Romuald Hoffmann[1]

Stochastyczne modele cyklu życia podatności oprogramowania

1. Wstęp

W trakcie eksploatacji oprogramowania uwidaczniają się podatności (ang. *software vulnerabilities*), które z punktu widzenia bezpieczeństwa systemów są istotnym zagrożeniem dla danych. Podatności oprogramowania traktować będziemy jako błędy oprogramowania wprowadzone na etapie jego specyfikacji, projektowania, implementacji lub konfiguracji, stanowiące jednocześnie podatności na atak cybernetyczny. Błędy te, nie powodując w normalnym trybie eksploatacji zaburzenia logiki funkcjonowania oprogramowania, stanowią bardzo istotne luki bezpieczeństwa systemu informatycznego, które można wykorzystać do ataku na system[2]. Atak może polegać na przejęciu kontroli nad nim lub użyciem go niezgodnie z wolą jego właściciela bądź użytkownika.

Szczególnie dzisiaj bezpieczeństwo systemów informatycznych jest jednym z najbardziej istotnych trosk ich użytkowników. Nie oznacza to, że jest to problem dnia dzisiejszego. Od 2. połowy lat 90. ubiegłego wieku, wraz z rozwojem informatyki, w odniesieniu do systemów informatycznych instytucji państwowych i prywatnych, a także oprogramowania urządzeń osobistych obserwuje się stale rosnącą liczbę incydentów związanych z wykorzystaniem podatności oprogramowania. Badacze zdali sobie sprawę, że podobnie jak w odniesieniu do systemów informatycznych, również w stosunku do podatności oprogramownia należy mówić o ich cyklu życia.

Cykl życia podatności oprogramowania, opierając się na dotychczasowych badaniach, ogólnie można podzielić na kilka faz, które zaczynają lub kończą się zdarzeniami: narodzin podatności (ang. *birth*, *creation*), wykrycia (ang. *discovery*),

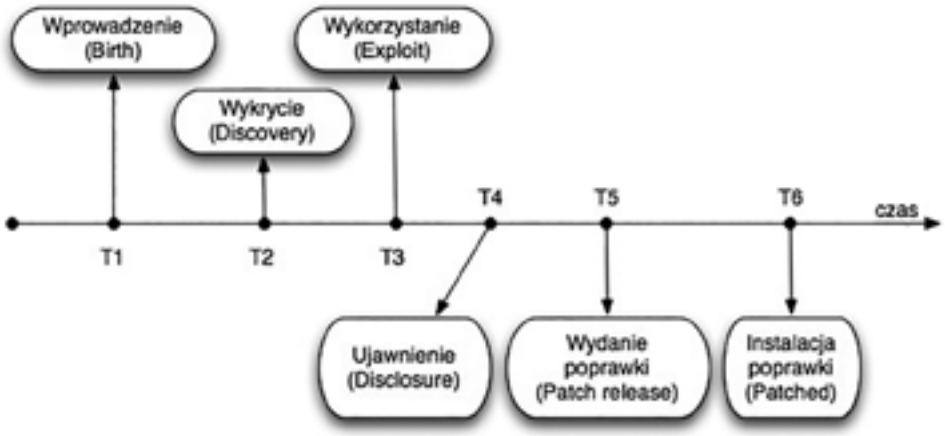
---

[1] Wojskowa Akademia Techniczna w Warszawie, Wydział Cybernetyki.

[2] R. Hoffmann, J. Stanik, J. Napiórkowski, *Modele wykrywania podatności oprogramowania w ujęciu dynamiki systemowej*, „Roczniki Kolegium Analiz Ekonomicznych" 2017, z. 45, s. 201–212.

wykorzystania (ang. *exploit*), ujawnienia (ang. *disclosure*), udostępnienia poprawki oprogramowania (ang. *patch release*, *patch available*), zainstalowania poprawki (ang. *patched*, *patch installed*).

Ogólny model cyklu życia podatności przedstawiono na rysunku 1.



Rysunek 1. Ogólny model cyklu życia pojedynczej podatności oprogramowania

Źródło: opracowanie własne na podstawie: S.M. Rajasooriya, Ch.P. Tsokos, P.K. Kaluarachchi, *Stochastic Modelling of Vulnerability Life Cycle and Security Risk Evaluation*, „Journal of Information Security" 2016, vol. 7, no. 4, s. 269–279.

W pracy rozszerzono ogólny model cykl życia podatności, wprowadzając zdarzenie wydania sygnatury antywirusowej (rysunek 2). Uwzględniając to, zaproponowano dwa stochastyczne modele cyklu życia pojedynczej podatności stanowiące rozwinięcia modeli zaczerpniętych z literatury[3]. Opisane modele obrazują postrzeganie cyklu życia z dwóch punktów widzenia: użytkowników końcowych oprogramowania i specjalistów zajmujących się bezpieczeństwem systemów informatycznych.

