Survivability Analysis of a Computer System under an Advanced Persistent Threat Attack

Ricardo J. Rodríguez[1], Xiaolin Chang[2], Xiaodan Li[3], and Kishor S. Trivedi[3]

[1] Dept. of Computer Science and Systems Engineering, University of Zaragoza, Spain
[2] Dept. of Information Security, Beijing Jiaotong University, P. R. China
[3] Dept. of Electrical and Computer Engineering, Duke University, USA
rjrodriguez@unizar.es, xlchang@bjtu.edu.cn, {xiaodan.li,ktrivedi}@duke.edu

Abstract. Computer systems are potentially targeted by cybercriminals by means of specially crafted malicious software called Advanced Persistent Threats (APTs). As a consequence, any security attribute of the computer system may be compromised: disruption of service (availability), unauthorized data modification (integrity), or exfiltration of sensitive data (confidentiality). An APT starts with the exploitation of software vulnerability within the system. Thus, vulnerability mitigation strategies must be designed and deployed in a timely manner to reduce the window of exposure of vulnerable systems. In this paper, we evaluate the survivability of a computer system under an APT attack using a Markov model. Generation and solution of the Markov model are facilitated by means of a high-level formalism based on stochastic Petri nets. Survivability metrics are defined to quantify security attributes of the system from the public announcement of a software vulnerability and during the system recovery. The proposed model and metrics not only enable us to quantitatively assess the system survivability in terms of security attributes but also provide insights on the cost/revenue trade-offs of investment efforts in system recovery such as vulnerability mitigation strategies. Sensitivity analysis through numerical experiments is carried out to study the impact of key parameters on system secure survivability.

Keywords: APT, Cyberattacks, Markov chains, Stochastic reward nets, Security metrics, Survivability, Transient analysis

1 Introduction

The number of incidents related to cyberattacks is increasing rapidly, according to numerous reports [1–3]. These cyberattacks have a cost of downtime and cleaning up of compromised systems, besides loss of customer confidence and of other possible long-term consequences due to loss and theft of information. This situation becomes specially critical when cybercriminals attempt to attack infrastructures that provide essential services to the society, such as financial services, power distribution, or water treatment plants [4]. In these systems, an intentional malfunction causing a discontinuity of service may lead to fatalities or injuries. Unfortunately, the number and sophistication of cyberattacks targeting these systems demonstrate an increasing trend [5, 6].

Malicious software (*malware*) are pieces of software specially crafted by cybercriminals to achieve their malicious goals [7]. There exist different types of malware depending on their behavior, such as viruses, worms, botnets, or keyloggers, among others [8]. When malware are designed to target a specific system, they are known as Advanced Persistent Threats (APTs) [9]. The term "advanced" means the target requires a sophisticated attack, since attackers make a previous reconnaissance of the target to know in advance as much as possible about the system to compromise. The term "persistent" means the goal of the threat is to maintain a presence on the targeted system for long-term control and data collection (which are later exfiltrated).

One of the first APTs was *Operation Aurora*, publicized by Google in 2010. Presumably coming from China and with an extremely wide-scale range, it targeted companies of different domains, such as Yahoo, Google, Symantec, Northrop Grumman, Morgan Stanley, and Dow Chemicals [10]. Another well-known APT is the *Stuxnet* attack, also discovered in 2010. This *cyber weapon*, attributed to the US and Israel, was specially designed to exploit Siemens PLCs in SCADA networks affecting Iranian nuclear facilities [9, 11]. APTs discovered in the wild from 2007 to 2013 with political intent, such as GhostNet, Flame, or DarkSeoul, among others, are summarized in [12].

An APT comprises of different stages. In *entry point* and *exploitation* stages, the APT gains access to a targeted system by means of zero-day vulnerabilities (i.e., a software flaw that is unknown to the vendor) or vulnerabilities already known but not yet patched. For instance, Stuxnet used four different zero-day vulnerabilities. After gaining access, the APT tries various methods to make itself persistent into the system (*infection* stage) and starts looking for data of interest to be stolen or modified (*lateral movement* stage). Once sensitive data are obtained, the APT will modify or send those data out of the organization's network boundaries (denoted *exfiltration* stage), thus compromising data integrity and confidentiality. In addition, the various actions undertaken during the attack may crash the system and then reduce system availability.

Assessing the impact of APTs on a system is important to characterize the system against these unexpected and intentional failures and to evaluate mitigation techniques that may be applicable. In this regard, survivability refers to a system's ability to withstand malicious attacks and support the system's mission even when parts of the system are damaged [13]. This paper, in particular, defines the *system secure survivability* as a transient measure of the ability of the system to provide pre-specified service with a certain security assurance during the system recovery from a vulnerability.

In this paper, we assess the survivability of a computer system targeted by an APT. A security model is developed to capture both the behavior of the system's response to a security attack and the actions performed by an attacker to cause such an attack. We make a simplifying assumption that all relevant event times are exponentially distributed and thus the model is a homogeneous continuous time Markov chain (CTMC). Note that a number of techniques are available to relax this assumption if needed [14]. We leave the relaxation of this assumption for future work. The generation and solution of the proposed Markov model is automated using a variant of stochastic Petri nets called Stochastic Reward Nets (SRNs). SRNs have been successfully used in the analysis of several domains [15–19], and can easily represent common characteristics of computer systems such as concurrency, synchronization, conditional branches, looping, and sequencing. We furthermore define four survivability metrics (see Section 3.1 for details) that account for: i) system recovery, ii) system availability, and iii) data confidentiality and/or integrity loss; after the public announcement of a software vulnerability and during vulnerability mitigation strategy is being deployed.

