Romuald Hoffmann[1], Jerzy Stanik[2], Jarosław Napiórkowski[3]

Modele wykrywania podatności oprogramowania w ujęciu dynamiki systemowej

1. Wstęp

Zawsze tam, gdzie funkcjonuje oprogramowanie, mogą wystąpić tzw. luki (ang. *vulnerabilities*) bezpieczeństwa oprogramowania lub inaczej – podatności oprogramowania. Badacze w literaturze różnie definiują to pojęcie. Na potrzeby niniejszego artykułu podatność oprogramowania będziemy rozumieć jako wykryte i zinwentaryzowane miejsce w kodzie lub konfiguracji oprogramowania, które można wykorzystać do przejęcia kontroli nad nim lub użyć niezgodnie z wolą jego właściciela lub użytkownika. Luki w oprogramowaniu, które najczęściej uwidaczniają się na etapie eksploatacji, traktuje się jako błędy oprogramowania wprowadzone na etapie jego specyfikacji, projektowania lub implementacji, z konfiguracją włącznie. Błędy te jednak nie powodują w normalnym trybie eksploatacji zaburzenia logiki funkcjonalności oprogramowania.

W dotychczas prowadzonych badaniach poszukuje się m.in. metod i modeli ilościowych, które mogą wspomóc osiągnięcie założonego poziomu bezpieczeństwa w wyniku właściwej alokacji zasobów do testowania, aktualizacji i łatania oprogramowania. W literaturze anglojęzycznej wskazano kilka modeli pod ogólną nazwą *Vulnerability Discovery Models* (VDM), umożliwiających estymację liczby podatności w oprogramowaniu w czasie. Modele te wyrosły na gruncie modeli niezawodnościowych oprogramowania, w tym głównie wzrastającej niezawodności oprogramowania SRGM (ang. *Software Reliability Growth Models*). Stąd też modele VDM należy postrzegać jako specjalizowane modele niezawodnościowe oprogramowania typu SRGM, które wcześniej zaproponowano do prognozowania całkowitej liczby defektów oprogramowania wykrytych podczas

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

procesu testowania i usuwania błędów[4]. Modele VDM podzielono na dwie kategorie – *Time-Based Models* oraz *Effort-Based Models*. Pierwsza grupa modeli służy do predykcji w czasie łącznej liczby podatności, druga natomiast pozwala na predykcję łącznej liczby podatności w zależności od udziału w rynku i liczby użytkowników. W artykule odniesiemy się tylko do modeli ilościowych w funkcji czasu, opierając się na modelach: Andersona (2002), Rescorli (2005) oraz Alhazmiego–Malaiyi (2005), dla których zaproponowano w niniejszym artykule odpowiedniki sformułowane w języku dynamiki systemowej. Wszystkie modele przedstawione w niniejszym artykule zostały zaimplementowane przy pomocy pakietu Vensim® ver. 5, autorstwa Ventana Systems Inc.

2. Krótka charakterystyka metody dynamiki systemowej

Metoda dynamiki systemowej (ang. *system dynamics*) jest metodą budowy modeli symulacji ciągłej, która umożliwia modelowanie struktury oraz dynamiki złożonych systemów i procesów w nich zachodzących. Została zaproponowana w latach 60. XX w. przez J. Forrestera, który opracował jej podstawowe zasady podczas swojej pracy w Sloan School of Management i przedstawił je w licznych publikacjach[5]. Obecnie bardzo dobrym wprowadzeniem do metod dynamiki systemowej jest praca J.D. Stermana[6]. Metoda dynamiki systemowej jest przeznaczona do modelowania złożonych systemów, w których występują tzw. sprzężenia zwrotne, opisujące zależności przyczynowo-skutkowe pomiędzy elementami systemu[7].

---

[4] A.L. Goel, K. Okumoto, *A time dependent error detection model for software reliability and other performance measures*, „IEEE Transactions on Reliability" 1979, vol. R-28, s. 206–211; S. Yamada, M. Ohba, S. Osaki, *S-shaped reliability growth modeling for software error detection*, „IEEE Transactions on Reliability" 1983, vol. R-32, s. 475–484; J.D. Musa, K. Okumoto, *A logarithmic Poisson execution time model for software reliability measurement*, w: *Proc. of the 7th International Conf. on Software Engineering, ICSE '84*, IEEE Press, Piscataway (New Jersey) 1984, s. 230–238.