[3] H. Joh, Y.K. Malaiya, *A Framework for Software Security Risk Evaluation using the Vulnerability Lifecycle and CVSS Metrics*, w: *Proceedings of the 2010 International Workshop on Risk and Trust in Extended Enterprises (RTEE'2010)*, IEEE, San Jose CA, USA 1–4 Nov. 2010, s. 430–434; H. Okamura, M. Tokuzane, T. Dohi, *Security Evaluation for Software System with Vulnerability Life Cycle and User Profiles*, w: *Procedings of 2012 Workshop on Dependable Transportation Systems/Recent Advances in Software Dependability (WDTS-RASD 2012)*, B. Werner (red.), IEEE, Niigata, Japan 18–19 Nov. 2012, s. 39–44.

2. Rozszerzony cykl życia podatności oprogramowania

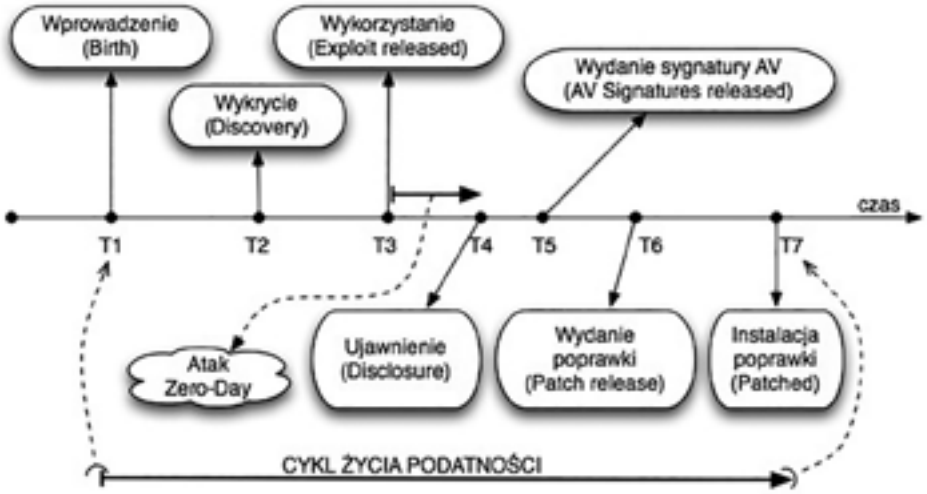
W literaturze[4] istnieją opisy cyklu życia podatności oprogramowania uwzględniające w mniejszym lub większym stopniu wymienione wcześniej zdarzenia: narodzin podatności, wykrycia, wykorzystania, ujawnienia, udostępnienia poprawki, zainstalowania poprawki oprogramowania. Należy zaznaczyć, że już w jednej z pierwszych prac[5] na przełomie ubiegłego i obecnego wieku, w której określono cykl życia podatności oprogramowania, autorzy uwzględnili narodziny (wydarzenie wprowadzenia podatności na etapie tworzenia oprogramowania), wykrycie, ujawnienie luki (określonej jako wewnętrzne rozpowszechnienie informacji w kręgu osób dokonujących zabezpieczenia systemów), wydanie poprawki, publiczne ujawnienie luki w zabezpieczeniach, dostępności „exploitów" i śmierci luki utożsamianej z zainstalowaniem poprawki. Pomimo tego, że w literaturze podobnie opisuje się cykl życia podatności, to jednak zdarzają się istotne różnice. Różnice w podejściu można prześledzić chociażby na przykładzie pracy[6] z roku 2005, gdzie wydanie oraz zainstalowanie poprawki traktuje się alternatywnie, i oba te zdarzenia zamykają proponowany cykl życia podatności.

Należy zauważyć, że dotychczasowe modele cyklu życia nie uwzględniają istnienia systemów antywirusowych, które w istotny sposób wpływają na ekspozycję na ryzyko wykorzystania podatności. W artykule uwzględnimy fakt udostępniania przez producentów systemów antywirusowych sygnatur oprogramowania złośliwego wykorzystującego podatności. Wobec tego przyjmiemy, że cykl życia podatności będzie składał się z faz wyznaczanych przez zdarzenia: narodziny, wykrycie, wykorzystanie, ujawnienie podatności, udostępnienie sygnatury antywirusowej, wydanie poprawki, zainstalowanie poprawki (rysunek 2).