Related Work. Research has been conducted on survivability modeling and analysis in various fields and from different perspectives [20–24]. Regarding survivability metrics, little research has proposed quantitative evaluation metrics in terms of survivability. Quantitative measures were proposed in [25] to analyze the survivability of a resilient database system against intrusions, modeled with CTMC. This work was later extended to semi-Markov processes in [26]. Similarly, a general approach for survivability quantification of networked systems using SRNs was given in [27]. Survivability assessment of the Saudi Arabia crude-oil pipeline network, modeled with Generalized Stochastic Petri nets, was performed in [28]. All these works only analyze availability under unexpected events.

However, to the best of our knowledge, no work exists that proposes a quantitative assessment of the system secure survivability. The developed model in this paper not only considers the response of the system to a security attack, but also the actions performed by an attacker to cause such an attack and consider the transient behavior of the system in face of an attack. The proposed model and metrics let us investigate system security attributes (namely, confidentiality, integrity, and availability [29]) during the transient period which starts after a vulnerability discovery and through all the stages of an APT, until the vulnerability is fully removed from the system. This paper shows that the results not only enable us to quantitatively assess the system survivability in terms of security attributes but also provide insights on the cost/benefit trade-offs of the investment in system recovery efforts such as vulnerability mitigation strategies.

This paper is organized as follows. Background on Petri nets and Stochastic Reward nets is provided in Section 2. The system description and the model considered in this paper are presented in Section 3. Then, Section 4 deals with numerical results and discussion. Finally, Section 5 concludes the paper and outlines future work.

2 Previous Concepts

A Petri net [30] (PN) is a 4–tuple $\mathcal{N} = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$, where P and T are disjoint non-empty sets of *places* and *transitions*, and **Pre** (**Post**) are the pre–(post–)incidence non-negative integer matrices of size $|P| \times |T|$. The *pre-* and *post-set* of a node $v \in P \cup T$ are respectively defined as ${}^\bullet v = \{u \in P \cup T | (u, v) \in F\}$ and $v^\bullet = \{u \in P \cup T | (v, u) \in F\}$, where $F \subseteq (P \times T) \cup (T \times P)$ is the set of directed arcs.

Graphically, a PN is a bipartite directed graph having two disjoint types of nodes: *places*, drawn as circles; and *transitions*, drawn as bars. A directed arc that connects a place (transition) to a transition (place) is called an input (output) arc of the transition. An arc never connects the same type of nodes. A positive integer inscribed next to an arc specifies the multiplicity associated with the arc. Places that connect to a transition by input arcs are named *input places* of the transition. Similarly, places that are connected to a transition by output arcs are named *output places* of the transition. Each place may contain zero or more tokens, depicted by an integer (or black dots) within the circle representing the place. The number of tokens of a place denotes the *marking* of the place. A *Petri net system*, or *marked Petri net* $\mathcal{S} = \langle \mathcal{N}, \mathbf{m_0} \rangle$, is a Petri net $\mathcal{N}$ with an *initial marking* $\mathbf{m_0} \in \mathbb{Z}_{\geq 0}^{|P|}$.

A transition $t \in T$ is *enabled* when each of its input places has, at least, as many tokens as the multiplicity of the corresponding input arc. An enabled transition t can *fire* triggering, upon firing, two actions: first, a number of tokens equal to the multiplicity of the corresponding input arc is removed from each of its input places; and second, a number of tokens equal to the multiplicity of the corresponding output arc is deposited in each of its output places. Thus, the firing of a transition may yield a new marking of the Petri net, named *reached marking*. The *reachability set* is defined as the set of all markings reachable through any possible sequence of transitions, starting from the initial marking $\mathbf{m_0}$.

Stochastic Petri nets are Petri nets where each transition has an exponentially distributed firing time. Generalized Stochastic Petri nets [31] (GSPN) allow transitions that fires in zero time, named as *immediate transitions* and represented by thin black bars. The transitions that follow any distribution firing time are named *timed transitions* and represented by unfilled rectangles. Immediate transitions have always priority over timed transitions to fire. Similarly, immediate transitions with the same input places may have defined a probability to calculate the one that fires when they compete for firing. GSPN also includes inhibitor arcs.

A Stochastic Reward Net (SRN) [32] is a GSPN augmented with reward functions. In SRN, an enabling function (also called a *guard*) defines the enabling function of a transition as a marking-dependent function. In addition, both arc multiplicities and firing rates are allowed to be marking-dependent. SRN allows us to compute measures of interests by defining reward rates at the net level.

3 System Description and Model

To analyze and quantify security attributes of a system, we consider not only the system defender's response to a security attack, but also the actions taken by an attacker to cause the attack. This requires that the security model incorporates the behavior of both elements. In the following, we first describe the system considered in this paper, and then we present a Stochastic Reward Net model for survivability analysis of this system.

3.1 System Description

Fig. 1 depicts a flowchart detailing the actions of both the attacker and the defender, and the system changes due to these actions during the recovery from a vulnerability.

Let us consider a system in which an attacker has some interest in accessing into it. The attacker has acquired a previous knowledge of the system, but at the beginning there are no known vulnerabilities the attacker can take advantage of. We consider the attacker is not skillful enough to find unknown vulnerabilities. When a vulnerability is fully disclosed, the system is in the vulnerable state. Meanwhile, the defender starts the patch implementation and the attacker starts the exploit implementation. In the following, we use patch and vulnerability mitigation strategy interchangeably. The shaded part with dotted line in Fig. 1 describes the system state transitions and the shaded rectangles in it represents the system states.

