for future research include the harnessing of commercial and open source tools for the automated construction of the dynamical Bayesian networks that are the input to the model, exploration of tools for the graphical depiction and manipulation of the models, and techniques for constructing optimal IT configurations over the space of possible configurations and their attendant risk profiles. It is the authors' hope that this paper will help stimulate additional research focused on the quantitative characterizations of the financial risks posed to organizations by cyber-related threats.

References

[ANSI2010] American National Standards Institute (ANSI) *"The Financial Management of Cyber Risk"* , available at: http://publicaa.ansi.org/sites/apdl/khdoc/Financial+Management+of+Cyber+Risk.pdf, accessed March 30, 2012.

[ANSI2012] ANSI, *The Financial Impact of Breached Protected Health Information: A Business Case for Enhanced PHI Security*, available at: webstore.ansi.org/phi, accessed March 30, 2012.

[ArcSight] http://www.arcsight.com, accessed April 4, 2012.

[Bart1996] R. Bartoszynski and M. Niewiadomska-Bugaj, *Probability and Statistical Inference*, Wiley Interscience, New York: 1996.

[Brenner2011] J. Brenner, *America the Vulnerable*, Penguin Press, New York: 2011.

[CAPEC] *Common Attack Pattern Enumeration and Classification*, release 1.7, http://capec.mitre.org.

[Clarke2010] R. Clarke, *Cyber War: The Next Threat to National Security and What to Do About It*, Harper-Collins e-books: 2010.

[CWG2011] Conficker Working Group (CWG) *Conficker Working Group Lessons Learned*, January 2011, available at:
http://www.confickerworkinggroup.org/wiki/uploads/
Conficker_Working_Group_Lessons_Learned_17_June_2010_final.pdf,
accessed March 30, 2011.

[CWE] *Common Weakness Enumeration*, version 2.1, http://cwe.mitre.org.

[DHS2011] Department of Homeland Security (DHS), *Information Technology Sector Risk Management Strategy for the Provide Domain Name Resolution Services Critical Function*, available at: http://www.dhs.gov/xlibrary/assets/it-sector-risk-management-strategy-domain-name-resolution-services-june2011.pdf, accessed March 30, 2012.

[DSH2009] DHS, *Information Technology Sector Baseline Risk Assessment*, available at: http://www.dhs.gov/xlibrary/assets/nipp_it_baseline_risk_assessment.pdf, accessed March 30, 2012.

[DOE2012] Department of Energy (DOE) *Electricity Subsector Cybersecurity: Risk Management*, available at: http://energy.gov/sites/prod/files/RMP
%20Guideline%20Second%20Draft%20for%20Public%20Comment%20-
%20March%202012.pdf, accessed March 30, 2012.

[EOPOTUS] Executive Office of the President of the United States, *The Comprehensive National Cybersecurity Initiative*, available at http://www.whitehouse.gov/sites/default/files/cybersecurity.pdf, accessed February 9, 2012.

[Glas2004] P. Glasserman, *Monte Carlo Methods in Financial Engineering*, Springer, New York: 2004.

[Gor2004] L. Gordon and M. Loeb, "The Economics of Information Security Investment," *ACM Transactions on Information and System Security*, Vol 5., No. 4, pp:4387457, November 2002. of Information Security, 2004, Springer, Camp and Lewis, eds.

[Grovitz1012] L. G. Grovitz, "Cybersecurity 2.0", *The Wall Street Journal*, February 27, 2012, available at: http://online.wsj.com/article/SB10001424052970203918304577243423337326122.html, accessed April 3, 2012.

[Hull2000] J. Hull, *Options, Futures, and Other Derivatives*, 4th ed, Prentice Hall, Upper Saddle River, NJ: 2000.

[Huw2008] J. Huwatashi, "Enhancing Cyber Risk Management with the Framework of ERM and Basel II," in *Cyberlaw for Global E-business: Finance, Payments and Dispute Resolution* , ed. T. Kubota, IGI Gloabal, Hershey PA: 2008.

[IEEE2010] IEEE Computer Society, *IEEE Standard 802.1X-2010*, available at: http://standards.ieee.org/getieee802/download/802.1X-2010.pdf.

[Jakob2008] M. Jakobsson, Z. Ramzan, *Crimeware: Understanding New Attacks and Defenses*, Addison-Wesley Professional (on-line vesion): 2008.

[JSON] http://www.json.org/, accessed July 23, 2012.

[JSONSchema] https://github.com/kriszyp/json-schema.

[JSONTutorial] http://nico.vahlas.eu/2010/04/23/json-schema-specifying-and-validating-json-data-structures/.