[4] W.A. Arbaugh, W.L. Fithen, J. McHugh, *Windows of vulnerability: A case study analysis*, „IEEE Computer" 2000, vol. 33, no. 12, s. 52–59; E. Rescorla, *Is finding security holes a good idea?*, „IEEE Security and Privacy" Jan.–Feb. 2005, vol. 3, no. 1, s. 14–19; J. Jones, *Estimating software vulnerabilities*, „IEEE Security & Privacy" July–Aug. 2007, vol. 5, no. 4, s. 28–32; S. Frei, *Security econometrics – the dynamics of (in) security*, w: *Dissertation 18197*, ETH Zurich, Zurich 2009.

[5] W.A. Arbaugh, W.L. Fithen, J. McHugh, op. cit.

[6] E. Rescorla, op. cit.



Rysunek 2. Rozszerzony cykl życia pojedynczej podatności oprogramowania

Źródło: opracowanie własne.

3. Stochastyczne modele cyklu życia podatności oprogramowania

W ostatniej dekadzie, w wyniku badania stochastycznej natury cyklu życia podatności oprogramowania, zaproponowano[7] kilka probabilistycznych modeli cyklu. Dotychczas opublikowane modele, bazujące na jednorodnych procesach Markowa, z czasem ciągłym lub dyskretnym oraz ze skończoną liczbą stanów, odnoszą się do ogólnego cyklu życia (rysunek 1). W pracy przy założeniu istnienia własności Markowa i jednorodności w czasie zaproponowano dwa modele stochastyczne odnoszące się do proponowanego w niniejszym artykule rozszerzonego cyklu życia podatności (rysunek 2).

[7] H. Joh, Y.K. Malaiya, *A Framework for Software…*; H. Joh, Y.K. Malaiya, *Defining and Assessing Quantitative Security Risk Measures Using Vulnerability Lifecycle and CVSS Metrics*, w: *Proceedings of The 2011 International Conference on Security and Management (SAM'11)*, H.R. Arabnia, M.R. Grimaila, G. Markowsky (red.), vol. 1, Las Vegas Nevada, USA 18–21 July 2011, s. 10–16; H. Okamura, M. Tokuzane, T. Dohi, op. cit.; S.M. Rajasooriya, Ch.P. Tsokos, P.K. Kaluarachchi, *Stochastic Modelling of Vulnerability Life Cycle and Security Risk Evaluation*, „Journal of Information Security" 2016, vol. 7, no. 4, s. 269–279.

W podrozdziałach przyjęto następującą konwencję oznaczeń. Przez S_i oznaczono stan podatności w jej cyklu życia, gdzie i oznacza kolejny numer stanu ($i = 0,1,2,...$). $p_{ij}(t)$, λ_{ij} oznaczają odpowiednio: prawdopodobieństwo przejścia w chwili $t \geq 0$, intensywność (niezależną od czasu) przejścia ze stanu S_i do stanu S_j. Natomiast $P_i(t)$ oznacza prawdopodobieństwo przebywania procesu Markowa w stanie S_i w chwili $t \geq 0$. Do graficznego zobrazowania proponowanych modeli wykorzystano grafy procesu Markowa[8], które jednocześnie jednoznacznie opisują zarówno macierz przejścia pomiędzy stanami, jak i równania Kołmogorowa. W celu zwiększenia czytelności w węzłach umieszczono obok symbolu stanu S_i jego znaczenie[9].

3.1. Model z punktu widzenia projektanta

Proponowany model w założeniach uwzględnia charakterystyczny sposób postrzegania cyklu życia podatności przez osoby zajmujące się bezpieczeństwem systemów informatycznych. Model zakłada, że proces rozszerzonego cyklu życia podatności będzie przebywał w następujących stanach:

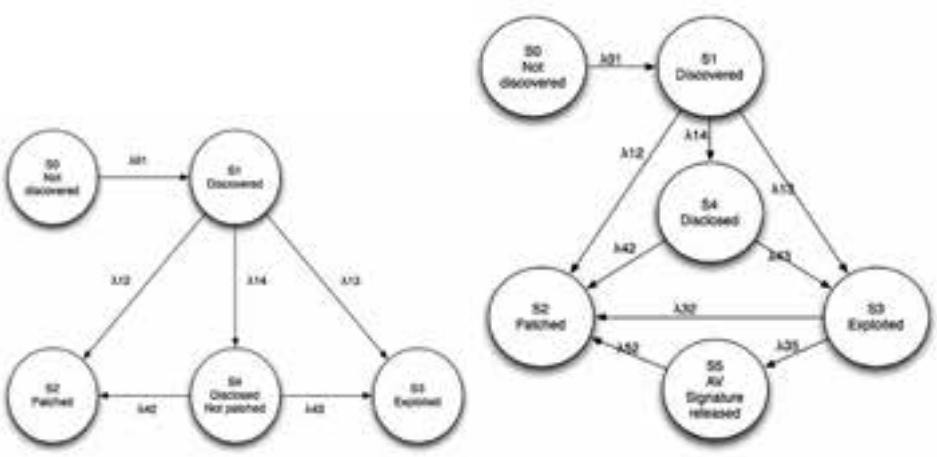
- S_0: podatność niewykryta (ang. *not discovered*) – podatność została wprowadzona na etapie implementacji i nie została ujawniona wcześniej podczas eksploatacji;
- S_1: podatność wykryta (ang. *discovered*) – podatność została wykryta, w tym również możliwe, że przez atakującego;
- S_2: poprawka została zainstalowana (ang. *patched*) – użytkownik zainstalował aktualizację bezpieczeństwa;
- S_3: wykorzystanie (ang. *exploited*) – oprogramowanie złośliwe zostało stworzone i podatność została wykorzystana przez atakującego;
- S_4: ujawnienie (ang. *disclosure*) – podatność została ujawniona i jednocześnie podana do publicznej wiadomości;
- S_5: sygnatura antywirusowa została udostępniona (ang. *antivirus signature released*) – sygnatura oprogramowania złośliwego została opracowana i udostępniona, i nie stworzono innej wersji „exploita" przed zainstalowaniem poprawki.