[5] J.W. Forrester, *Industrial Dynamics*, MIT Press, Cambridge 1961; J.W. Forrester, *The collected papers of Jay W. Forrester*, Wright-Allen Press, Cambridge 1975; J.W. Forrester, *Urban Dynamics*, MIT Press, Cambridge 1969.

[6] J.D. Sterman, *Business Dynamics: Systems Thinking and Modeling for a Complex World*, McGraw-Hill, New York 2000.

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

W modelach matematycznych zbudowanych z wykorzystaniem metody dynamiki systemowej wyróżniamy równania: poziomów (opisane równaniami różniczkowymi pierwszego rzędu), przepływów (dane równaniami algebraicznymi) i zmiennych pomocniczych (określone jako równania algebraiczne). Z równań tych otrzymujemy układ równań różniczkowo-algebraicznych, stanowiący opis matematyczny związków przyczynowo-skutkowych występujących w modelowanym systemie. W metodzie dynamiki systemowej są stosowane równania różniczkowe zwyczajne pierwszego rzędu wyprowadzone z ogólnej postaci zagadnienia Cauchy'ego[8]: $dY(t)/dt = F(Y,\alpha,t), Y(t_0) = Y_0$, gdzie: $Y(t)$, Y jest wektorem zmiennych stanu, α – wektorem parametrów modelu, t – zmienną niezależną (czasem), Y_0 – wektorem wartości początkowych dla wektora Y.

3. Modele wykrywania podatności oprogramowania

W opisie modeli będziemy rozumieć przez: $Y(t)$ – łączną liczbę wykrytych podatności do chwili t, natomiast przez $y(t) = dY(t)/dt$ – wskaźnik (stopę) wykrywalności podatności w chwili t. $Y(t)$ i $y(t)$ dla dowolnego $t \geq 0$ są związane zależnością $Y(t) = \int y(t)\,dt$ lub $Y(t) = \int_0^t y(\tau)\,d\tau$. Te same oznaczenia, bez utraty precyzji, będą odnosić się do modeli wykrywania podatności oprogramowania zaczerpniętych z literatury oraz do modeli dynamiki systemowej przedstawionych dalej. Przy opisie poszczególnych modeli przyjęto konwencję przedstawienia modeli w języku dynamiki systemowej na rysunkach umieszczonych tuż pod opisem analitycznym poszczególnych modeli.

3.1. Model termodynamiczny Andersona

Model termodynamiczny został zaproponowany przez R. Andersona w 1999 r. jako model niezawodności oprogramowania. Później został przez

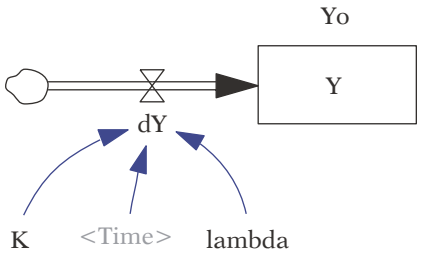
R. Hoffmann, T. Protasowicki, *Modelowanie pola walki z zastosowaniem koncepcji dynamiki systemowej*, „Biuletyn Instytutu Systemów Informatycznych" 2013, vol. 12, s. 29–34.

[8] E. Kasperska, *Dynamika systemowa. Symulacja i optymalizacja*, Wydawnictwo Politechniki Śląskiej, Gliwice 2005.

niego zaadaptowany[9] jako model opisujący proces wykrywania podatności oprogramowania. Historycznie jest to pierwszy model opisujący ilościowo proces wykrywania podatności oprogramowania. Zakłada on, że wykryte podatności oprogramowania zostaną usunięte i nie zostaną wprowadzone nowe błędy programowe, w tym podatności. Anderson dowodzi, że $y(t) \le K/t$, gdzie K jest wielkością stałą. Ponadto argumentuje, że $y(t)$ można przedstawić następującym wzorem:

$$y(t) = \frac{K}{\lambda \cdot t}, \text{ dla } t > 0, \tag{1}$$

gdzie λ jest współczynnikiem, do którego ustalenia brana jest pod uwagę zmniejszająca się stopa błędów (awarii) dla fazy beta testów w porównaniu z fazą alfa testów.