[Jor2007] P. Jorion, *Value at Risk: The New Benchmark for Managing Financial Risk*, 3rd. ed, McGraw-Hill, New York: 2007.

[JPM1996] Morgan Guaranty Trust Company, *Risk Metrics Technical Document*, 4th ed, 1996, available at: http://www.msci.com/resources/research_papers/, accessed March 30, 2012.

[Kirk1983] S. Kirkpatrick, C.D. Gellat, M.P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol 220, pp. 671-680, 1983.

[Kol2009] D. Koller and F. Friedman, *Probabilistic Graph Models: Principles and Techniques*, MIT Press, Cambridge MA: 2009.

[KL1951] S. Kullback and R.A. Leibler, "On Information and Sufficiency", *Annals of Mathematics*, vol 22, no. 1, 79-86.

[Lando2004] D. Lando, *Credit RIsk Modeling,Theory and Practice*, Princeton UP, Princeton: 2004.

[libpgm] http://packages.python.org/libpgm/.

[LLOYDS2011] , LLoyds, *Lloyd's Risk Index 2011*, available at: http://www.lloyds.com/News-and-Insight/Risk-Insight/Lloyds-Risk-Index, accessed March 30, 2012.

[McAfee] http://www.mcafee.com/us/, accessed April 4, 2012.

[Metasploit] D. Kennedy, et. al. , *Metasploit: the Penetration Tester's Guide*, No Strach Press: Safn Francisco: 2011.

[Nessus] http://www.tenable.com/products/nessus.

[NIST2011] National Institue of Standards and Technology (NIST), U.S. Department of Commerce, *Managing Information Security Risk: Organization, Mission, and Information System View*, NIST Special Publication 800-39, available at: http://csrc.nist.gov/publications/nistpubs/800-39/SP800-39-final.pdf, accessed March 30, 2012.

[NIST2011B] NIST, *The Technical Specification for the Security Content Automation Protocol (SCAP): SCAP Version 1.2* Special Publication 800-126 Revision 2, available at: http://csrc.nist.gov/publications/nistpubs/800-126-rev2/SP800-126r2.pdf, accessed March 31, 2012.

[NIST2009] NIST, *Recommended Security Controls for Federal Information Systems and Organizations* NIST Special Publication 800-53 Rev 3, available at:

http://csrc.nist.gov/publications/nistpubs/80053Rev3/sp80053rev3-final.pdf, accessed March 30, 2012.

[NIST2007] P. Mell, K. Scarfone, NIST/ S. Romaonsky, Carnegie Mellon University, *CVSS: A Complete Guide to the Common Vulnerability Scoring System Version 2.*, June 2007, available at: http://www.first.org/cvss/cvss-guide.pdf, accessed April 4, 2012.

[Numpy] The NumPy package for scientific computing in Python, "http://numpy.scipy.org/", accessed February 9, 2012.

[NVD] National Vulnerabilities Database, version 2.2, available at: http://nvd.nist.gov/home.cfm/

[OVAL] *Open Vulnerability and Assessment Langauge*, http://oval.mitre.org/

[Peter2012] L Peterson and B. Davie, *Computer Networks: a Systems Approach*, 5th ed, Elsevier, Burlington MA: 2012.

[Pol2012] N. Poolsappasit, et. al., Dynamic Security Risk Management Using Bayesian Attack Graphs, *IEEE Transactions on Dependable and Secure Computing*, Vol. 9, No. 1, January/Februrary 2012.

[Press07] William Press, et. al.,*Numerical Recipes*, 3rd ed, Cambridge UP: Cambridge: 2007.

[Prov2008] N. Provos and T. Holz, *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*, Addison-Wesley: Upper Saddle River NJ: 2008.

[Rapid7] http://www.rapid7.com, accessed April 4, 2012.

[Raft2011] E. Raftopolous, M. Dimitropoulos,"Detecting, Validating, and Characterizing Computer Infections in the Wild," *IMC '11*, ACM: Berlin, 2011.

[Robert2004] C. Robert and G. Casella, *Monte Carlo Statistical Methods*, 2nd. ed, Springer: New York, 2004.

[Rob2010] J. Robinson, A. Hartemik, "Learning Non-Stationary Dynamic Bayesian Networks", *Journal of Machine Learning Research* Vol. 11 (2010) 3647-3680, available at: http://jmlr.csail.mit.edu/papers/volume11/robinson10a/robinson10a.pdf, accessed March 31, 2012.

[Roy2010] A. Roy, et. al., "Cyber Security Analysis using Attack CounterMeasure Trees," in *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, ACM, New York: 2010.