[8] G.F. Lawler, *Introduction to Stochastic processes*, 2nd Edition, Chapman and Hall/CRC Taylor and Francis Group, London, New York 2006.

[9] Przyjęto anglojęzyczny opis stanu w celu zachowania jednoznaczności w odniesieniu do literatury.

Modelem opisanej sytuacji jest jednorodny proces Markowa z czasem ciągłym i skończoną liczbą stanów. Macierz intensywności przejścia między stanami tego łańcucha ma postać:

$$\Lambda = \begin{bmatrix} -\lambda_{00} & \lambda_{01} & 0 & 0 & 0 & 0 \\ 0 & -\lambda_{11} & \lambda_{12} & \lambda_{13} & \lambda_{14} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_{32} & -\lambda_{33} & 0 & \lambda_{35} \\ 0 & 0 & \lambda_{42} & \lambda_{43} & -\lambda_{44} & 0 \\ 0 & 0 & \lambda_{52} & 0 & 0 & -\lambda_{55} \end{bmatrix}, \text{ gdzie } \lambda_{ij} \geq 0,\ \lambda_{ii} = \sum_{\substack{j=0 \\ j \neq i}}^{6} \lambda_{ij} \qquad (1)$$



a) model Joh-Malaiya[10] b) model proponowany

Rysunek 3. Model z punktu widzenia projektanta – graf stanów procesu Markowa

Źródło: opracowanie własne.

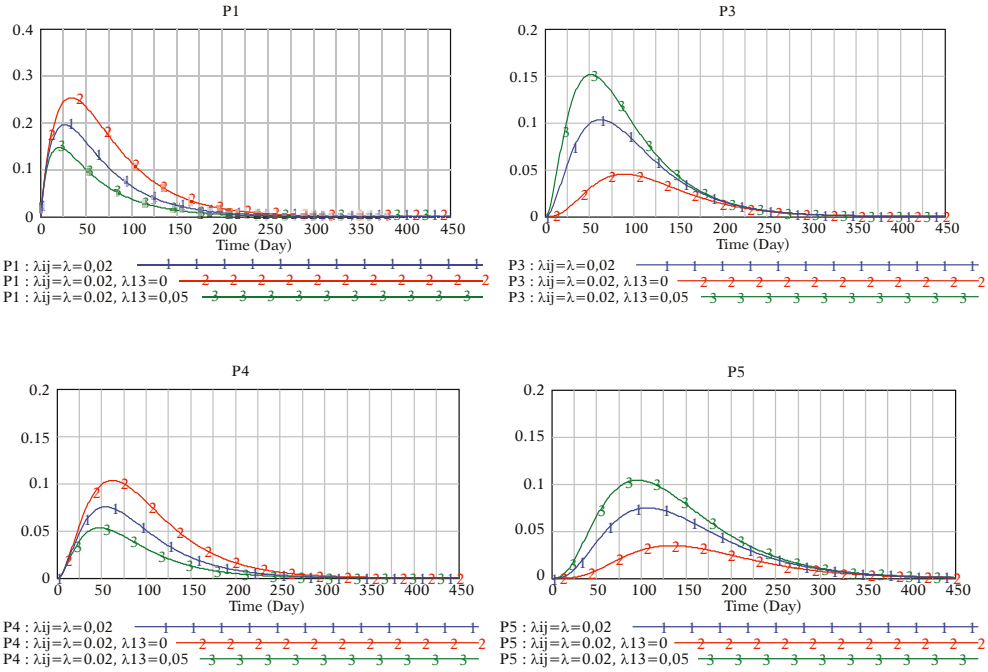
Macierz (1) intensywności Λ można jednoznacznie przedstawić w postaci grafu stanów procesu Markowa[11] (rysunek 3b). Ukazany w celu porównania na rysunku 3a graf Markowa odnosi się do modelu wyjściowego Joh-Malaiya[12].