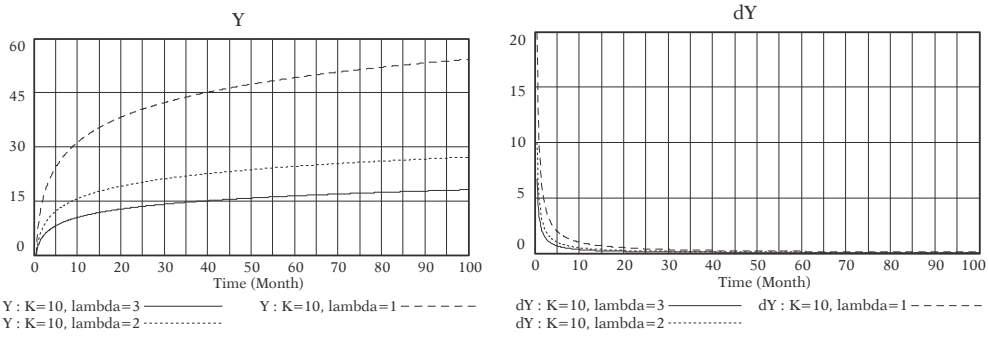


(01) dY = K/(lambda*Time)
(02) FINAL TIME = 100
(03) INITIAL TIME = 0.5
(04) K = 10
(05) lambda = 1
(06) SAVEPER = TIME STEP
(07) TIME STEP = 0.125
(08) Y = INTEG (dY, Yo)
(09) Yo = 0

Rysunek 1. Model termodynamiczny Andersona w języku dynamiki systemowej
Źródło: opracowanie własne.

Biorąc powyższe pod uwagę, po wykonaniu operacji całkowania wielkości $y(t)$ daną wzorem (1) otrzymujemy rozwiązanie ogólne $Y(t) = K \cdot \ln(C \cdot t) / \lambda$ dla każdego $t > 0$.

[9] R. Anderson, *Security in Open versus Closed Systems – The Dance of Boltzmann, Coase and Moore*, w: *Conference on Open Source Software Economics*, Cambridge University, Toulouse 2002.



Rysunek 2. Model termodynamiczny Andersona – wynik symulacji

Źródło: opracowanie własne.

Definicja w języku dynamiki systemowej modelu termodynamicznego Andersona oraz wyniki przykładowej symulacji zostały przedstawione na rysunkach 1 i 2.

3.2. Model liniowy i wykładniczy Rescorli

E. Rescorla[10] identyfikował trendy procesów wykrywania podatności oprogramowania, analizując dane historyczne z wykorzystaniem testów statystycznych dotyczące systemów operacyjnych Windows NT 4.0, Solaris 2.5.1, FreeBSD 4.0 i ReadHat 6.2.

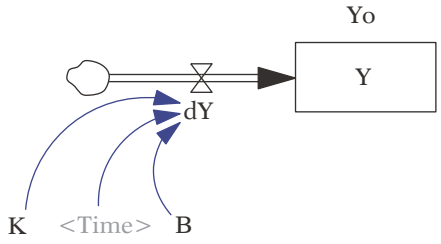
3.2.1. Model liniowy Rescorli

Pierwszy z nich to model liniowy dany wzorem:

$$y(t) = K + B \cdot t \ , \ t \geq 0. \tag{2}$$

Całka ogólna równania (2), opisująca łączną liczbę wykrytych podatności w funkcji czasu, przedstawia się następująco: $Y(t) = \int (B \cdot t + K) dt$. Ostatecznie $Y(t) = \frac{B \cdot t^2}{2} + K \cdot t + C$. Jeśli przyjmiemy $C = 0$, tak aby łączna liczba podatności w chwili $t = 0$ wynosiła zero, $Y(t)$ przyjmuje postać: $Y(t) = \frac{B \cdot t^2}{2} + K \cdot t$, $t \geq 0$.

