Multi-Method Modeling and Analysis of the Cybersecurity Vulnerability Management Ecosystem

Andrew P. Moore, apm@sei.cmu.edu
Allen D. Householder, adh@sei.cmu.edu

The CERT® Division of the Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213
412-268-5465

Abstract

This paper presents modeling and analysis of two critical foundational processes of the cybersecurity vulnerability management ecosystem using a combination of system dynamics and agent-based modeling techniques. The preliminary result from this analysis is that misapplication of either of these foundational processes could contribute to the fragility and risk associated with the many national infrastructures and organizational missions that rely on the Internet. We use data from the CERT Coordination Center that characterizes coordinated vulnerability disclosure for our previous and continuing calibration and validation efforts. Our to-date analysis has identified additional areas for future work: new questions to consider, alternate social cost measures to investigate, and new avenues for validation. While the results of our initial efforts should be viewed as preliminary due to limited calibration and validation, we believe that the approaches used and depth of the modeling and simulation are sufficient to begin to understand key implications of these processes and possible avenues for their improved application in the future.

Keywords: cybersecurity, system dynamics, agent-based modeling, vulnerability management, coordinated vulnerability disclosure, multi-method modeling

1 Introduction

Vulnerability management (VM) is the common term used to describe tasks such as technical cybersecurity vulnerability[1] scanning, patch testing, and deployment (NIST 2013, Caralli 2010). VM practices focus on the positive action of identifying specific systems affected by known (post-disclosure) vulnerabilities and reducing the risks they pose through the application of mitigations or remediation, such as patches or configuration changes (Householder 2017).

® CERT and CERT Coordination Center are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

1 Henceforth, we refer to *technical cybersecurity vulnerabilities* simply as *vulnerabilities*.

VM practices nearly always deal with the output of a set of practices called *Coordinated Vulnerability Disclosure* (CVD). Because many modern products are, in fact, composed of software and hardware components from multiple vendors, any of which may contain vulnerabilities, the CVD process increasingly involves the cooperation of vendors with competing or misaligned priorities (Householder 2017, Thomson 2018). The CVD process involves a number of phases that are generally sequential in time, although they can sometimes occur out of order. The phases are based on the *ISO/IEC 3011 Standard* (ISO 2013), as expanded in *The CERT Guide to Coordinated Vulnerability Disclosure* (Householder 2017), as summarized below:

- **A vulnerability is found** – An individual or group finds a vulnerability in an existing system. The individual or group is referred to as the *finder*. Many finders want the vendor to fix the problem, so they report it to the vendor. Not all finders report though.
- **Vendor awareness** – One or more vendors whose products are affected by the vulnerability may become aware of it, whether through their own testing, via reports from cooperative finders, or as a result of analyzing incidents or malware that exploits it.
- **Analysis and prioritization** – The vendor confirms the report to ensure accuracy before action can be taken and prioritizes reports relative to other work, including other reports.
- **Remediation** – A remediation plan (ideally a software patch, but it could also be other mechanisms) is developed and tested.
- **Public Awareness** – The vulnerability and its remediation plan is disclosed to the public. Public awareness is often facilitated by the inclusion of information about the vulnerability in a vulnerability database such as the National Vulnerability Database operated by NIST (NIST Information Technology Laboratory, 2019) . Vulnerability databases use identifiers, such as the Common Vulnerabilities and Exposures (CVE®) ID, to disambiguate distinct vulnerabilities. Such cataloging can happen regardless of the availability of remediation advice.
- **Deployment** – The remediation is applied to deployed systems.

Figure 1 provides a high-level diagram of the CVD process.

® The CVE and the CVE logos are registered trademarks of The MITRE Corporation.

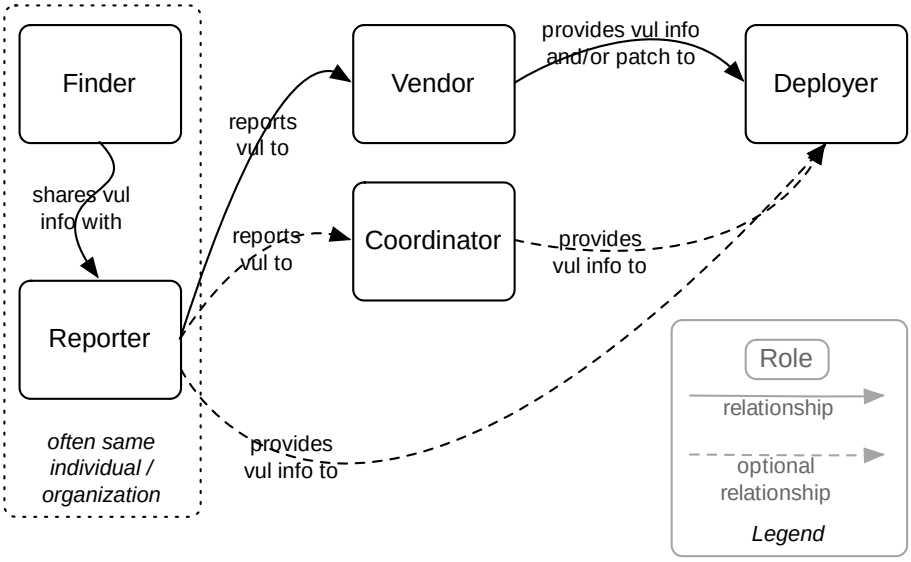

Figure 1: CVD Process (Householder 2017)

This paper presents modeling and analysis of two critical foundational aspects of the cybersecurity vulnerability management ecosystem:

1. the process of identifying and responding to vulnerabilities that most organizations rely on for their security defense (We refer to this as the vulnerability management [VM] process below.)
2. the Multi-Party CVD (MPCVD) process for coordinating the development of software patches across a set of vendors whose products have an identified vulnerability

While our efforts are a work in progress, our preliminary analysis shows that the misapplication of either of these foundational aspects could contribute to the fragility and risk associated with the many national infrastructures and organizational missions that rely on the Internet.

Section 2 uses system dynamics modeling (and the VenSim toolset) to analyze core aspects of the vulnerability management process. Section 3 takes a deeper dive using agent-based modeling (and the Ventity toolset) to analyze the MPCVD process for vulnerability patching and disclosure. The approaches used and the depth of the modeling and simulation are sufficient to understand the key implications of these processes and possible avenues for their improved application in the future. Section 4 summarizes the contributions of this research and directions for the future.

2 System Dynamics Analysis of the Cybersecurity Vulnerability Management Process

2.1 Model Description

This section describes the system dynamics model as a series of incremental builds. Each build is displayed in a separate figure. Sometimes the figure builds on the previous figure. Other figures are components of the larger model, which is displayed in Appendix B for reference. Key parameters of the model, including their types and initial values, are described in Appendix C.

Figure 2 shows a very basic vulnerability discovery and patch process. People discover vulnerabilities (i.e., the vulnerabilities become "known") and initially decide to create patches for some fraction of them. We refer to the vulnerabilities that are not initially targeted for patching as *dormant*. After taking some time to generate and publish vulnerability patches, they are made available to the general public.

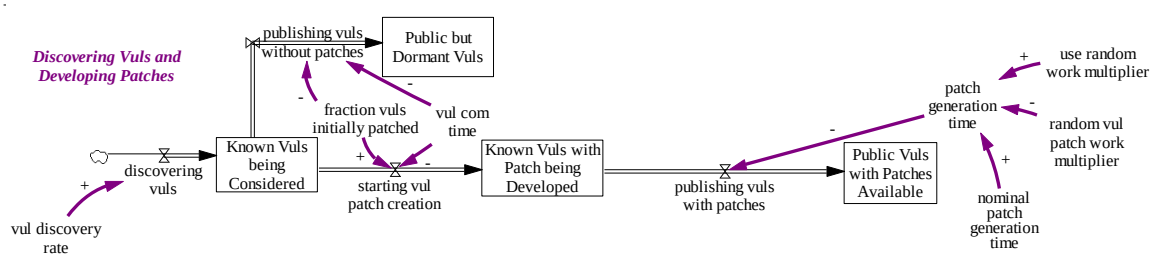

Figure 2: Vulnerability Discovery and Patching Process

This model can run in one of two modes: one that uses an average patch-generation time as constant in the model, and one that uses a random vulnerability patch work multiplier of that average time. Based on previous analysis of the vulnerability work factor, we choose the multiplier from an exponential distribution as shown in Figure 3. The graph shows the multiplier along the x axis (i.e., the y values) and the number of times each multiplier is generated during the simulation along the y axis.