There are eight system states: *System is vulnerable*, *Patching*, *Fixing*, *Failing*, *Infected*, *Lmoved*, *Exfiltrated*, *Crashing.* The transitions to any of the left four states are triggered by attacker actions. Upon finishing the exploitation software, the attacker starts a sequence of actions to destroy the system security at least in terms of confidentiality, integrity, and availability. These actions include infecting the system, keeping itself persistent in the system, searching sensitive data and making these data benefit them. The last two actions are repeated, forming a loop. Each attacker action may crash the system, denoted by *Crashing* state. In addition, the software bugs, such as Mandelbugs [33], may lead to system failure, denoted by *Failing* state. If the system crashes or fails, it must be fixed immediately. During the fixing, both the defender and the attacker can do nothing to the system. We assume that as long as patch is ready, it must be deployed into the system immediately. Thus, for each attack action, *System fails* and *Patch ready* must be checked in the first place. Since each attacker action may crash the system, for each attack action, we will check whether the attacker action succeeds before a system state transition occurs.

We assume that when the system completes this fixing, there is no APT code in the system but the vulnerability still exists. Such constrain may be relaxed. We leave the modeling of the relaxed system for future work. The system is unavailable before system failure or crash is fixed. Upon completing recovery, the system enters into good state, denoting that the vulnerability does not exist in the system. But when the system fails or crashes, the strategy can only be deployed after the system failure or crash is fixed. Without loss of generality, the deployment process is assumed to never fail and is not affected by APT. In addition, the system is unavailable during the mitigation strategy deployment.

Survivability has been defined by ANSI T1A1.2 committee as the transient performance of a system after an undesirable event [34]. The metrics used to

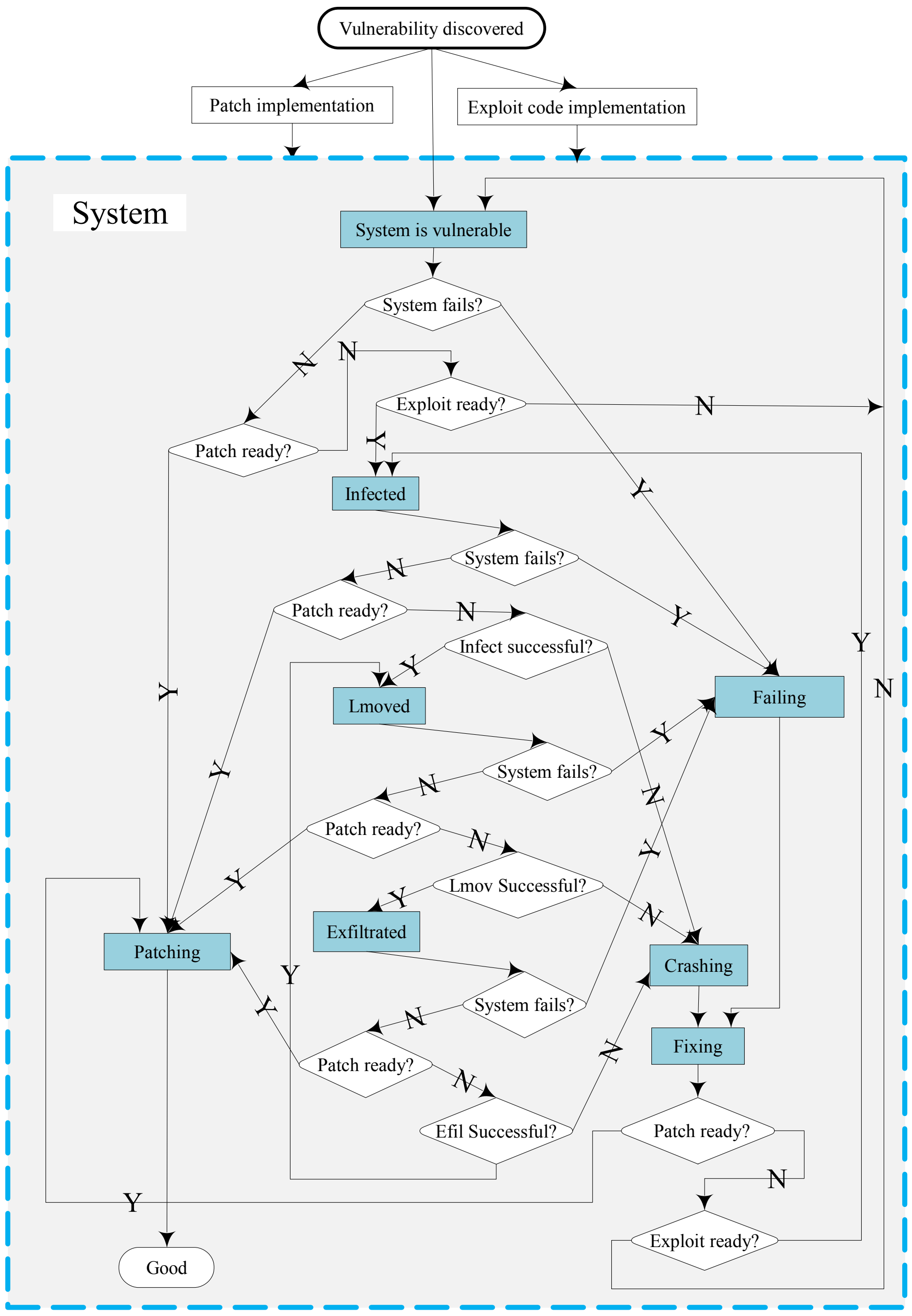


Fig. 1. Flowchart depicting events in a system under an APT attack, after a vulnerability is announced and during the mitigation strategy implementation.