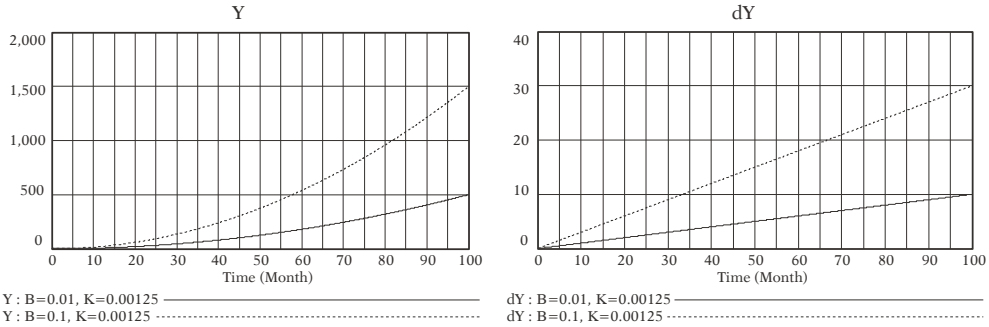
[10] E. Rescorla, *Is finding security holes a good idea?*, „Security and Privacy" 2005, January–February, s. 14–19.



(01) B = 0.1
(02) dY = B*Time+K
(03) FINAL TIME = 100
(04) INITIAL TIME = 0
(05) K = 0.00125
(06) SAVEPER = TIME STEP
(07) TIME STEP = 0.125
(08) Y = INTEG(dY,Yo)
(09) Yo = 0

Rysunek 3. Model liniowy Rescorli w języku dynamiki systemowej

Źródło: opracowanie własne.



Rysunek 4. Model liniowy Rescorli – wynik symulacji

Źródło: opracowanie własne.

Definicja w języku dynamiki systemowej modelu liniowego Rescorli oraz wyniki przykładowej symulacji zostały przedstawione na rysunkach 3 i 4.

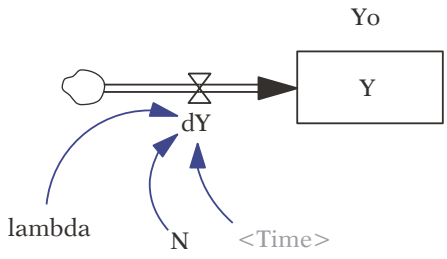
3.2.2. Model wykładniczy Rescorli

Rescorla wykorzystał model rosnącej niezawodności oprogramowania SRGM Goel–Okumoto[11] w celu dopasowania do zgromadzonych danych. Model wykładniczy Rescorli wykrywania znanych podatności oprogramowania w ogólnej postaci można przedstawić następująco:

$$y(t) = N \cdot \lambda \cdot e^{-\lambda \cdot t}, \tag{3}$$

[11] A.L. Goel, K. Okumoto, *Time-Dependent Error Detection Rate Model for Software and Other Performance Measures*, „IEEE Transactions on Reliability" 1979, vol. R-28, no. 3, August, s. 206–211.

gdzie: N oznacza całkowitą (maksymalną) liczbę podatności, które można wykryć w systemie; λ – stały współczynnik opisujący stopę wykrywania podatności oprogramowania.

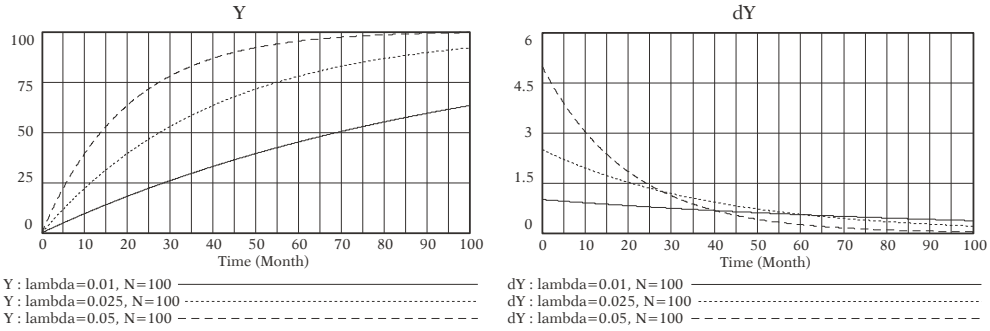


(01) dY = N*lambda/EXP(lambda*Time)
(02) FINAL TIME = 100
(03) INITIAL TIME = 0
(04) lambda = 0.01
(05) N = 100
(06) SAVEPER = TIME STEP
(07) TIME STEP = 0.125
(08) Y = INTEG(dY, Yo)
(09) Yo = 0

Rysunek 5. Model wykładniczy Rescorli w języku dynamiki systemowej

Źródło: opracowanie własne.