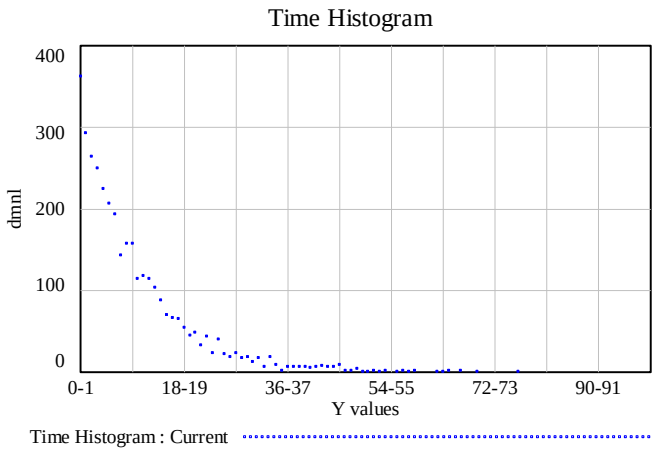

Figure 3: Distribution of the Vulnerability Patch Work Multiplier

Some dormant vulnerabilities remain dormant only until they are used in an attack, in which case there becomes an urgent need for a patch by susceptible organizations. As shown in Figure 4, patches are generated in the same way as before, but there is an acceleration factor due to the urgency that multiplies the speed of patching. This factor is a parameter of the model since we do not yet have firm data on the increased speed of patching for urgent vulnerabilities. Some fraction of vulnerabilities will never be patched; we refer to these as public forever-day vulnerabilities. Zero-day vulnerabilities, by definition, become public the day they are exploited and remain "dormant" only as long as the time it takes to generate the urgent patch.

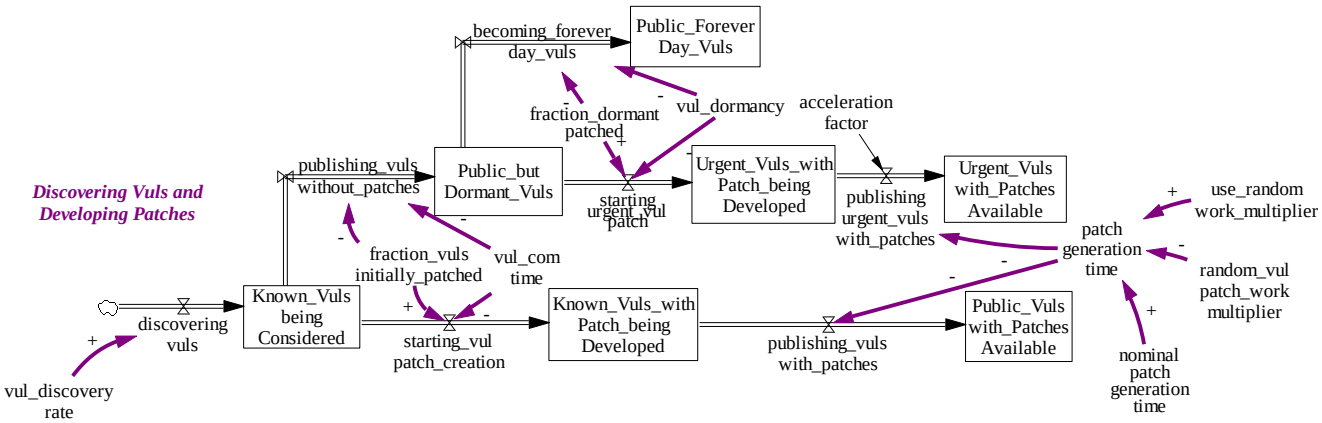

Figure 4: Dormant Vulnerability Possibly Becoming Urgent

Switching gears a bit in Figure 5, we now turn to the lifecycle of vulnerabilities as they are processed in CVE.[2] Vulnerabilities may be discovered after a patch has become available,

[2] We use "CVE" here for convenience since CVE is the most well-known vulnerability identifier in use in the vulnerability management space. However, the model is intended to apply to any process by which vulnerabilities are selected for inclusion into a list or database and subsequent remediation.

but CVE generally targets those vulnerabilities discovered before patching and only once those vulnerabilities become publicly known. A five-tier framework is used to decide whether to report a discovered vulnerability (Ragan, 2016). Each tier is associated with a set of organizations—the vendors—whose product vulnerabilities are assigned to the tier. Each tier represents qualitatively whether vulnerabilities in that tier will be reported in CVE. The system dynamics model of this tier framework—specifically the array variable "fraction CVE vuls reported"—assigns probabilities associated with whether a vulnerability in the tier is processed as follows:

- tier 1 (vulnerabilities *must* be reported) : p=1.0
- tier 2 (vulnerabilities *should* be reported) : p=0.95
- tier 3 (vulnerabilities *may* be reported) : p=0.60
- tier 4 (vulnerabilities *may not* be reported) : p=0.3
- tier 5 (vulnerabilities *must not* be reported) : p=0.0

These probabilities are initial estimates only; they are based strictly on the qualitative wording. They can be updated as more information comes to light about how the CVE responders interpret the tiering requirements. Absent that, we can run Monte Carlo simulations in VenSim to determine the effect of probability variance on simulation behavior. Lastly, the simulation must determine what fraction of vulnerabilities discovered falls into each tier. This fraction, which may change over time, is represented in the array variable "tier fractions over time" as an effect function of model simulation time on the tier percentages. Based on these variables and the inflow of discovered vulnerabilities, each considered vulnerability is processed and eventually published with a unique CVE ID.

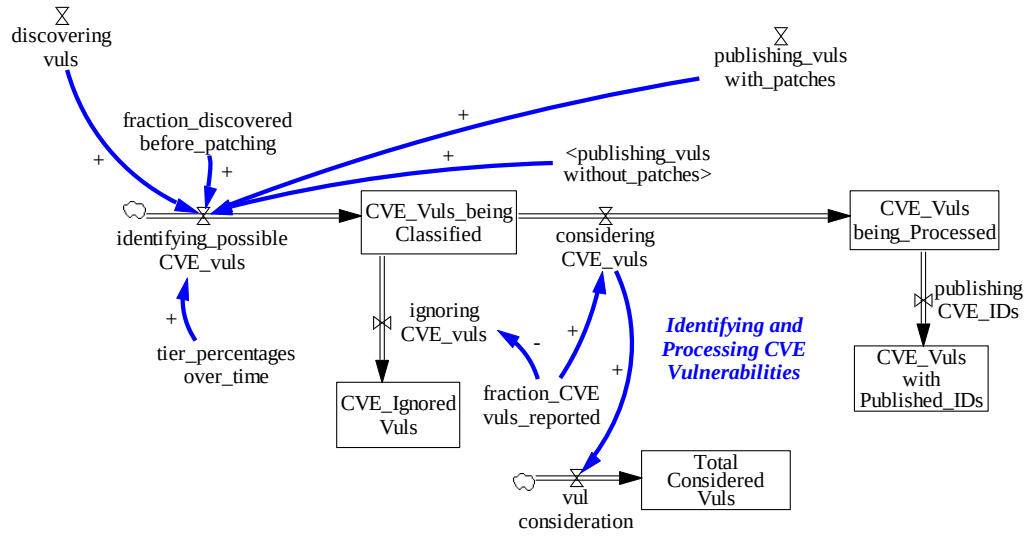

Figure 5: CVE Vulnerability Processing Lifecycle

Figure 6 deals directly with staffing to respond to vulnerabilities. CVE staff can handle the workload demand created by CVE vulnerability processing at a rate given by their per-responder productivity and the number of responders. Ideally, CVE managers would hire more staff if the perceived adequacy of CVE reporting is not keeping up with the desired adequacy. Perceived adequacy is measured as the ratio of the CVE vulnerabilities published and the total vulnerabilities considered in the CVE process, ranging from 0 to 1:

Perceived adequacy of vul reporting
= *CVE Vuls with Published IDs / Total Considered Vuls*

This hiring process creates a balancing feedback loop that (optimally) drives staff hiring to a level of desired adequacy, but funding levels and available capability may not permit hiring to the needed level.

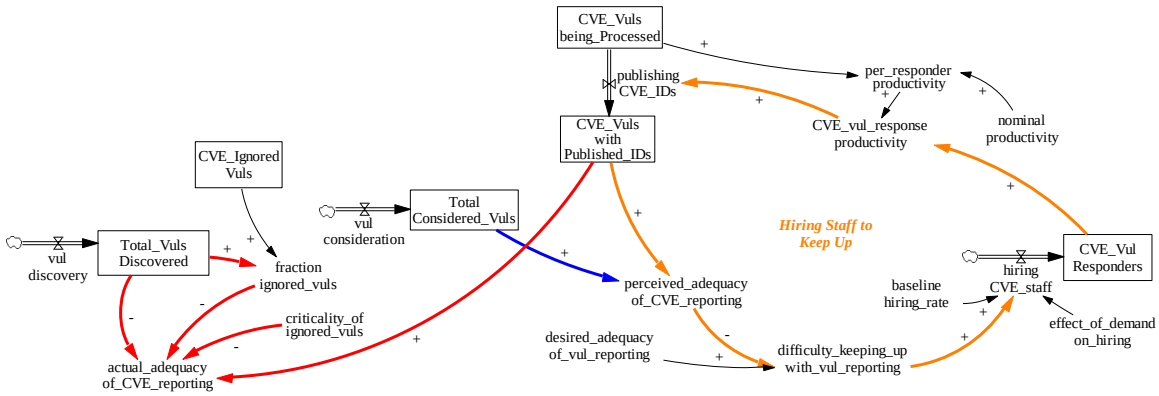

Figure 6: Expanding CVE Vulnerability Response Capacity Based on Perceived Demand