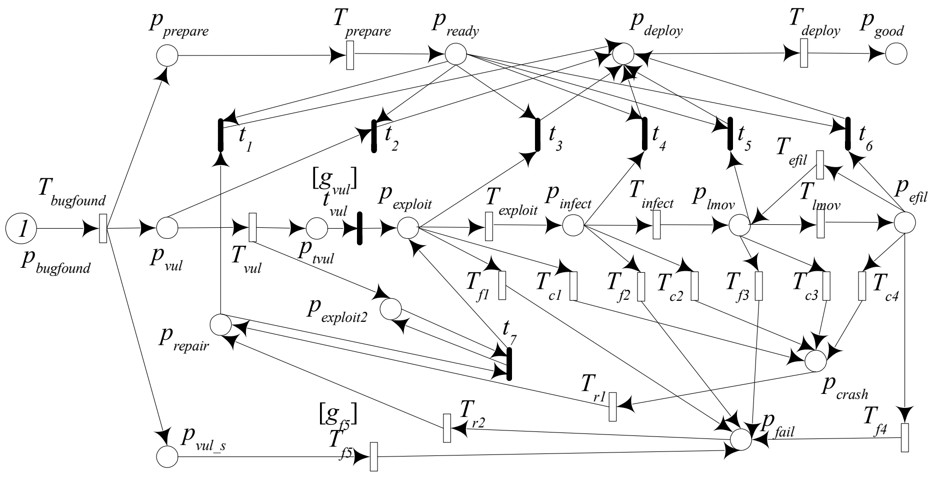


Fig. 2. Stochastic Reward Net model.

quantify survivability vary according to applications, and depend on a number of factors such as the minimum level of performance necessary for the system to be considered functional and the maximum acceptable security loss of a system. Performance levels are assigned as reward rates. In this paper, we classify the metrics into two broad categories:

– *Instantaneous metrics* are transient metrics that capture the state of the system at time t after the occurrence of an undesired event. An example of an instantaneous metric is the probability that the vulnerable system has been recovered at time t.
– *Cumulative metrics* are integrals of instantaneous metrics, that is, expected accumulated rewards in the interval $(0, t]$.

The metrics considered in this paper include

Metric m_1. Probability that the vulnerable system has been patched at time t;
Metric m_2. Probability that the system is unavailable at time t;
Metric m_3. Mean accumulated time that the system is unavailable in the interval $(0, t]$; and
Metric m_4. Mean accumulated loss of system confidentiality and integrity loss in the interval $(0, t]$.

Note that survivability metrics are transient metrics computed after the announcement of a vulnerability. In the remainder of this paper, time t refers to the time since a vulnerability is found and is measured in days.

3.2 Stochastic Reward Net Model

Fig. 2 shows an SRN model for the survivability analysis of a system under an APT attack. Table 1 shows the definition of variables, while Table 2 shows guard definitions. When a software vulnerability is identified, $T_{bugfound}$ fires with a quick rate δ. One token is removed from $p_{bugfound}$ and one token is put in p_{vul_s}, p_{vul}, and $p_{prepare}$ each, representing that system failure, exploitation code implementation and mitigation strategy implementation start concurrently.

When the exploit code is ready, T_{vul} fires. Then, one token is taken from p_{vul} and one token is deposited in $p_{exploit_2}$ and one token is deposited in p_{tvul}. Guard function g_{vul} determines whether or not a token is put in $p_{exploit}$. Only when the system does not fail (represented by a token in p_{vul_s}), t_{vul} fires and a token is put in $p_{exploit}$. That is, the number of tokens in place $p_{exploit_2}$ is used for determining whether the exploit code is ready after the system failure or crash is fixed. Places $p_{exploit}$, p_{infect}, p_{lmov}, and p_{efil} represent the status of the attacker in the system. When $T_{exploit}$ fires, one token is taken from $p_{exploit}$ and one token is put in p_{infect}, representing that exploit code is injected into the system successfully with mean time $1/(\lambda_{exploit} \cdot \rho_1)$. This injection process may fail resulting in the system crash, represented by firing T_{c_1}. For this situation, one token is taken from $p_{exploit}$ and one token is put in p_{crash}. The firing rate is $(1-\rho_1)\cdot\lambda_{exploit}$. In addition, the system may fail due to other reasons, represented by firing T_{f_1} with mean time $1/\lambda_{fail}$. That is, one token is taken from $p_{exploit}$ and one token is put in p_{fail}. Similar explanations are applicable to transitions from p_{infect}, p_{lmov}, and p_{efil}.

A token in place $p_{prepare}$ denotes the condition that the mitigation strategy is under implementation. When $T_{prepare}$ fires, one token is taken from $p_{prepare}$ and one token is put in p_{ready}, representing that the strategy is ready for deployment. When there is a token in p_{ready} and p_{repair} ($p_{vul}, p_{exploit}, p_{infect}, p_{lmov}$, or p_{efil}), the immediate transition t_1 (t_2, t_3, t_4, t_5, or t_6, respectively) fires. Then, a token is taken from p_{ready} and p_{repair} ($p_{vul}, p_{exploit}, p_{infect}, p_{lmov}$, or p_{efil}) and deposited in place p_{deploy}. The place p_{deploy} represents that the system begins the deployment of the mitigation strategy. When T_{deploy} fires, one token is taken from p_{deploy} and one token is put in p_{good}, representing that the system completes the mitigation strategy deployment and enters into state `GOOD`.

The priority of t_1 over t_7 aims to achieve the following goal: when the mitigation strategy and exploit code are both available, the mitigation strategy must be deployed immediately. Then, t_1 is fired and one token is taken from p_{ready} and p_{repair} each, and one token is put in p_{deploy}. If the strategy is not ready but the exploit code is available, then t_7 is fired. At this time, one token is taken from p_{repair} and $p_{exploit_2}$ each, and one token is put in $p_{exploit}$ and $p_{exploit_2}$. If both are unavailable, then t_7 does not fire and the token is kept in p_{repair}. Later, when mitigation strategy or exploit code is available, transition t_1 or t_7 fires, respectively.

Based on this model, we use the SPNP software package [35] to calculate the four metrics mentioned at the end of Section 3.1 as follows:

– The value of m_1 at time t is the *expected number of tokens of* p_{good} *at time* t.
– The value of m_2 at time t is the *expected number of tokens of* $(p_{crash}+p_{fail}+p_{deploy})$ *at time* t.