Układ równań Kołmogorowa pozwalający na wyznaczenie wektora rozkładu prawdopodobieństw procesu Markowa z macierzą intensywności Λ jest postaci:

10 H. Joh, Y.K. Malaiya, *A Framework for Software Security Risk…*
11 Zwany również grafem Markowa.
12 Ibidem.

$$\left[\begin{array}{ccc} \frac{d}{dt}P_0(t) & \cdots & \frac{d}{dt}P_5(t) \end{array} \right] = \left[\begin{array}{ccc} P_0(t) & \cdots & P_5(t) \end{array} \right] \cdot \Lambda, \; t \geq 0 \qquad (2)$$

z warunkiem początkowym $P_0(0) = 1,\ P_i(0) = 0$ dla $i = 1,2,...,5$.



Rysunek 4. Przykładowe rozwiązania układu równań Kołmogorowa dla modelu z punktu widzenia projektanta

Źródło: opracowanie własne.

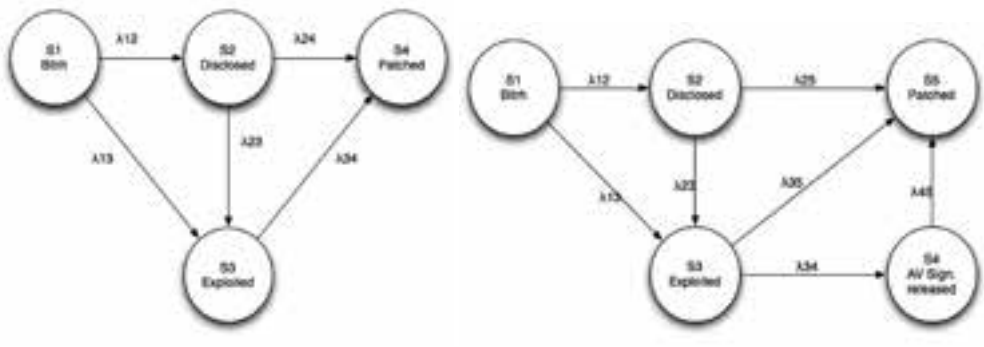
Rozwiązanie układu równań (2) pozwala na wyznaczenie poszczególnych prawdopodobieństw $P_i(t)$. Szczególnie istotne jest wyznaczenie niezbędnego do szacowania ryzyka prawdopodobieństwa wykorzystania podatności przez atakującego $P_3(t)$. Układ równań został rozwiązany numerycznie[13], a przykładowe wyniki symulacji ilustruje rysunek 4.

[13] Z wykorzystaniem pakietu symulacyjnego Vensim® ver. 5 firmy Ventana Systems, Inc.

3.2. Model z punktu widzenia użytkownika końcowego

W tym miejscu rozważymy model uwzględniający rozszerzony cykl życia podatności oprogramowania widziany z perpektywy użytkownika końcowego (rysunek 5b). Model zakłada, że proces cyklu życia podatności będzie przebywał w następujących stanach:

- S_1: narodziny podatności (ang. *birth*, *creation*) – podatność została wprowadzona na etapie implementacji i nie została ujawniona wcześniej;
- S_2: ujawnienie (ang. *disclosure*) – podatność została ujawniona i jednocześnie podana do publicznej wiadomości;
- S_3: wykorzystanie (ang. *exploited*) – oprogramowanie złośliwe zostało stworzone i podatność została wykorzystana;
- S_4: sygnatura antywirusowa udostępniona (ang. *antivirus signature released*) – sygnatura oprogramowania złośliwego została opracowana i udostępniona i nie stworzono innej wersji „exploita" przed zainstalowaniem poprawki;
- S_5: poprawka zainstalowana (ang. *patched*, *patch installed*) – użytkownik zainstalował aktualizację bezpieczeństwa.