While the *perceived adequacy of CVE reporting* is calculated out of the *Total Considered Vuls*, the *actual adequacy of vul reporting* is calculated out of the *Total Vuls Discovered,* which includes the vulnerabilities ignored in CVE processing. Rather than treating ignored vulnerabilities as full weight, we allow them to have diminished influence as determined by the fraction *criticality of ignored vulnerabilities*:

actual adequacy of vul reporting :
= CVE Vuls with Published IDs
/ ((1-fraction ignored vuls)*Total Vuls Discovered
+ (criticality of ignored vuls*fraction ignored vuls*Total Vuls Discovered))

The benefit associated with creating patches that address identified vulnerabilities comes as organizations apply relevant patches in defense of their systems and data. Figure 7 shows that although patches are made available to defenders, as those patches are published for many organizations, it is the publication of the CVE ID that is critical to whether they decide to actually apply the patch or not. Some fraction of available patches will be ignored,

which in subsequent refinements can create risk for the defender. The model is parameterized on the

- number of infrastructure systems being defended (initially, 1000 systems)
- average number of patches per vulnerability required per system (initially, 1 patch/vul)
- fraction of patches applied (initially, 0.9)

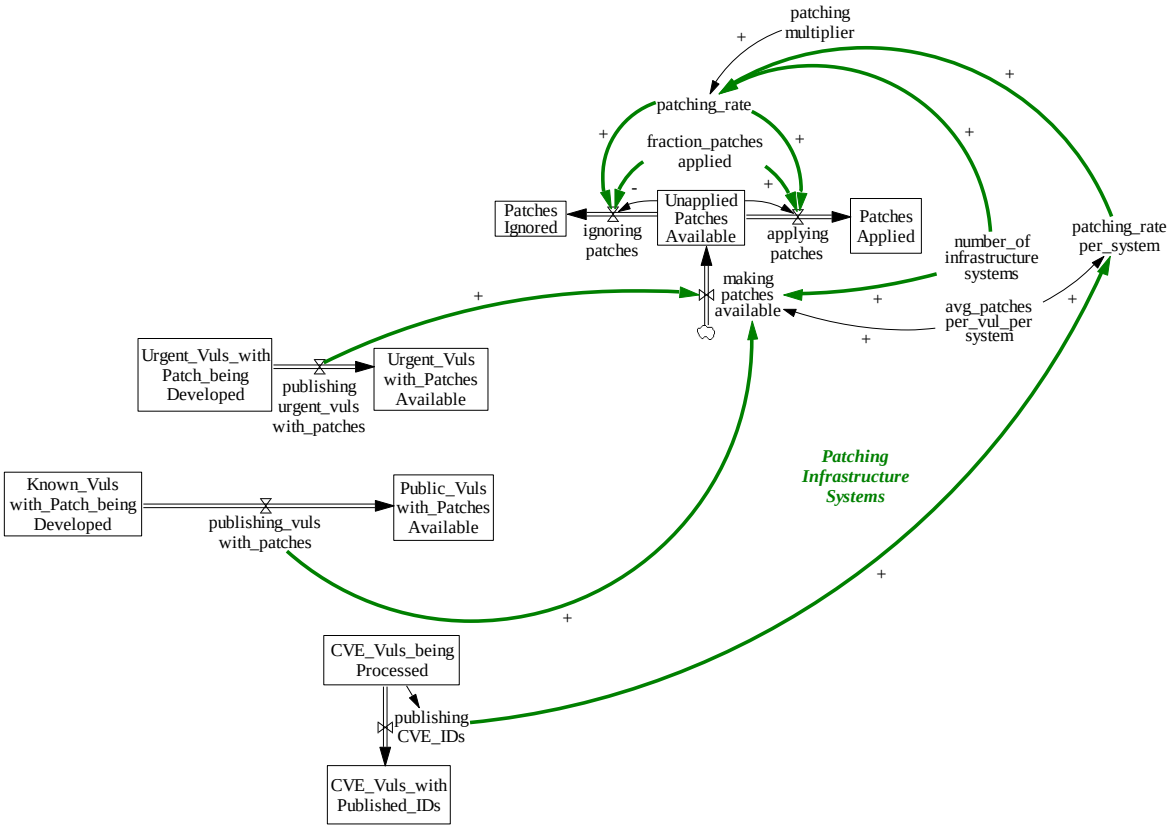

Figure 7: Patching Systems Based on CVE Publishing

Just as we had a measure for the *perceived adequacy of vulnerability reporting* described above, we have a measure of *perceived adequacy of vulnerability management* as shown in Figure 8. The measure is the ratio of the defender's *perception of the patches applied* to the *total patches needed*, which ranges from 0 to 1. For convenience in the model, we actually define this value equivalently as 1 minus the ratio of the defender's *perception of patches not applied* to the *total patches needed*. Both the numerator and denominator of the ratio could be multiplied by the term *fraction patches applied*, but this term cancels out to provide the following formulation:

perceived adequacy of vul management :

= 1 – (perceived patches not applied / total patches needed)

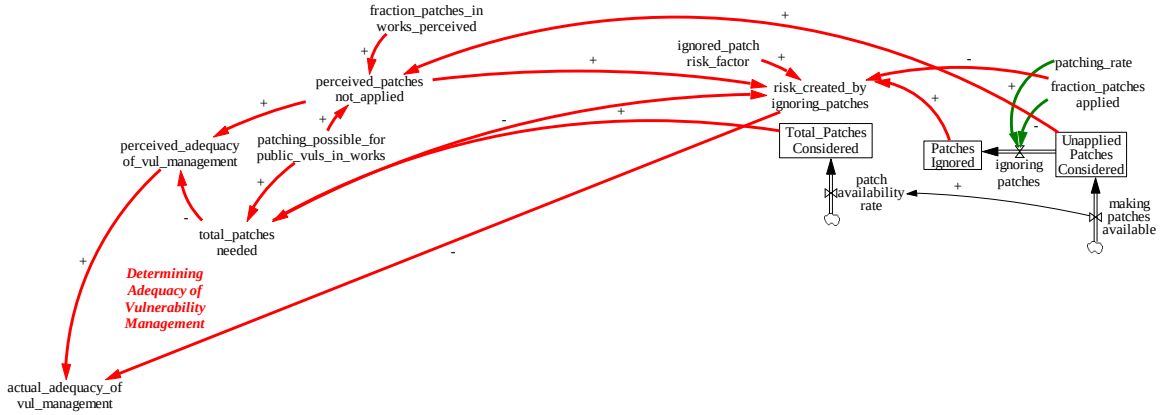

Figure 8: Adequacy of Vulnerability Management Perceived Operationally

Both *perceived patches not applied* and *total patches needed* depend on patches currently made available to defenders as well as patches currently "in works," that is patches that have yet to be developed. The primary difference between the two is that the *perceived patches not applied* includes only the unapplied patches available out of the total patches available, and only a fraction – *fraction patches in works perceived* (assumed to be 0.2 initially) – of the *patching possible for public vulnerabilities in works*:

perceived patches not applied
= Unapplied Patches Available
+ (patching possible for public vuls in works
* fraction patches in works perceived)

total patches needed
= Total Patches Available
+ patching possible for public vuls in works

patching possible for public vuls in works
= number of infrastructure systems
* avg patches per vul per system
* (Public but Dormant Vuls
+ Known Vuls with Patch being Developed
+ Urgent Vuls with Patch being Developed)

The measure *perceived adequacy of vul management* described above admits that defenders probably understand that their vulnerability and patch management is not perfect, at least to the extent that their systems are susceptible to a vulnerability between the point of discovery and the point at which a patch to that vulnerability is made available. Beyond that, we can measure the *actual adequacy of vulnerability management* that also measures the risk created by erroneously ignoring patches by the defenders. The measure can be broken down into two components as seen below.

actual adequacy of vul management
= (perceived adequacy of vul management - risk created by ignoring patches)
- ((perceived adequacy of vul management - risk created by ignoring patches)
* (1 - actual adequacy of vul reporting)

risk created by ignoring patches
= ignored patch risk factor
*(Patches Ignored
+ (1-fraction patches applied)*total vuls without patches applied)
/ total patches needed)

The first term subtracts the *risk created by ignoring patches* from the *perceived adequacy of vulnerability management*. This reflects that the actual adequacy is less than the perceived adequacy due to the potential for erroneously ignoring critical patches. But the situation is worse than that alone due to the fact that vulnerability reporting itself is not perfect. The second term of *actual adequacy of vulnerability management* subtracts from this amount, a fraction of the amount based on the inadequacy of vulnerability reporting. This adjustment is needed since the *perceived adequacy of vulnerability management* only accounts for those vulnerabilities reported to defenders through the CVE process, which itself only deals with a fraction of the total vulnerability population.