Symbol	Definition	Mean value
$1/\delta$	Mean time that the discovered vulnerability is known to all	30 min
$1/\lambda_{prepare}$	Mean time for implementing a mitigation strategy	20 days
$1/\lambda_{deploy}$	Mean time for installing the mitigation strategy	12 days
$1/\lambda_{vuln}$	Mean time for generating the exploit code	4 days
$1/\lambda_{fail}$	Mean time that the computer system fails	365 days
$1/\lambda_{fix}$	Mean time that the computer system completes the failure or crash fixing	2 days
$1/\lambda_{efil}$	Mean time that the attacker obtains the desired information	2 days
$1/\lambda_{exploit}$	Mean time for injecting the exploit code into the system	7 days
$1/\lambda_{inf}$	Mean time that the exploit code is persistent	1 days
$1/\lambda_{lmov}$	Mean time that the attacker finds sensitive data of interest	7 days
ρ_1	Probability that the exploit code works in the system	0.6
ρ_2	Probability that the exploit code is persistent	0.6
ρ_3	Probability that the attacker finds its target	0.6
ρ_4	Probability that the attacker obtains the desired information	0.6

Table 1. Definition of variables used in monolithic SRN model.

Guard Values	
g_{vul}	**if** $(\#(p_{vul_s}) == 1)$ **then** 1 **else** 0
g_{f_5}	**if** $(\#(p_{vul}) == 1)$ **then** 1 **else** 0

Table 2. Guard functions for the SRN model.

- The value of m_3 in the interval $(0, t]$ is the *expected accumulated reward of* $(p_{crash} + p_{fail} + p_{deploy})$ *by time t.*
- The value of m_4 in the interval $(0, t]$ is the *expected accumulated reward of* p_{exfil} *by time t.*

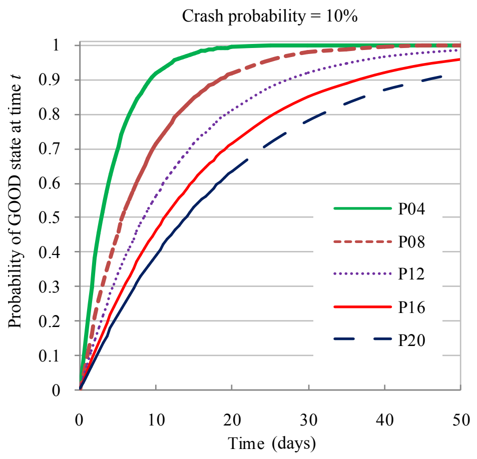
4 Numerical Results and Discussions

In this section we present the numerical results obtained using SPNP software package [35] to solve the SRN model. In particular, we report the metrics previously described.

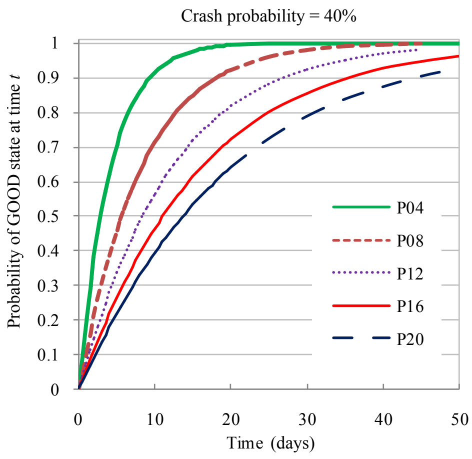
$1/\lambda_{prepare}$ is set to 20 days according to [36] and our analysis of vulnerability data set of Google Project Zero security team [37]. Similarly, $1/\lambda_{vuln}$ is set to 4 days according to [38]. Values of the other parameters now are unknown to us and are set based on intuition for sensitivity analysis. Table 1 gives the default values considered of each parameter.

We first investigate the effect of $\lambda_{prepare}$ on both the transient probability of GOOD state and the probability that the system is unavailable (that is, m_1 and m_2 metrics, respectively). Fig. 3 and 4 plot these results, respectively. Probabilities ρ_1, ρ_2, ρ_3, and ρ_4 are set to the value denoted by *crash probability*. P04, P08, P12 , P16, and P20 represent the results of $1/\lambda_{prepare} = 4$ days, 8 days, 12 days, 16 days, 20 days respectively. We carry out numerical analysis under crash probability of 10% and 40%.

From Fig. 3(a) and (b), we observe that crash probability has little effect on the probability of `GOOD` state for each $\lambda_{prepare}$ in our parameter configurations. The reason is that as long as the mitigation strategy becomes ready, the system can immediately enter into the deployment phase no matter which state of $p_{vuln}, p_{exploit}, p_{infect}, p_{lmov}$, and p_{exfil} is. However, the influence of $\lambda_{prepare}$ value is significant. The larger $\lambda_{prepare}$, the larger increase in the probability of `GOOD` state at time t. That is, the smaller $1/\lambda_{prepare}$ is, the quicker the system is recovered.



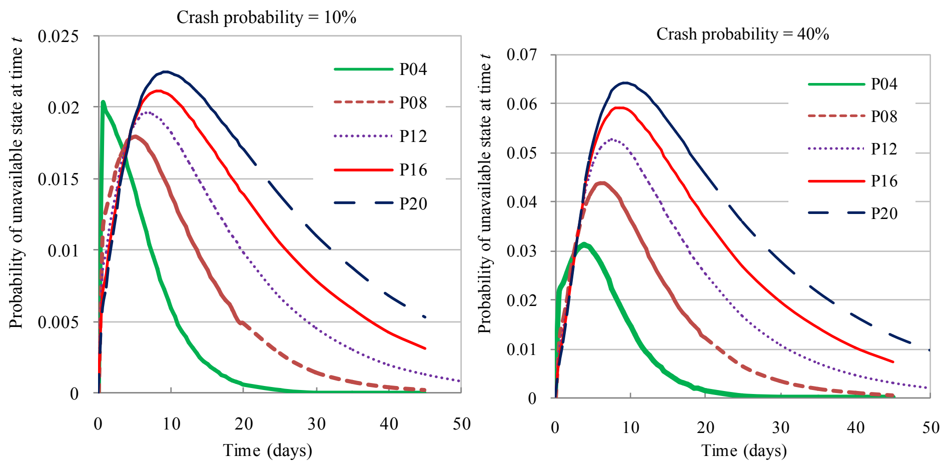
(a) Crash probability of 10%



(b) Crash probability of 40%

Fig. 3. Probability of `GOOD` state at time t under different crash probabilities (metric m_1).