a) model Okamura-Tokuzane-Dohi[14] b) model proponowany

Rysunek 5. Model z punktu widzenia użytkownika – graf Markowa

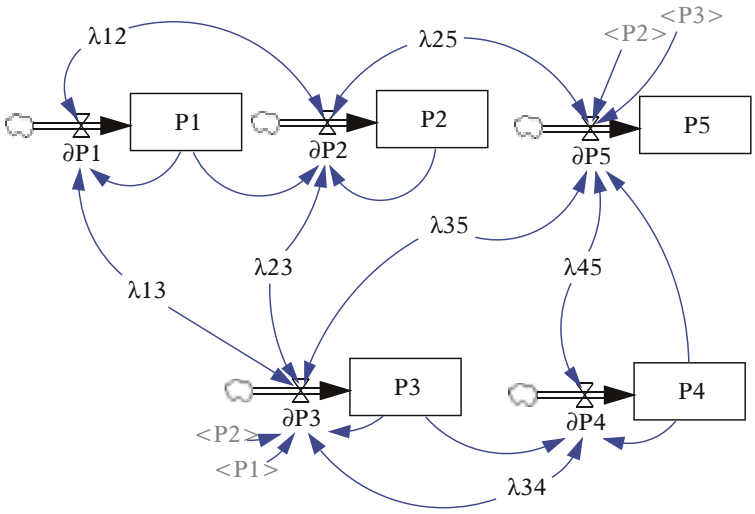
Źródło: opracowanie własne.

Podobnie jak w poprzednim paragrafie, zgodnie z przyjętymi założeniami modelem jest jednorodny proces Markowa z czasem ciągłym i skończoną liczbą stanów, gdzie macierz intensywności Λ przedstawiona jest w postaci grafu stanów procesu Markowa (rysunek 5b). Ukazany na rysunku 5a graf Markowa

[14] Ibidem.

odnosi się do modelu Okamura-Tokuzane-Dohi[15] nawiązujacego do ogólnego modelu cyklu życia podatności.

Analogicznie jak w poprzednim paragrafie, również dla tego modelu numeryczne rozwiązanie układu równań Kołmogorowa (określonego na podstawie grafu Markowa – rysunek 5b) przy warunku początkowym $P_1(0) = 1$, $P_i(0) = 0$ dla $i = 2,...,5$ pozwala na wyznaczenie poszczególnych prawdopodobieństw $P_i(t)$.



∂P1=- (λ12+λ13) *P1
∂P2=λ12*P1–(λ23+λ25)*P2
∂P3=λ13*P1+λ23*P2–(λ34+λ35)*P3
∂P4=λ34*P3–λ45*P4
∂P5=λ25*P2+λ35*P3+λ45*P4

P1= INTEG (∂P1,1)
P2= INTEG (∂P2,0)
P3= INTEG (∂P3,0)
P4= INTEG (∂P4,0)
P5= INTEG (∂P5,0)

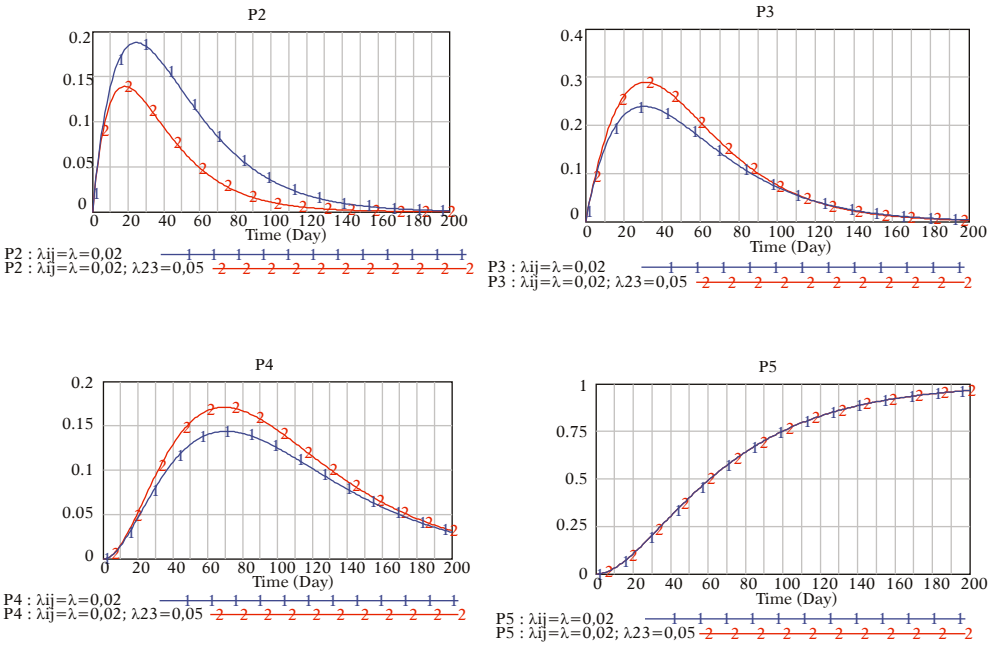
Rysunek 6. Równania Kołmogorowa w języku dynamiki systemowej

Źródło: opracowanie własne.

Układ równań Kołmogorowa zdefiniowany w języku dynamiki systemowej[16] przedstawia rysunek 6, a uzyskane wyniki obliczeń prezentuje rysunek 7. Obliczeń numerycznych dokonano, wykorzystując do tego celu pakiet symulacyjny dynamiki systemowej Vensim® ver. 5 firmy Ventana Systems, Inc.