2.2 Simulation Results

We run the model described above with a rate of vulnerability discovery sufficient to generate the historical rate of assigning CVE IDs, which are reported publicly. We had data on CVE ID assignment from the beginning of 1999 to the middle of 2017, which establishes the time period of the simulation. In Figure 9, the behavior-over-time graph on the left shows the historical figures regarding CVE IDs assigned (simulation run #2) compared to that generated by the model (run #1). The graph on the right shows the total number of vulnerabilities discovered that were required to generate these CVE ID assignment numbers given the initial model setup.

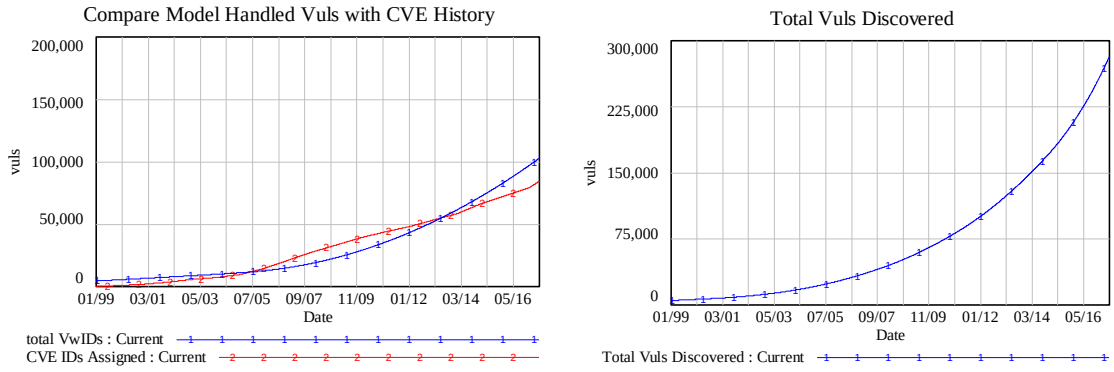

Figure 9: Vulnerability Discovery Level and Calibration with Historical CVE Reporting Levels

The graphs in Figure 10 demonstrate the partitioning of CVE vulnerabilities into the four tiers described previously. The graph on the left shows how the fraction of vulnerabilities in each tier changes over time in the model. While we did not have hard data on these fractions, we believe that the displayed trend over the 18-year period is plausible. The fraction of vulnerabilities in tier 1 goes down steadily over the period and is replaced with slightly increasing fractions in the other three tiers. The second graph simply verifies that the model accurately reflects the level of reporting of CVE vulnerabilities (assigned IDs), as described previously in qualitative guidance for reporting. The fraction of CVE vulnerabilities ignored in each tier is simply one minus the fraction seen in this graph (i.e., vulnerabilities are either reported or ignored; there is no other option). Figure 11 splits out the quantity and timing of reported and ignored vulnerabilities based on these fractions.

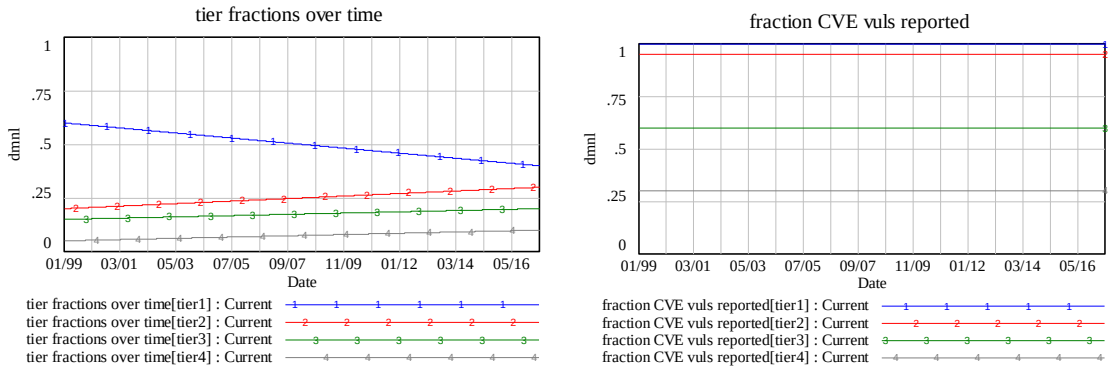

Figure 10: Vulnerabilities by Tier

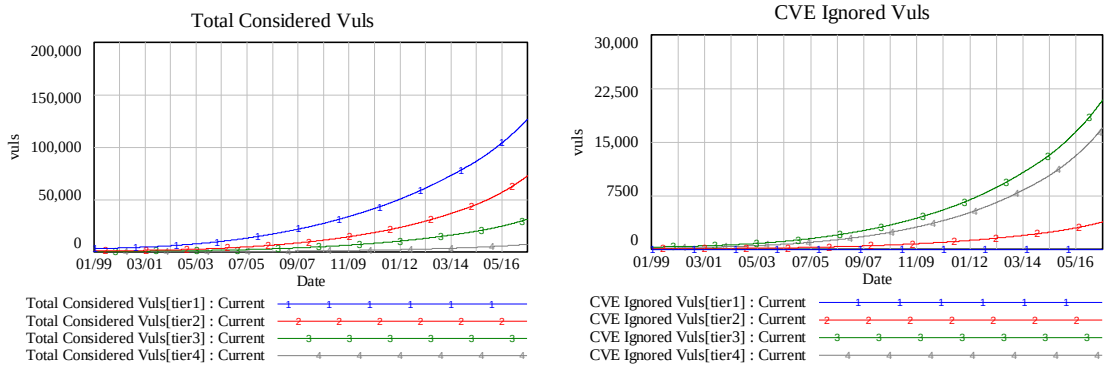

Figure 11: Considered Versus Ignored CVE Vulnerabilities over Time

Figure 12 shows the fraction of total vulnerabilities ignored in the CVE process as it relates to the level of cumulative system patching. The tiered approach described above combined with the general decline in tier 1 vulnerabilities and the increase in lower tier vulnerabilities gives rise to a steady increase in the fraction of ignored vulnerabilities. This increase results in the displayed decline in the fraction of system patches applied by defenders since many of the vulnerabilities ignored in the CVE process have critical patches available but are not on the radar screens of defenders since no CVE identifier was assigned.

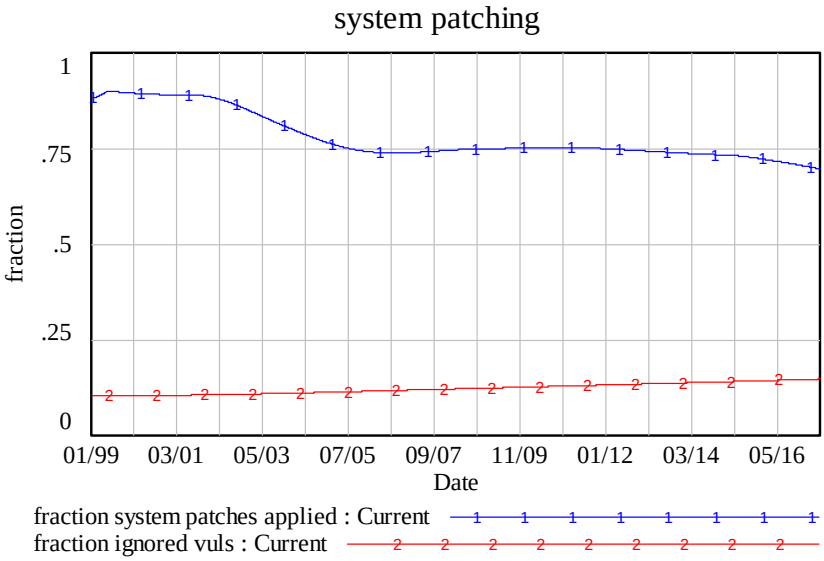

Figure 12: Correlation between Cumulative System Patching and Fraction Vulnerabilities Ignored

The results of model execution, as seen in the measures of perceived and actual vulnerability management (shown in Figure 13), are striking. The behavior-over-time graph on the left shows that perceived adequacy of vulnerability management drops, hovering around 0.8 in the time frame of the simulation. However, actual adequacy of vulnerability management drops significantly more, to less than 0.4 in the time frame of the simulation. The graph on the right shows that even if the perception of the adequacy of vulnerability management on the part of the defenders was near perfect (i.e., if the defender were able to immediately patch any vulnerabilities they were aware of or could become aware of), the actual adequacy of vulnerability management still hovers below 0.5. In this case, the actual adequacy of vulnerability management approaches the actual adequacy of CVE reporting.