However, Fig. 4(a) and (b) indicate that both crash probability and $\lambda_{prepare}$ have obvious effects on the probability that the system is unavailable at time t. This probability increases first and then decreases with t. The reason is as follows. At the beginning, only the system failure with very small rate contributes the probability of unavailable system. When the exploitation code is ready, the system crashes frequently. In addition, when the mitigation strategy is ready, the strategy deployment also contributes to the probability of unavailable system. Thus, this probability increases first and later decreases with the system entering into state `GOOD`. Fig. 4(a) and (b) indicate that in most time, the larger $1/\lambda_{prepare}$, the larger probability of unavailable system at time t. However, it is not hold at the beginning: we can observe that the smaller $1/\lambda_{prepare}$, the larger probability of unavailable system at time t. This is due to the probability of `DEPLOY` state. Fig. 5(a) and (b) show the probabilities of `CRASH+FAIL` and `DEPLOY` states, respectively, when crash probability is 10%. The same discussions apply to the results in Fig. 6, which depicts the mean accumulated time of un-



(a) Crash probability of 10% (b) Crash probability of 40%

Fig. 4. Probability of unavailable system at time t under different crash probabilities (metric m_2).

available system under different crash probabilities (metric m_3). Table 3 defines the reward rate functions used.

Measure	**Definition**
System Unavailability	1: **if** ($\#(p_{fail})$ == 1 **or** $\#(p_{crash})$ == 1 **or** $\#(p_{deploy})$ == 1)
	0: otherwise
Confidentiality loss	1: **if** ($\#(p_{efil})$ == 1)
	0: otherwise

Table 3. Reward rate definition.

Metric m_4 defined in Section 3.1 can be used for computing loss of confidentiality+integrity, integrity, or confidentiality. It depends on the attacker objective. Without loss of generality, we consider the confidentiality in the experiments. The system confidentiality loss per day is defined as 1. Fig. 7(a) and (b) plot the mean accumulated loss of system confidentiality by time t, under different crash probabilities (metric m_4). We can see that the larger $1/\lambda_{prepare}$ and/or the smaller crash probability, the larger mean accumulated time and loss.

5 Conclusions and Future Work

Cyberattacks are increasing rapidly according to numerous security vendor reports. These attacks affect normal operations of computer systems leading to large periods of unavailability, unauthorized data modification, or exfiltration of sensitive data. Furthermore, these attacks become specially critical when infrastructures that provide essential services are targeted, since disruptions or

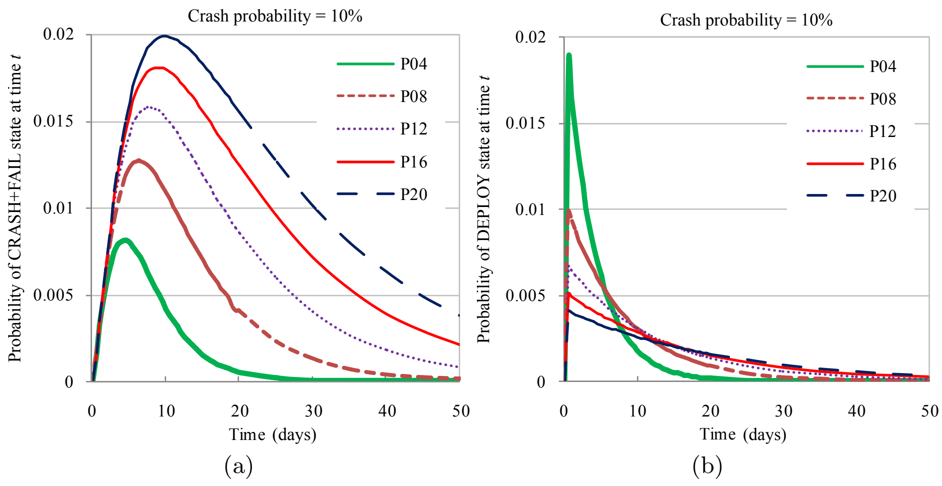
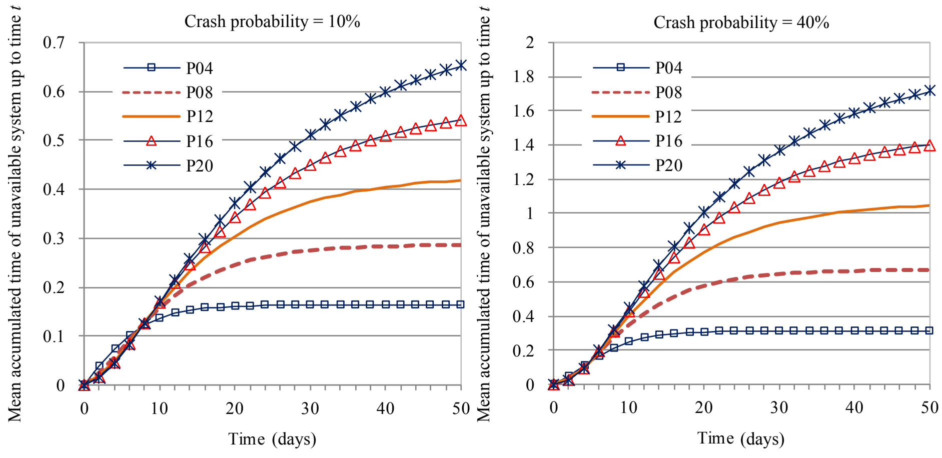
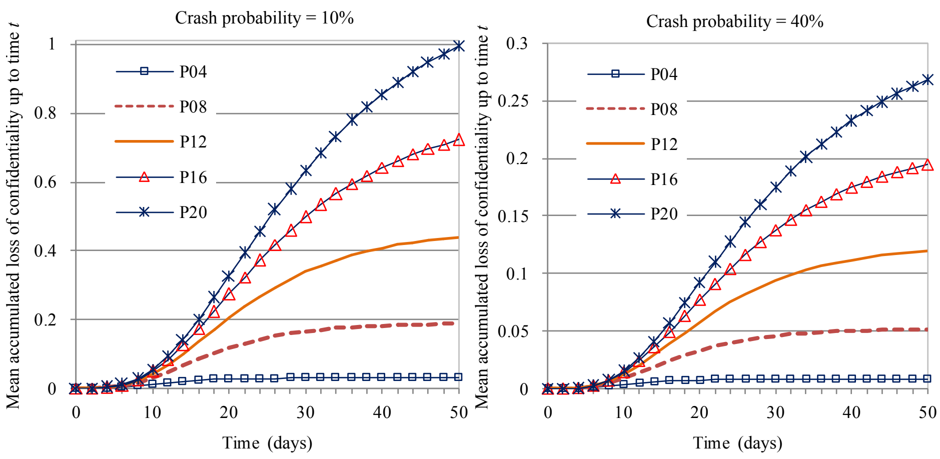