Po scałkowaniu $y(t)$ otrzymujemy skumulowaną liczbę wykrytych podatności oprogramowania w funkcji czasu. Całka ogólna równania (7), opisująca łączną liczbę wykrytych podatności w funkcji czasu, w modelu wykładniczym Rescorli przedstawia się następująco: $Y(t) = \int N \cdot \lambda \cdot e^{-\lambda \cdot t}\, dt = -N \cdot e^{-\lambda \cdot t} + C$. Stąd aby $Y(0) = 0$, przyjmujemy stałą całkowania równą N, $Y(t) = N \cdot (1 - e^{-\lambda \cdot t})$.



Rysunek 6. Model wykładniczy Rescorla – wynik symulacji

Źródło: opracowanie własne.

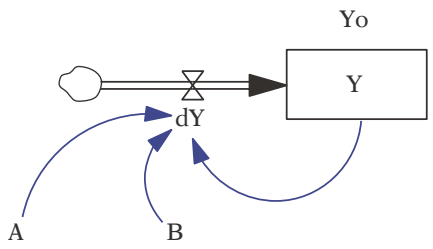
Definicja w języku dynamiki systemowej modelu wykładniczego Rescorla oraz wyniki przykładowej symulacji zostały przedstawione na rysunkach 5 i 6.

3.3. Model logistyczny Alhazmiego–Malaiyi

Model logistyczny Alhazmiego–Malaiyi[12] został zaproponowany w odniesieniu do podatności oprogramowania systemów operacyjnych Windows 98 i Windows NT 4.0. W modelu tym stopę wykrywalności znanych podatności oprogramowania w funkcji czasu (ang. *a time-based known-vulnerability discovery model*) opisuje następujące równanie różniczkowe:

$$\frac{d}{dt}Y(t) = A \cdot Y(t) \cdot (B - Y(t)),\ t \geq 0, \tag{4}$$

gdzie: czas (t) traktuje się jako czas kalendarzowy; $Y(t)$ oznacza skumulowaną liczbę podatności wykrytych łącznie do chwili t; A i B są stałymi wyznaczonymi empiryczne w wyniku analizy zgromadzonych danych historycznych; A interpretuje się jako intensywność wykrywania podatności; B oznacza łączną maksymalną liczbę podatności, które mogą zostać wykryte w oprogramowaniu. Rozwiązaniem ogólnym równania (4) jest funkcja $Y(t) = B/(B \cdot C \cdot e^{-A \cdot B \cdot t} + 1)$



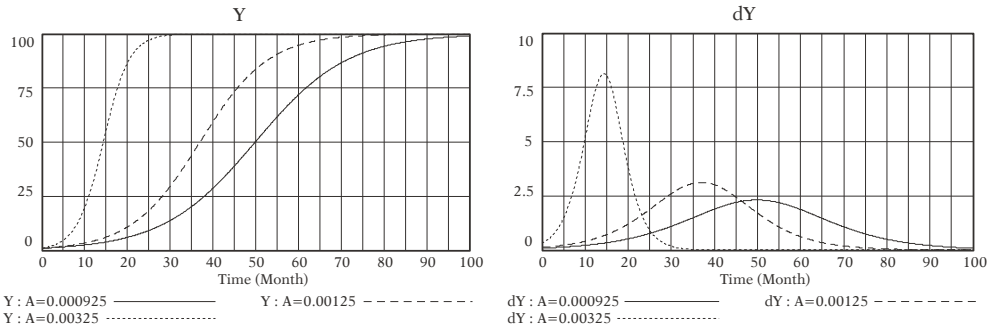
(01) A = 0.00125
(02) B = 100
(03) dY = A*Y*(B–Y)
(04) FINAL TIME = 100
(05) INITIAL TIME = 0
(06) SAVEPER = TIME STEP
(07) TIME STEP = 0.125
(08) Y = INTEG(dY, Yo)
(09) Yo=1

Rysunek 7. Model logistyczny Alhazmiego–Malaiyi w języku dynamiki systemowej
Źródło: opracowanie własne.