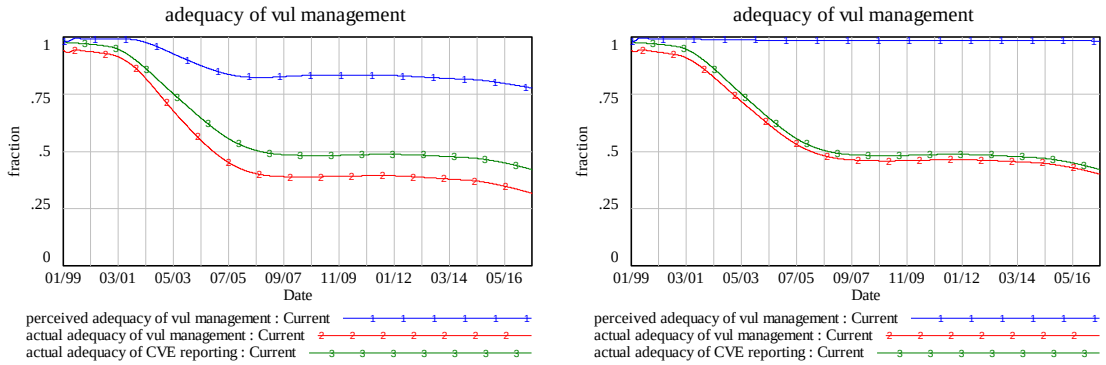

Figure 13: Adequacy of Vulnerability Management over Time

3 Agent-Based Analysis of Multi-Party Coordinated Vulnerability Disclosure

The agent-based model used to analyze MPCVDs was constructed using Ventity, a tool developed by Ventana Systems, Inc., that supports the modular construction of socio-technical models for scalable development by independent teams. Ventity enables the development of hybrid agent-based and system dynamics models. Three primary agent types are specified: Finders, Vendors, and Coordinators. Figure 14 elaborates the behavior drivers for each of these agent types:

- **Finders**: Finders are motivated by making money, by establishing their reputation, or a combination of these two. Bug bounties may serve as the financial incentive, while vulnerability publication may serve as the reputational incentive.
- **Vendors**: Vendors are motivated ultimately by revenue generation, and secondarily by generating and maintaining their customer base. Keeping quiet about vendor product vulnerabilities serves the vendor well before the patch is available; but after the patch is available, it may wish to report the availability of patches as soon as possible.
- **Coordinators:** The coordinator is, of course, the manager of the coordinated vulnerability disclosure, with the purpose of minimizing deployer exposure. The general rule of good behavior and purpose for MPCVDs is to maintain secrecy during the term of the MPCVD, called the embargo period.

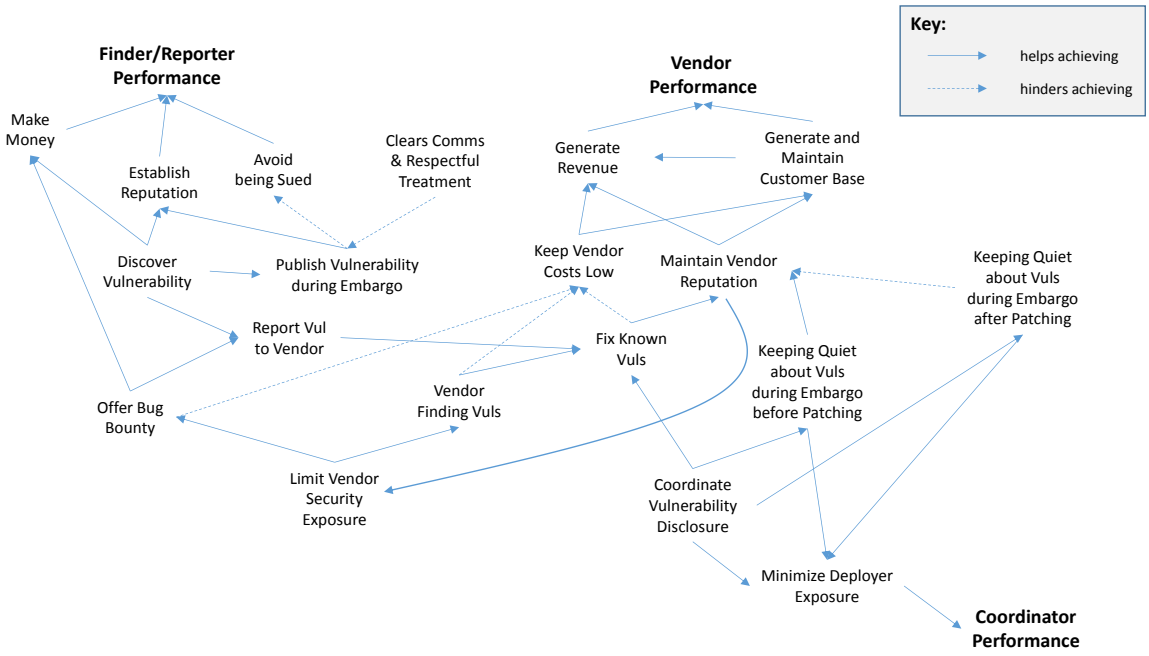

Figure 14: Agent Goal Alignment

The Ventity simulation runs many MPCVDs over two years to assess management strategies and policies for the coordinator to try out. Adjustable model parameters include

the number of finders and vendors, size distribution of the MPCVDs and vendors, embargo duration, and the likelihood of accidental and purposeful disclosure. The measure of social cost for deployers due to technical vulnerabilities is elaborated in Appendix E and includes the likelihood of vulnerability exploitation, the maximum amount of damage, hacker vulnerability discovery time, attack rate per deployer, the amplification of the attack rate after disclosure, and user workaround costs over time.[3]

3.1 Calibration

The model was calibrated based on operational data collected by the CERT Coordination Center (CERT/CC) at Carnegie Mellon University's Software Engineering Institute. During the entire period of study covering 24 years, 1,400 of 11,000 cases (12%) involved pre-disclosure coordination. In the 5-year period between 2013 and 2018, the CERT/CC coordinated approximately 1,000 vulnerability disclosure cases per year. Only a small fraction of these were large MPCVD cases (Householder 2018). Interviews with the CERT/CC's vulnerability coordination team helped us arrive at the model estimates for embargo failures and vendor participation rates.

The current model under development has been calibrated along four dimensions:

1. the quantity of MPCVDs per year (60-80 per year)
2. the distribution of the size of the MPCVDs (fat-tailed, exponential)
3. the ratio of MPCVDs in which the embargo held (percentages in the 1990s)
4. the greater participation in MPCVDs of larger vendors than smaller ones

Figure 15 shows the Ventity simulation results, as behavior-over-time graphs spanning 10 years (120 months).

[3] adapted from (Cavusoglu, 2007), equations 3 and 4 on page 175

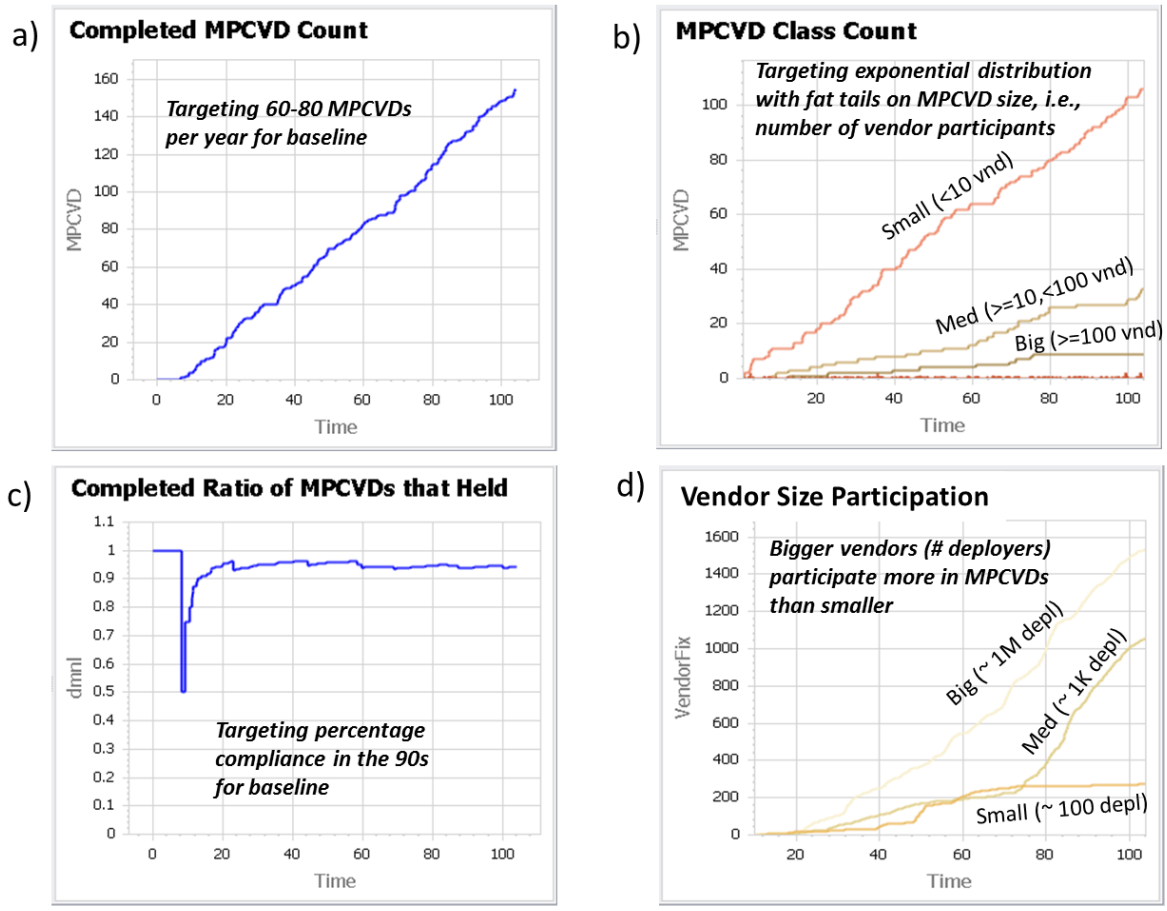

Figure 15: Calibration Dimensions of the MPCVD Model

3.2 Simulation Results

One question we started our analysis with was how to improve the cooperation of vendors in terms of not divulging the existence of vulnerabilities within an MPCVD until the end of the embargo period. An embargo period of 45 days (or about 6.5 weeks in the simulation) is current common practice and was used for model calibration in the previous section.[4] While faster patching is generally more expensive for all vendors (in terms of pulling developers off planned development work), a small number of vendors choose to quickly develop patches and leak them to their users prior to the end of the embargo period. In the simulation, this resulted in the under 10% of MPCVDs that did not hold shown in Figure 15c.