[15] H. Okamura, M. Tokuzane, T. Dohi, op. cit.

[16] J.D. Sterman, *Business Dynamics. Systems Thinking and Modeling for a Complex World*, McGraw-Hill, Boston 2000; R. Hoffmann, T. Protasowicki, *Metoda dynamiki systemowej w modelowaniu złożonych systemów i procesów*, „Biuletyn Instytutu Systemów Informatycznych" 2013, nr 12, s. 19–28.

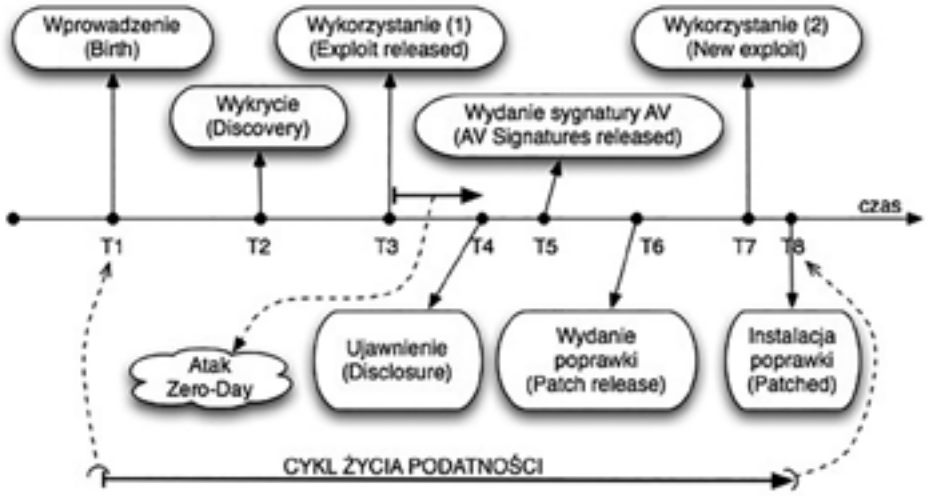


Rysunek 7. Przykładowe rozwiązania układu równań Kołmogorowa dla modelu z punktu widzenia użytkownika końcowego

Źródło: opracowanie własne.

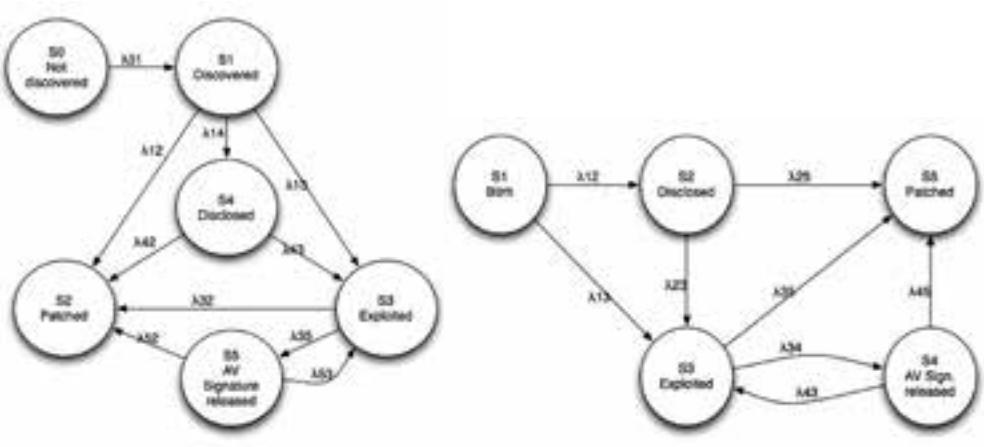
4. Podsumowanie i kierunki dalszych badań

Zaprezentowane probabilistyczne modele cyklu życia podatności oprogramowania można dalej rozbudowywać. Na przykład wystarczy zauważyć, że po wydaniu sygnatury antywirusowej i udostępnieniu poprawki bezpieczeństwa, ale jeszcze przed jej instalacją, atakujący może stworzyć inną wersję oprogramowania złośliwego niewykrywalnego przez skanery bezpieczeństwa. Następnie tak przygotowane oprogramowanie wykorzystać do ataku. Zatem cykl życia podatności oprogramowania, uwzględniający ten przypadek, ilustruje rysunek 8, a grafy stanów procesów Markowa dla rozszerzonych modeli przyjmują postać jak na rysunku 9. Na tej podstawie wyznaczenie wartości poszczególnych prawdopodobieństw $p_{ij}(t)$, $P_i(t)$, w tym szczególnie wartości prawdopodobieństwa wykorzystania podatności przez atakującego $P_3(t)$ w chwili t pozwala na szacowanie ryzyka bezpieczeństwa rozpatrywanego oprogramowania w założonym czasie.