Model Alhazmiego–Malaiyi bazuje na obserwacji dojrzałości oprogramowania systemowego oraz stopnia uwagi (zainteresowania) poświęcanej eksploatacji systemu operacyjnego. W fazie wprowadzania na rynek nowego systemu operacyjnego lub nowej jego edycji zainteresowanie stopniowo wzrasta, aby później szybko „poszybować w górę" i ostatecznie w końcowej fazie osłabnąć

[12] O.H. Alhazmi, Y.K. Malaiya, *Quantitative Vulnerability Assessment of Systems Software*, w: *Annual Reliability and Maintainability Symposium, 2005. Proceedings*, IEEE, 2005, s. 615–620.

z powodu wprowadzenia nowej edycji. Stąd też stopień wykrywania podatności oprogramowania wolno rośnie w początkowej fazie wprowadzania na rynek, szybko rośnie – prawie liniowo – w drugiej fazie zwiększania się liczby instalacji (zwiększania się liczby użytkowników), aż w końcu w trzeciej fazie (końcowej) zaczyna maleć z powodu przechodzenia użytkowników na nowszą wersję lub znacznego zwiększenia niezawodności oprogramowania. Należy tutaj zaznaczyć, że ostatnia faza w przypadku danego systemu operacyjnego może nie nastąpić, jeżeli system nie będzie eksploatowany dostatecznie długo, np. z powodu wycofania z rynku.



Rysunek 8. Model logistyczny Alhazmiego–Malaiyi – wynik symulacji

Źródło: opracowanie własne.

Definicja w języku dynamiki systemowej modelu logistycznego Alhazmiego–Malaiyi oraz wyniki przykładowej symulacji zostały przedstawione na rysunkach 7 i 8.

4. Podsumowanie i kierunki dalszych badań

Praktyka pokazuje, że budowa złożonego systemu całkowicie pozbawionego luk bezpieczeństwa w oprogramowaniu jest z reguły niemożliwa. Wynika to z faktu, że jakość i bezpieczeństwo oprogramowania systemu są swoistym ciągiem kompromisów, trudnych do pogodzenia oczekiwań właściciela systemu, posiadacza przetwarzanych danych, użytkownika końcowego z warsztatem projektantów i programistów, a także presją związaną z budżetem i czasem. Stąd w praktyce eksploatacji oprogramowania dąży się do właściwego zarządzania

podatnościami, w tym ich wykrywania i usuwania podczas eksploatacji, a zaprezentowane modele dynamiczne mogą ułatwić to zadanie.

Chociaż dotychczasowe modele VDM nie służą identyfikacji podatnych komponentów (tak jak np. zaproponowali to S. Neuhaus[13] i Y. Shin)[14], ale do oceny profilu bezpieczeństwa oprogramowania jako całości, to jednak mogą mieć swoje praktyczne zastosowania. Mogą być wykorzystywane przez wytwórców i użytkowników oprogramowania do zrozumienia trendów zabezpieczeń, planowania poprawek i aktualizacji oprogramowania, a także np. planowania inwestycji w zabezpieczenia. Ponadto mogą być stosowane jako narzędzia ułatwiające ocenę oprogramowania i podjęcie decyzji dotyczących tego, które i jakie zastosować w swoich własnych systemach lub produktach. Modele te również mogą zostać użyte do analizy ryzyka bezpieczeństwa. Zastosowanie metod dynamiki systemowej w tym obszarze pozwala na bardzo elastyczne włączenie zdefiniowanych modeli do analiz procesów wytwórczych oprogramowania (bazując na modelach R.J. Madachy'ego[15]) oraz procesów bezpieczeństwa informacji zachodzących we współczesnych organizacjach.