[4] The CERT/CC's default embargo period is 45 days, with exceptions for active exploitation (shorter) or situations where extensive work by multiple parties is needed (longer) https://vuls.cert.org/confluence/pages/viewpage.action?pageId=30638083

In Figure 16, the simulation also shows that shortening the embargo period (from 6.5 weeks to 4 weeks) does, in fact, decrease MPCVD reneging rates; increasing the embargo period (from 6.5 weeks to 8 weeks) increases reneging rates. This is expected, since vendors will not be tempted to renege on MPCVD until after they've developed a patch. In addition, the longer an embargo endures after patch development, the greater the chance of reneging.

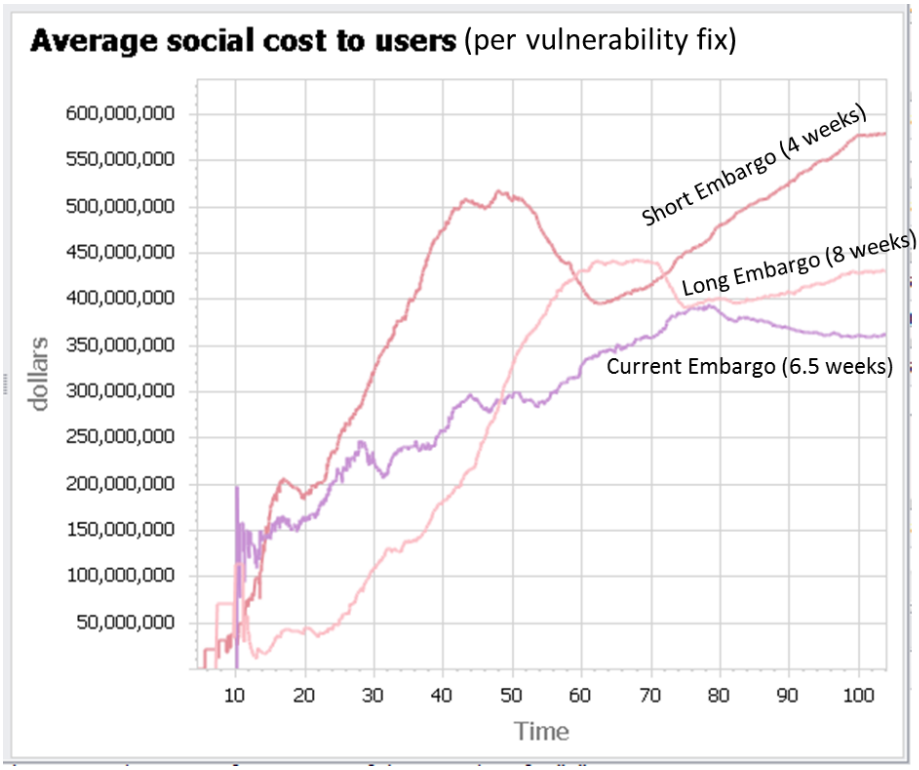

Figure 16: Percentage of MPCVD Embargos that Hold by Length of Embargo Period

While the short embargo ensures more MPCVDs hold through the embargo period, they are the most costly to users, as seen in Figure 17. The current embargo period is a good middle ground to reduce cost to users. The sooner patches are distributed, the lower the social cost to deployers, whether the patch is distributed (and vulnerability disclosed) before or after the embargo. Shortening the embargo time leads to lower rates of reneging, but high rates of not having a patch available to deployers after the embargo period. This is the worst situation for the deployer and results in the highest social costs. We therefore conclude that adjusting the embargo period to increase the likelihood that patches can be developed **just in time** appears to be a good strategy for reducing cost.

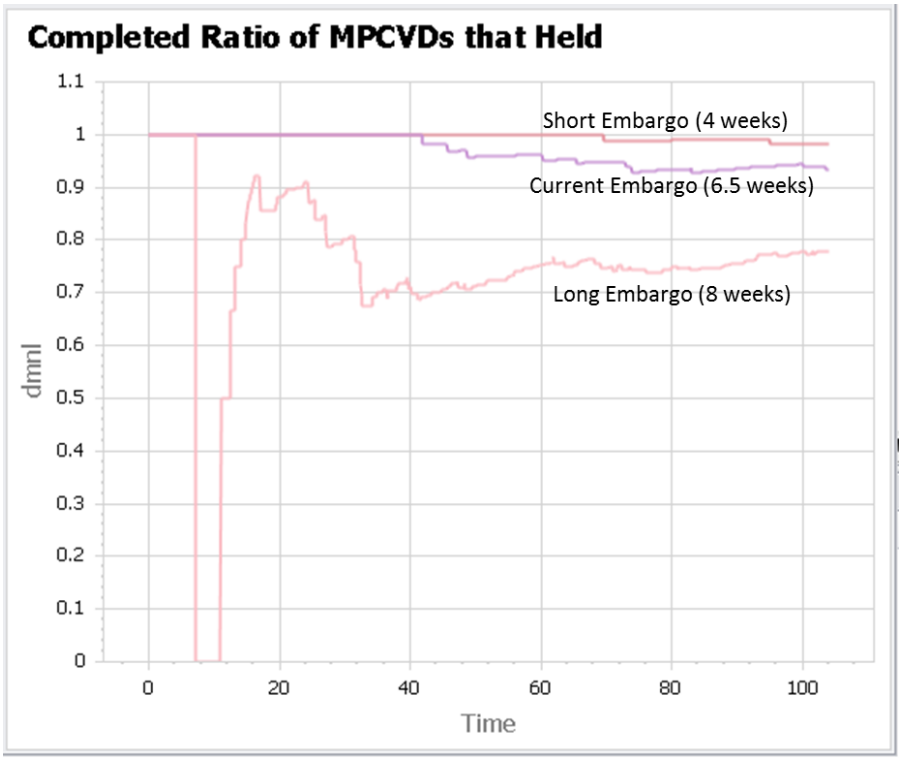

Figure 17: Average Social Cost to Users by Length of Embargo Period

4 Summary and Conclusions

This paper presents multi-method modeling and analysis of two critical foundational processes of the cybersecurity vulnerability management ecosystem:

1. the vulnerability management process for discovering, cataloging, and remediating vulnerabilities
2. the MPCVD process for coordinating the patching and disclosing vulnerabilities among multiple vendors

Our work applied system dynamics analysis, using VenSim, to the vulnerability management process and agent-based model analysis using Ventity. Both tools are from Ventana Corporation.

Some may argue that you can conduct agent-based modeling and analysis directly in VenSim. To some extent, this is true, but as agents become more and more heterogeneous in their behavior, VenSim models (and system dynamics models generally) become more awkward to specify, implement, and analyze. Ventity overcomes this problem by allowing the modeling and execution of fully heterogeneous agents, while also improving the ability of the modeler to build modular models constructed by disparate teams. Ultimately, it is desirable to have one unified, coherent, and comprehensive tool to enable full-capability modeling of agent-based and system dynamic aspects of a problem, as appropriate, without

switching tools. Ventity may be headed in this direction, but currently the system dynamics modeling and analysis in VenSim is more capable than it is in Ventity.

Preliminary results from our multi-method analyses show that the cybersecurity infrastructure can become more vulnerable over time simply as a result of the vendor-based tiering of vulnerabilities used in the CVE process. In addition, the goal of simply maintaining the secrecy of MPCVDs is not necessarily the right criteria, in and of itself. If that were the criteria, you might decide to increase the length of the embargo period in order for organizations to develop the patches. It appears that adjusting the embargo period to increase the likelihood that patches can be developed by most vendors **just in time** is a good strategy for reducing cost. The larger conclusion from our multi-method analysis is that the misapplication of either of these foundational aspects could contribute to the fragility and risk associated with the many national infrastructures and organizational missions that rely on the Internet.