Rysunek 8. Rozszerzony cykl życia pojedynczej podatności oprogramowania uwzględniający wprowadzenie nowej wersji kodu złośliwego (tzw. exploita)

Źródło: opracowanie własne.



a) z punktu widzenia projektanta b) z punktu widzenia użytkownika

Rysunek 9. Model w przypadku użycia innego „exploita" – graf Markowa

Źródło: opracowanie własne.

Reasumując, zaproponowane stochastyczne modele cyklu życia podatności rozpatrywane w kontekście cyklu życia konkretnego systemu informatycznego mogą posłużyć do opracowania metod i zaplanowania minimalizacji ryzyka

wykorzystania przez atakującego luk bezpieczeństwa. W konsekwencji pozwolą również końcowemu użytkownikowi nie tylko na szacowanie ryzyka bezpieczeństwa eksploatowanych systemów, ale także na analizę porównawczą oprogramowania potencjalnych systemów w procesie zakupu.

Bibliografia

Arbaugh A., Fithen W.L., McHugh J., *Windows of vulnerability: A case study analysis*, „IEEE Computer" 2000, vol. 33, no. 12, s. 52–59.

Frei S., *Security econometrics – the dynamics of (in) security*, w: *Dissertation 18197*, ETH Zurich, Zurich 2009.

Joh H., Malaiya Y.K., *A Framework for Software Security Risk Evaluation using the Vulnerability Lifecycle and CVSS Metrics*, w: *Proceedings of the 2010 International Workshop on Risk and Trust in Extended Enterprises (RTEE'2010)*, IEEE, San Jose CA, USA 1–4 Nov 2010, s. 430–434.

Joh H., Malaiya Y.K., *Defining and Assessing Quantitative Security Risk Measures Using Vulnerability Lifecycle and CVSS Metrics*, w: *Proceedings of The 2011 International Conference on Security and Management (SAM'11)*, H.R. Arabnia, M.R. Grimaila, G. Markowsky (red.), vol. 1, Las Vegas Nevada, USA 18–21 July 2011, s. 10–16.

Hoffmann R., Protasowicki T., *Metoda dynamiki systemowej w modelowaniu złożonych systemów i procesów*, „Biuletyn Instytutu Systemów Informatycznych" 2013, nr 12, s. 19–28.

Hoffmann R., Stanik J., Napiórkowski J., *Modele wykrywania podatności oprogramowania w ujęciu dynamiki systemowej*, „Roczniki Kolegium Analiz Ekonomicznych SGH" 2017, z. 45, s. 201–212.

Jones J., *Estimating software vulnerabilities*, „IEEE Security & Privacy" July–Aug. 2007, vol. 5, no. 4, s. 28–32.

Lawler G.F., *Introduction to Stochastic processes*, 2nd Edition, Chapman and Hall/CRC Taylor and Francis Group, London, New York 2006.

Okamura H., Tokuzane M., Dohi T., *Security Evaluation for Software System with Vulnerability Life Cycle and User Profiles*, w: *Procedings of 2012 Workshop on Dependable Transportation Systems/Recent Advances in Software Dependability (WDTS-RASD 2012)*, B. Werner (red.), IEEE, Niigata, Japan 18–19 Nov. 2012, s. 39–44.

Rajasooriya S.M., Tsokos Ch.P., Kaluarachchi P.K., *Stochastic Modelling of Vulnerability Life Cycle and Security Risk Evaluation*, „Journal of Information Security" 2016, vol. 7, no. 4, s. 269–279.

Rescorla E., *Is finding security holes a good idea?*, „IEEE Security and Privacy" Jan.–Feb. 2005, vol. 3, no. 1, s. 14–19.

Sterman J.D., *Business Dynamics. Systems Thinking and Modeling for a Complex World*, McGraw-Hill, Boston 2000.

* * *

Stochastic Models of Software Vulnerability Life Cycle

Summary

Software Vulnerability Life Cycle (SVLC) illustrates changes in the detection process of software vulnerability during the system exploitation. In the detection process generally two groups of actors can be distinguished: the potential exploiters and the patch developers. In this paper, there was proposed an expansion of SVLC by adding events of an anti-virus signature release and a new exploit execution to the existing general definition of vulnerability life cycle. The presented approach in this article models the extended software vulnerability life cycle as a stochastic process: a continuous time Markov chain. Consequently, there were proposed two stochastic models of the expanded vulnerability life cycle. The models can be used for evaluating the risk of vulnerability exploitation and information system security.

Keywords: software vulnerability life cycle, SVLC, expanded software vulnerability life cycle, stochastic model, homogeneous Markov process, Markov Chain, Continuous Time Markov Chain, CTMC, system dynamics.