Na zakończenie warto wspomnieć, że w odniesieniu do modelowania aspektów podatności oprogramowania należy rozważać dwa procesy – wykrywania podatności oprogramowania oraz eksploracji podatności przez atakującego. Z uwagi na ograniczoną objętość niniejszego artykułu odnieśliśmy się w nim tylko do procesu wykrywania podatności oprogramowania w zakresie charakterystyk ilościowych, bazując jedynie na wybranych modelach VDM. Zintegrowane modele łączące aspekty dwóch procesów – wykrywania i eksploracji podatności – będą stanowić bardzo interesujące pole badań. Metody dynamiki systemowej mogą, w opinii autorów, być skutecznym narzędziem modelowania w tym obszarze.

---

[13] S. Neuhaus, T. Zimmermann, C. Holler, A. Zeller, *Predicting vulnerable software components*, w: *Proc. of the 14th ACM Conf. on Computer and Comm. Security (CCS'07)*, 2007, s. 529–540.

[14] Y. Shin, A. Meneely, L. Williams, J.A. Osborne, *Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities*, „IEEE Transactions on Software Engineering" 2011, vol. 37(6), s. 772–787.

[15] R.J. Madachy, *Software Process Dynamics*, IEEE Press, Wiley, 2007.

Bibliografia

Alhazmi O.H., Malaiya Y.K., *Prediction Capabilities of Vulnerability Discovery Models*, w: *RAMS '06. Annual Reliability and Maintainability Symposium*, IEEE, Newport Beach 2006.

Alhazmi O.H., Malaiya Y.K., *Quantitative Vulnerability Assessment of Systems Software*, w: *Annual Reliability and Maintainability Symposium, 2005. Proceedings*, IEEE, Alexandria (Virginia) 2005, s. 615–620.

Alhazmi O.H., Malaiya Y.K., Ray I., *Vulnerabilities in Major Operating Systems, Technical Report*, Computer Science Department, Colorado State University, Fort Collins 2004.

Anderson R., *Security in Open versus Closed Systems – The Dance of Boltzmann, Coase and Moore*, w: *Conference on Open Source Software Economics*, Cambridge University, Toulouse 2002.

Goel A.L., Okumoto K., *Time-Dependent Error Detection Rate Model for Software and Other Performance Measures*, „IEEE Transactions on Reliability" 1979, vol. R-28, no. 3, August, s. 206–211.

Forrester J.W., *Industrial Dynamics*, MIT Press, Cambridge 1961.

Forrester J.W., *The collected papers of Jay W. Forrester*, Wright-Allen Press, Cambridge 1975.

Forrester J.W., *Urban Dynamics*, MIT Press, Cambridge 1969.

Handbook of Software Reliability Engineering, red. M.R. Lyu, McGraw-Hill, New York 1995.

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., Protasowicki T., *Modelowanie pola walki z zastosowaniem koncepcji dynamiki systemowej*, „Biuletyn Instytutu Systemów Informatycznych" 2013, vol. 12, s. 29–34.

Kasperska E., *Dynamika systemowa. Symulacja i optymalizacja*, Wydawnictwo Politechniki Śląskiej, Gliwice 2005.

Massacci F., Nguyen V.H., *An Empirical Methodology to Evaluate Vulnerability Discovery Models*, „IEEE Transactions on Software Engineering (TSE)" 2014, vol. 40(12), s. 1147–1162.

Musa J.D., *Software Reliability Engineering*, McGraw-Hill, New York 1999.

Musa J.D., Okumoto K., *A logarithmic Poisson execution time model for software reliability measurement*, w: *Proceeding ICSE '84, Proceedings of the 7th international conference on Software engineering*, IEEE Press, Piscataway (New Jersey) 1984, s. 230–238.

Rescorla E., *Is finding security holes a good idea?*, „Security and Privacy" 2005, January–February, s. 14–19.

* * *

Vulnerability Discovery Models in Terms of System Dynamics

Abstract

So far a few Vulnerability Discovery Models have been published. Such models will allow effective resource allocation for patch development and update as well as for software upgrade. The models are also needed for evaluating the risk of vulnerability exploitation and information security. Here are examined Anderson Thermodynamic Model (2002), Rescorla Linear and Exponential Model (2005) and Alhazmi–Malaiya Logistic Model for the vulnerability discovery process in terms of system dynamics as examples. The models are presented both analytically and in the system dynamics language. The applicability of the proposed models and their significance are shortly discussed.

Keywords: software vulnerability, Vulnerability Discovery Model, vulnerability management, system dynamics, system dynamics model