We describe our analysis effort as a work in progress and our conclusions as preliminary due to the limited calibration and validation of our models. We are using data from the CERT Coordination Center's CVD function, as described, for our continuing calibration and validation efforts. In addition, our to-date analysis has identified additional areas to consider:

- Additional questions to investigate include the following: Are there policies that can improve the cooperation of vendors in MPCVDs AND reduce social costs? OR is the best policy to shun non-cooperators? If so, when can you optimally bring them in?
- Consider other measures of social cost due to cybersecurity vulnerabilities over that used in *Efficiency of Vulnerability Disclosure Mechanisms to Disseminate Vulnerability Knowledge* (Cavusoglu 2007). This area of study has been relatively active and one we need to continue to review.
- Continue to tune the model parameters to what we know or can easily find out. Where concrete data is not available, either direct future data collection efforts in this direction or, in the near term, focus on plausibility, based on subject matter expert opinion.
- Consider additional review and refinement of model structure and logic:
    a. logic for the (purposeful) decision to disclose early based on the extent patch developed
    b. logic for accidental vs. purposeful disclosure in whether early disclosure occurs
    c. the impact that the size of the vendor has on its behavior (accidental vs. purposeful disclosure)

d. the impact of whether a vendor discloses early or not has on other vendors in the party, or the community at large

The last of these considerations has the potential to incorporate richer feedback dynamics that may be a central driver in the cybersecurity of national infrastructures. While the results of our initial efforts, described in this paper, should be viewed as preliminary, we believe that the approaches used and the depth of the modeling and simulation are sufficient to begin to understand key implications of these processes and possible avenues for their improved application in the future.

5 Acknowledgements

The authors thank SEI/CERT colleagues Soumyo Moitra, William Casey, and Jeffrey Chrabaszcz for their insights and data analysis that helped ground this modeling effort. We also appreciate the capable review and technical edits by Sandy Shrum.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM19-0192

References

Caralli, R., Allen, J., & & White, D. (2010). *CERT Resilience Management Model: A Maturity Model for Managing Operational Resilience.* Addison-Wesley Professional . Retrieved from https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=514489

Cavusoglu, H. &. (2007). Efficiency of Vulnerability Disclosure Mechanisms to Disseminate Vulnerability Knowlege. *IEEE Transactions on Software Engineering*, 171-185. Retrieved from https://ieeexplore.ieee.org/document/4084135/

Householder, A. (2018). Analyzing 24 Years of CVD. *FIRST Vendor TC.* Atlanta, GA. Retrieved from https://www.first.org/resources/papers/atlanta2018/20180227-Analyzing-24-Years-of-CVD-Allen-Householder-FIRST-PSIRT-TC.pdf

Householder, A., Wassermann, G., Manion, A., & & King, C. (2017). *The CERT Guide to Coordinated Vulnerability Disclosure.* Pittsburgh, PA: Carnegie Mellon University. Retrieved from https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=503330

ISO/IEC Technical Committee JTC 1/SC 27 IT Security Techniques. (2013). Information Technology - Security Techniques - Vulnerability Handling Processes. Retrieved from https://www.iso.org/standard/53231.html

NIST Information Technology Laboratory. (2019). National Vulnerability Database. Retrieved from https://nvd.nist.gov

NIST. (n.d.). NIST Special Publication 800-40 Revision 3: Guide to Enterprise Patch Management Technologies. Retrieved from https://www.nist.gov/publications/guide-enterprise-patch-management-technologies

Ragan, S. (2016). Over 6,000 Vulnerabilities Went Unassigned by MITRE's CVE Project in 2015. *CSO Online*. Retrieved from https://www.csoonline.com/article/3122460/over-6000-vulnerabilities-went-unassigned-by-mitres-cve-project-in-2015.html

Thomson, I. (2018). Revialed: El Reg blew lid off Meltdown CPU bug before Intel told US govt - and how bitter tech rivals teamed up. *The Register*. Retrieved from https://www.theregister.co.uk/2018/08/09/meltdown_spectre_cert_timing/

Appendix A: System Dynamics Modeling Overview

System dynamics helps analysts model and analyze critical behavior as it evolves over time within complex socio-technical domains. Figure 18 summarizes the notation used in our system dynamics model.

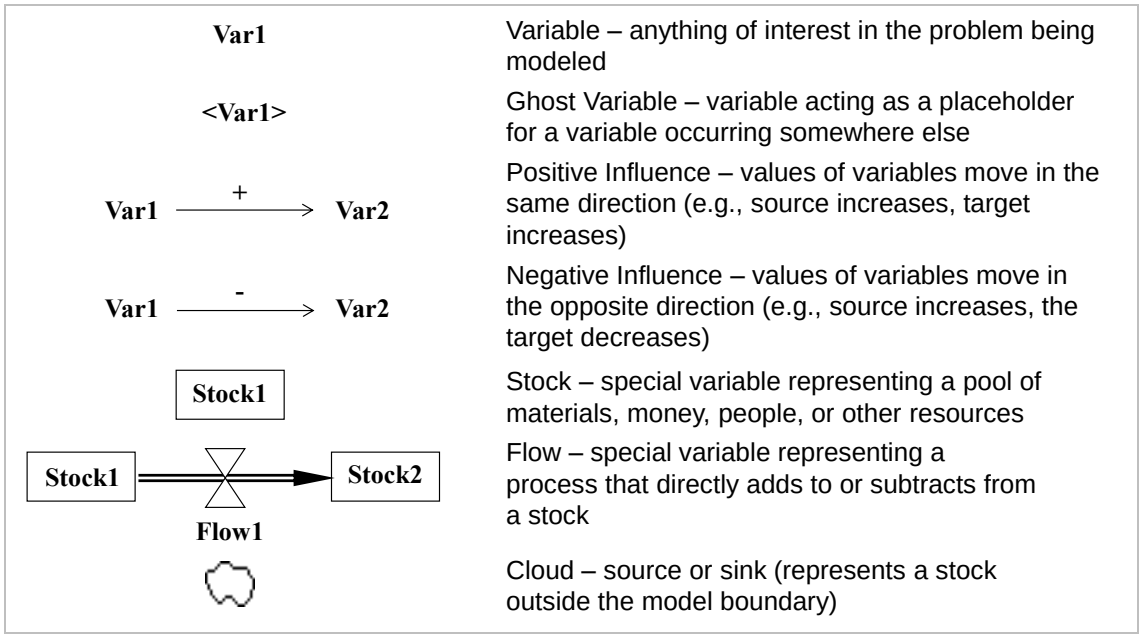

Figure 18: System Dynamics Notation

The primary elements are variables of interest, stocks (which represent collections of resources, such as dissatisfied employees), and flows (which represent the transition of resources between stocks, such as satisfied employees becoming dissatisfied). Signed arrows represent causal relationships, where the sign indicates how the variable at the arrow's source influences the variable at the arrow's target. A positive (+) influence indicates that the values of the variables move in the same direction, and a negative (–) influence indicates that they move in opposite directions.

A connected group of variables, stocks, and flows can create a path that is referred to as a feedback loop. At this stage in our modeling effort, we have not identified any significant feedback loops.