[Salter98] C. Salter, et. al. , "Towards A Secure System Engineering Methodology", www.nspw.org/papers/1998/nspw1998-slater.pdf, accessed February 9, 2012.

[Sans2011] SANS Institute, *"Twenty Critical Security Controls for Effective Cyber Defense: Consensus Audit Guidelines"*, http://www.sans.org/critical-security-controls/, accessed March 29, 2012.

[SEC] U.S. Securities and Exchange Commission, *Questions and Answers about the new "Market Risk" disclosure rules*, July 31, 1997 , available at: http://www.sec.gov/divisions/corpfin/guidance/derivfaq.htm#qvar, accessed June 11, 2012.

[Schneier99] B. Schneier, "Attack Trees", *Dr. Dobb's Journal*, December 1999, available at: http://www.schneier.com/paper-attacktrees-ddj-ft.html, accessed Jun 12, 2012.

[SOURCEfire] http://www.sourcefire.com, accessed April 4, 2012.

[SRI2009] P. Porras, et. al. "An analysis of Conficker's Logic and Rendezvous Points", *SRI Technical Report*, MArch 2009, available at: http://mtc.sri.com/Conficker/, accessed June 25, 2012.

[Sym2009] N. Falliere and E. Chien , *Zeus: King of the Bots*, Symantec Security Response, 2009, available at:
http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/zeus_king_of_bots.pdf, accessed March 30, 2012.

[Sym2011] N. Falliere, L. Murchu, E. Chien, *W32.Stuxnet Dossier*, Symantec Security Response, 2012, available at: http://www.symantec.com/security_response/whitepapers.jsp, accessed May 25, 2012.

[Sym2011B] E. Chien and G. O'Gorman, *The Nitro Attacks: Stealing Secrets from the Chemical Industry*, Symantec Security Response, 2011, available at: http://www.symantec.com/security_response/whitepapers.jsp, accessed June 25, 2012.

[Sym2012] S. Doherty, P. Krysiuk, *Trojan.Taidoor: Targeting Think Tanks*, Symantec Security Response, 2011, available at: http://www.symantec.com/security_response/whitepapers.jsp, accessed June 25, 2012.

[US-CERT] US CERT, http://www.us-cert.gov/.

[Wain2008] M. Wainwright and M. Jordan, "Graphical Models, Exponential Families, and Variational Inference", *Foundations and Trends in Machine Learning* Vol. 1, Nos. 12 (2008) 1305.

[Whit2006] A. Whitaker, D. Newman, *Penetration Testing and Network Defense*, Cisco Press, Indianapolis: 2006.

Disclaimers:
Cyber Point International, LLC is committed to benefiting the security community in its efforts to address cyber theft, cyber terrorism and other security issues. Information in this [paper][article] is being provided to security professionals and other readers as a resource for their consideration with respect to analyzing and preventing cyber threats and security vulnerabilities, and not for any malicious or unauthorized use. THE DATA AND INFORMATION IN THIS WHITE PAPER ARE PROVIDED "AS IS" AND CYBER POINT INTERNATIONAL, LLC DOES NOT REPRESENT OR WARRANT THAT THE INFORMATION IS RELIABLE OR ERROR-FREE. CYBER POINT INTERNATIONAL, LLC SHALL NOT BE LIABLE FOR ANY CLAIMS, EXPENSES, COSTS, LOSSES OR DAMAGES (WHETHER IN CONTRACT, TORT, OR OTHERWISE) ARISING OUT OF OR RELATING TO THE USE OF THE INFORMATION. No use of this information in contravention with applicable laws is permitted. Export or re-export the information to any county may require export license.

About CyberPoint International, LLC

CyberPoint delivers innovative, leading-edge cyber security products, solutions and services to customers worldwide. We discover the threats and vulnerabilities that expose data, systems, and infrastructure to compromise and design defenses that provide critical protection. Our approach is tailored to meet the individual needs of our customers, reduce risks, and to ensure ongoing protection in a world of continuously emerging cyber threats. At CyberPoint, we are always learning, exploring, and looking for new ways to put our knowledge and experience to work. We seek out hard problems, develop new products and solutions, and are driving innovation in the field of cyber security.

CyberPoint is dedicated to advancing the discipline of cyber security through imagination, technical excellence, and an unparalleled passion for our work. Employing world-class engineers, mathematicians, computer scientists, and other industry experts, CyberPoint supports a broad array of commercial and government customers.

621 East Pratt Street, suite 300
Baltimore, MD 21202-3140
phone +1 410 779 6700
www.cyberpointllc.com