Fig. 5. Probability of (a) `CRASH+FAIL` and (b) `DEPLOY` state at time t under crash probability of 10%.



(a) Crash probability of 10% (b) Crash probability of 40%

Fig. 6. Mean accumulated time that the system is unavailable under different crash probabilities (metric m_3).



(a) Crash probability of 10% (b) Crash probability of 40%

Fig. 7. Mean accumulated of system confidentiality and integrity loss by time t under different crash probabilities (metric m_4).

malfunctioning of services may lead to fatalities or injuries. Malicious software specially crafted to target a specific system are known as Advanced Persistent Threats (APTs). An APT aim at compromising the security of the targeted system and gather sensitive data or steal intellectual property, among other goals.

This paper explored the CTMC model-based survivability analysis of a computer system under an APT attack. A variant of stochastic Petri nets (in particular, Stochastic Reward Nets) was used to automate the generation and solution of the Markov model. We defined four survivability metrics, in terms of system recovery, system availability, data confidentiality loss, and data integrity loss. In addition, numerical results have been presented to study the impact of the underlying parameters on the system survivability. These results may also provide insights on the cost/benefit trade-offs of investment efforts in system recovery strategies including vulnerability mitigation schemes.

There are several future work directions. This paper does not consider the security improvement schemes which could reduce the security loss during the system recovery from a vulnerability. These security schemes include using blacklist/whitelist to enforce system access control, using backup software and so on. Extending to our proposed model to capture these schemes and then quantitatively evaluating the abilities of various security protection schemes is our next research. We also plan to extend our survivability-based model to the scenario where multiple vulnerabilities are found and some event times are non-exponentially distributed. Furthermore, the modeling in this paper is an approximation of the real restoration process. We shall extend the current study to approximate the model in a more accurate way to real system behaviors.

Acknowledgments

The research of Ricardo J. Rodríguez was supported by the Spanish MINECO project CyCriSec (TIN2014-58457-R). The research of Xiaolin Chang was supported by NSF 61572066 of China. The research of Xiaodan Li and Kishor S. Trivedi was supported in part by US NSF grant number CNS-1523994, by IBM under a faculty grant, by NATO under Science for Peace project number 984425, and by US Navy under grant N00174-16-C-0036.

References

1. Symantec: Internet Security Threat report 2013. [Online] `http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v18_2012_21291018.en-us.pdf`.
2. Emm, D., Garnaeva, M., Ivanov, A., Makrushin, D., Unuchek, R.: IT Threat Evolution in Q2 2015. Technical report, Kaspersky Lab (July 2015)
3. McAfee: McAfee Labs Threats Report. Technical report (August 2015)
4. Department of Homeland Security: National Security Strategy. The White House (May 2010) Available at `http://www.whitehouse.gov/sites/default/files/rss_viewer/national_security_strategy.pdf`.
5. Kozik, R., Choras, M.: Current Cyber Security Threats and Challenges in Critical Infrastructures Protection. In: Proceedings of the 2nd International Conference on Informatics and Applications (ICIA). (September 2013) 93–97
6. Walters, R.: Cyber Attacks on U.S. Companies in 2014. The Heritage Foundation – National Security and Defense (4289) (October 2014) 1–5 Issue Brief.
7. Moser, A., Kruegel, C., Kirda, E.: Exploring Multiple Execution Paths for Malware Analysis. In: Proceedings of the IEEE Symposium on Security and Privacy. (2007) 231–245
8. Bayer, U., Habibi, I., Balzarotti, D., Kirda, E., Kruegel, C.: A View on Current Malware Behaviors. In: Proceedings of the 2nd USENIX Conference on Large-scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More (LEET), Berkeley, CA, USA, USENIX Association (2009) 1–11
9. Sood, A., Enbody, R.: Targeted Cyberattacks: A Superset of Advanced Persistent Threats. IEEE Security & Privacy **11**(1) (Jan 2013) 54–61
10. Tankard, C.: Advanced Persistent threats and how to monitor and deter them. Network Security **2011**(8) (2011) 16–19
11. Farwell, J.P., Rohozinski, R.: Stuxnet and the Future of Cyber War. Survival **53**(1) (2011) 23–40
12. Rauscher, K.: Writing the Rules of Cyberwar. IEEE Spectrum **50**(12) (2013) 30–32
13. Ellison, R.J., Fisher, D.A., Linger, R.C., Lipson, H.F., Longstaff, T.A., Mead, N.R.: Survivability: protecting your critical systems. IEEE Internet Computing **3**(6) (November 1999) 55–63
14. Bolch, G., Greiner, S., de Meer, H., Trivedi, K.S.: Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications. Second edn. Wiley-Interscience (April 2006)
15. Ramani, S., Trivedi, K.S., Dasarathy, B.: Performance Analysis of the CORBA Event Service using Stochastic Reward Nets. In: Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems (SRDS). (2000) 238–247