Appendix B: The Vulnerability Management Ecosystem System Dynamics Model

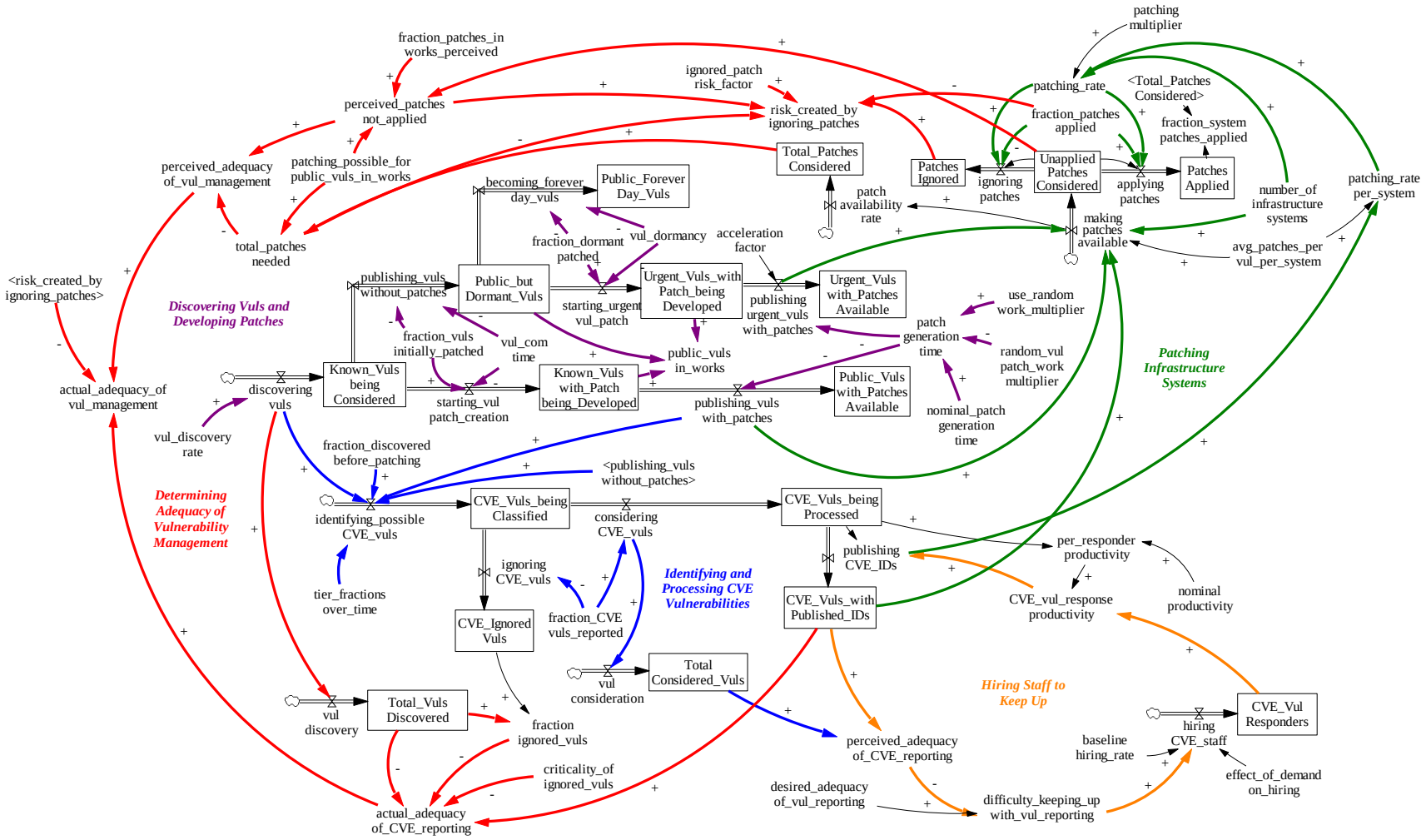

Figure 19: Vulnerability Management Ecosystem System Dynamics Model

Appendix C: System Dynamics (VenSim) Model Parameters

Model Variables	Value in Baseline	Units
acceleration factor	2.00	range 1 to 10
avg patches per vul per system	1.00	patches/vul/system
baseline hiring rate	0.10	people//month
criticality of ignored vuls	0.10	range 0 to 1
desired adequacy of vul reporting	0.90	range 0 to 1
eval period	12.00	months
fraction CVE vuls reported[tier1]	1.00	range 0 to 1
fraction CVE vuls reported[tier2]	0.95	range 0 to 1
fraction CVE vuls reported[tier3]	0.60	range 0 to 1
fraction CVE vuls reported[tier4]	0.30	range 0 to 1
fraction discovered before patching	0.60	range 0 to 1
fraction dormant patched	0.50	range 0 to 1
fraction dormant perceived	0.20	range 0 to 1
fraction patches applied	0.90	range 0 to 1
fraction vuls initially patched	0.50	range 0 to 1
init vul responders	3.00	people
initial tier percentages[tier1]	0.60	range 0 to 1
initial tier percentages[tier2]	0.20	range 0 to 1
initial tier percentages[tier3]	0.15	range 0 to 1
initial tier percentages[tier4]	0.05	range 0 to 1
initial unapplied patches available	0.02	range 0 to 1
initial vuls being classified fraction	0.00	range 0 to 1
initial vuls being processed fraction	0.00	range 0 to 1
initial vuls processed	5000.00	vuls
max vpf	45.00	positive real
min vpf	0.00	positive real
nominal patch generation time	0.50	months
nominal productivity	30.00	vuls/people/month
number of infrastructure systems	1000.00	systems
patch pub delay	3.00	months
patching delay	3.00	months

Model Variables	Value in Baseline	Units
patching multiplier	10.00	positive real
shift vpf	1.00	positive real
stretch pwf	10.00	positive real
use random work multiplier	0.00	toggle 0/1
vul com time	1.00	months
vul dormancy	3.00	months

Appendix D: The Ventity Interface

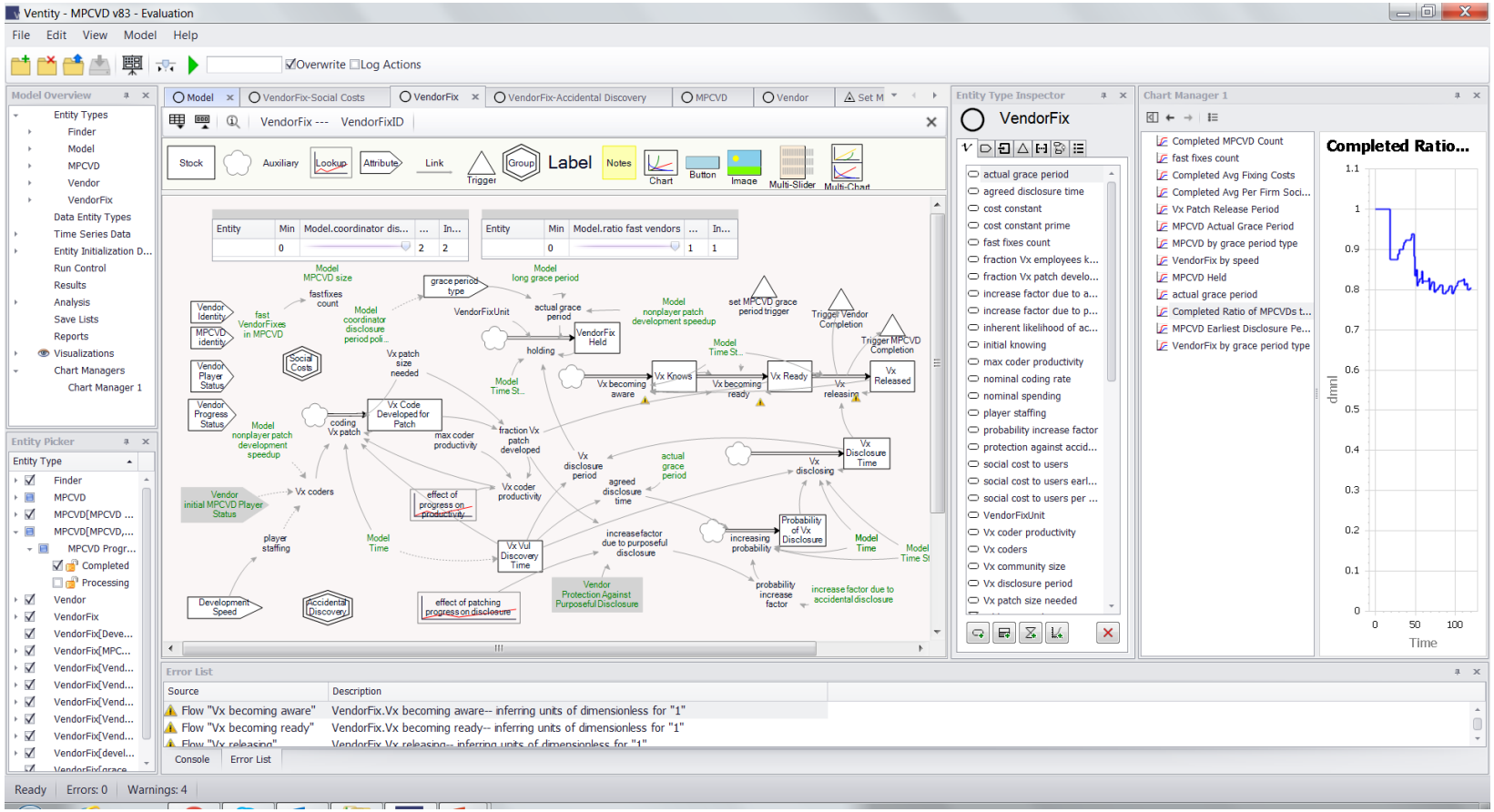

Figure 20: Ventity Interface

Appendix E: Agent-Based Model (Ventity) Parameters of the Social Cost Measure

The social cost measure from *Efficiency of Vulnerability Disclosure Mechanisms to Disseminate Vulnerability Knowledge*[5] appears below. The cost due to patch development is not included, to represent only the social cost to users (deployers).

Social cost to users before embargo period over (p) = $\alpha N \delta D a p^2/4$
Social cost to users after embargo period over (p) = $\gamma N \delta D k a (p-t_0-T)/2 + Ns(p-t_0-T)$
Social cost to users (p)
= IF $p<=t_0+T$ THEN Social cost to users before embargo period over (p)
ELSE Social cost to users before embargo period over (t_0+T)
+ Social cost to users after embargo period over (p)

Equation Variables	Description	Value in Baseline	Units
t_0	time MPCVD starts	varies	weeks
δ	likelihood of vul exploitation	0.01	dmnl
T	agreed disclosure time	short (4 weeks), current (6.5 weeks), long (8 weeks)	weeks
p	time period vendor releases patch	varies	weeks
γ	inefficiency measure for user workaround due to missing patch	0.5	dmnl
α	hacker vul discovery time	2	weeks
a	attack rate per deployer prior to disclosure	0.1	attacks/deployer/week
k	amplification of attack rate after disclosure	10	dmnl
N	number of deployers for the vendor	big (~1M), medium (~1K), small (~100)	deployers
D	maximum amount of damage to deployer due to missing patch	$50K	dollars/attack

[5] See (Cavusoglu, 2007), equations 3 and 4 on page 175.