16. Philip, A., Sharma, R.K.: A stochastic reward net approach for reliability analysis of a flexible manufacturing module. International Journal of System Assurance Engineering and Management **4**(3) (2013) 293–302
17. Bruneo, D.: A Stochastic Model to Investigate Data Center Performance and QoS in IaaS Cloud Computing Systems. IEEE Transactions on Parallel and Distributed Systems **25**(3) (March 2014) 560–569
18. Entezari-Maleki, R., Trivedi, K.S., Movaghar, A.: Performability Evaluation of Grid Environments Using Stochastic Reward Nets. IEEE Transactions on Dependable and Secure Computing **12**(2) (March 2015) 204–216
19. Kumar, N., Lee, J.H., Chilamkurti, N., Vinel, A.: Energy-Efficient Multimedia Data Dissemination in Vehicular Clouds: Stochastic-Reward-Nets-Based Coalition Game Approach. IEEE Systems Journal **10**(2) (June 2016) 847–858
20. Kawamura, R., Ohta, H.: Architectures for ATM Network Survivability and Their Field Deployment. IEEE Commun. Mag. **37**(8) (August 1999) 88–94
21. Wylie, J.J., Bigrigg, M.W., Strunk, J.D., Ganger, G.R., Kiliccote, H., Khosla, P.K.: Survivable Information Storage Systems. Computer **33**(8) (Aug 2000) 61–68
22. Jha, S., Wing, J.M.: Survivability Analysis of Networked Systems. In: Proceedings of the 23rd International Conference on Software Engineering (ICSE). ICSE '01, Washington, DC, USA, IEEE Computer Society (2001) 307–317
23. Castet, J.F., Saleh, J.H.: On the concept of survivability, with application to spacecraft and space-based networks. Reliab. Eng. Syst. Saf. **99** (2012) 123–138
24. Paulauskas, N., Garsva, E., Gulbinovic, L., Stankevicius, A., Poviliauskas, D.: Survivability Modelling of Lithuanian Government Information System. Elektronika Ir Elektrotechnika **120**(4) (2012) 95–98 2012.
25. Wang, H., Liu, P.: Modeling and Evaluating the Survivability of an Intrusion Tolerant Database System. In Gollmann, D., Meier, J., Sabelfeld, A., eds.: Proceedings of the 11th European Symposium on Research in Computer Security (ESORICS), Berlin, Heidelberg, Springer Berlin Heidelberg (2006) 207–224
26. Wang, A.H., Yan, S., Liu, P.: A Semi-Markov Survivability Evaluation Model for Intrusion Tolerant Database Systems. In: Proceedings of the 2010 International Conference on Availability, Reliability, and Security (ARES). (February 2010) 104–111
27. Trivedi, K.S., Xia, R.: Quantification of system survivability. Telecommunication Systems **60**(4) (2015) 451–470
28. Rodríguez, R.J., Merseguer, J., Bernardi, S.: Modelling Security of Critical Infrastructures: A Survivability Assessment. The Computer Journal **58**(10) (2015) 2313–2327
29. Pfleeger, C.P., Pfleeger, S.L.: Security in Computing. 4th edn. Prentice Hall (2006)
30. Murata, T.: Petri Nets: Properties, Analysis and Applications. In: Proceedings of the IEEE. Volume 77. (April 1989) 541–580
31. Ajmone Marsan, M., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G.: Modelling with Generalized Stochastic Petri Nets. Wiley Series in Parallel Computing. John Wiley and Sons (1995)
32. Muppala, J., Ciardo, G., Trivedi, K.S.: Stochastic Reward Nets for Reliability Prediction. Communications in Reliability, Maintainability and Serviceability **1**(2) (1994) 9–20
33. Grottke, M., Trivedi, K.: Fighting Bugs: Remove, Retry, Replicate, and Rejuvenate. Computer **40**(2) (February 2007) 107–109
34. ANSI T1A1.2 Working Group on Network Survivability Performance: Enhanced Network Survivability Performance. Technical Report 68 (2001)

35. Ciardo, G., Muppala, J., Trivedi, K.: SPNP: stochastic Petri net package. In: Proceedings of the 3rd International Workshop on Petri Nets and Performance Models (PNPM). (December 1989) 142–151
36. Temizkan, O., Kumar, R., Park, S., Subramaniam, C.: Patch Release Behaviors of Software Vendors in Response to Vulnerabilities: An Empirical Analysis. J. Manage. Inf. Syst. **28**(4) (April 2012) 305–338
37. Google Project Zero: List of vulnerabilities reported by Google security research team. [Online] `https://bugs.chromium.org/p/project-zero/issues/list?can=1&q=&colspec=ID+Type+Status+Priority+Milestone+Owner+Summary&cells=ids.`
38. Nzoukou, W., Wang, L., Jajodia, S., Singhal, A.: A Unified Framework for Measuring a Network's Mean Time-to-Compromise. In: Proceedings of the 2013 IEEE 32nd International Symposium on Reliable Distributed Systems (SRDS). (September 2013) 215–224