软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.000000]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向漏洞生命周期的安全风险度量方法*

胡 浩[1,3], 叶润国[4], 张红旗[1,3], 常德显[1,3], 刘玉岭[2]

[1](信息工程大学, 三院,河南 郑州 450001)

[2](中国科学院, 软件研究所, 可信计算与信息保障实验室,北京 100190)

[3](河南省信息安全重点实验室,河南 郑州 450001)

[4](中国电子技术标准化研究院,北京 100007)

通讯作者: 胡浩, E-mail: wjjhh_908@163.com

摘 要: 为了反映信息系统安全漏洞的风险随时间动态变化的规律, 构建基于吸收 Markov 链的漏洞生命周期模型, 计算先验历史漏洞信息作为模型输入, 构造漏洞生命周期的状态转移概率矩阵, 在时间维度上利用矩阵对状态演化过程进行推导。借鉴通用漏洞评分标准分析漏洞威胁影响, 给出安全漏洞的时间维度风险量化方法, 并对漏洞生命周期各状态发生概率的演化规律进行总结和分析。最后以典型 APT 攻击场景中“WannaCry”勒索病毒的漏洞利用过程为例, 验证了模型及方法的合理性和有效性。

关键词: 风险度量;漏洞生命周期;随机模型;吸收 Markov 链;动态评估

中图法分类号: TP393.8

中文引用格式: 胡浩,叶润国,张红旗,常德显,刘玉岭.基于漏洞生命周期的安全风险度量方法.软件学报. http://www.jos.org.cn/1000-9825/ 0000.htm

英文引用格式: Hu H, Ye RG, Zhang HQ, Chang DX, Liu YL. Vulnerability Life Cycle Oriented Security Risk Metric Method. Ruan Jian Xue Bao/Journal of Software, 2017 (in Chinese). http://www.jos.org.cn/1000-9825/0000.htm

Vulnerability Life Cycle Oriented Security Risk Metric Method

HU Hao[1,3], YE Run-Guo[4], ZHANG Hong-Qi[1,3], CHANG De-Xian[1,3], LIU Yu-Ling[2]

[1](The 3rd School, Information Engineering University, Zhengzhou 450001, China)

[2](Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

[3](Henan Key Laboratory of Information Security, Zhengzhou 450001, China)

[4](China Electronics Standardization Institute, Beijing 100007, China)

Abstract: In order to reflect dynamic change rules of vulnerability security risk with time evolution in the information system, a life cycle stochastic model based on the absorbing Markov chain is built. The prior historical vulnerability information is used as the input. Then the state transition probability matrix of vulnerability life cycle is constructed. Specifically, the state evolution process is simulated in the dimension of time using matrix deduction. Meanwhile, the common vulnerability scoring system (CVSS) is utilized to measure the threat impact of vulnerabilities in the network system. After that, a quantitative risk method to measure security vulnerability in terms of time dimension is given finally, through which some probabilities evolution rules with respect to the states of vulnerability life cycle are analyzed and summarized. In

* 基金项目: 国家“863”高技术研究发展计划基金项目 (2012AA012704，2015AA016006)；郑州市科技领军人才项目 (131PLJRC644)； 国家重点研发计划课题 (2016YFF0204003)；公安部信息网络安全重点实验室开放课题 (C15604)；“十三五”装备预研领域基金（61400020201）

收稿时间: 0000-00-00; 修改时间: 0000-00-00; 采用时间: 0000-00-00; jos 在线出版时间: 0000-00-00

CNKI 在线出版时间: 0000-00-00

the end, the exploits by the ransomware “WannaCry” in a typical APT attack scenario are taken as an example to verify the rationality and validity of our model as well as the proposed method.

Key words: security metric; vulnerability life cycle; stochastic model; absorbing Markov chain; dynamic evaluation

网络系统在提供各种便利的同时，也带来了诸多安全问题，目前基于漏洞利用的攻击渗透日益猖獗，对网络系统安全造成极大威胁，因此研究安全漏洞模型[1-7]，分析漏洞利用规律[8-10]，度量漏洞安全风险[11-19]，为主动防御提供理论基础和技术支撑，受到学术界和工业界的广泛关注和高度重视[20]。

在安全漏洞模型研究方面，陈恺等人[1]提出了一种多周期的漏洞发布模型，从宏观上预测漏洞数量与软件发布时间的关系。在此基础上，聂楚江等人[2]提出了一种微观漏洞数量预测模型，通过分析与软件代码相关的微观参数，并结合历史版本软件漏洞数量预测新版本中的漏洞数量。此外，高志伟等人[3]提出了基于漏洞严重度分类的预测模型，不仅能预测软件发布过程中存在的漏洞总数和时间间隔，而且能预测漏洞的种类及每一类漏洞的数量。Arbaugh 等人[4]最早提出了漏洞生命周期的概念，分析了漏洞从产生到消亡可能经历的几种状态，并结合美国 CERT/CC 报告，展示了历年来漏洞数量在不同状态上的分布情况。类似地，Frei[5]利用系统动力学对漏洞生命周期进行建模与分析，但实验数据集较小，且未分析软件厂商的影响。Kaaniche 等人[6]结合开源的 OSVDB 漏洞数据库分析了 Windows，Unix 和 Mobile OS 操作系统漏洞处于生命周期不同阶段的时间长度分布，结果表明该时间分布与具体操作系统类型相关。宋明秋等人[7]通过量化漏洞生命周期时间维的攻击热度与攻击技术，基于 Mamdani 模型，利用模糊推理法计算漏洞安全风险值。上述研究各有侧重地分析了漏洞数量在软件发布时间、漏洞类型和漏洞生命周期时间维上的分布情况，设计能够全面融合漏洞数量、生命周期、可利用性、威胁影响与风险度量的漏洞模型还有待进一步研究。

在漏洞利用规律分析方面，文献[8]提出了与 0-day 漏洞相关的漏洞生命周期的几种模式，梳理了 439 个 Linux 和 Microsoft 系统的漏洞信息，结果表明攻击脚本和漏洞补丁的扩散对漏洞利用概率变化具有重要影响。Joh 等人[9]将漏洞生命周期划分为漏洞产生、漏洞发现、漏洞公开、漏洞利用和漏洞修复 5 个阶段，指出安全漏洞在其生命过程中都存在被攻击者利用的风险，且不同阶段的安全风险大小不等。通过对早期报道的包含 46310 个漏洞的开源数据集 OSVDB 的统计分析，Shahzad 等人[10]展示了 1988 年-2011 年不同软件厂商的漏洞在各自生命周期不同阶段的数量分布情况，同时呈现了补丁发布周期以及漏洞利用平时耗时的变化趋势，结果表明漏洞利用耗时往往小于补丁的研发耗时。上述研究主要从统计学角度出发，基于大量数据的统计结果，试图展示漏洞利用概率和耗时在其生命周期内随软件供应商类型、补丁发布时间等影响因素的变化规律。如何从概率论角度出发，依据先验历史数据分析漏洞利用规律，并对未来趋势进行预测，对进一步提高研究的实用性意义重大。

在漏洞安全风险评估方面，对于具体漏洞攻击事件，文献[11]将攻击威胁的严重程度和攻击发生的可能性划分多个等级，通过建立风险矩阵，借鉴综合评判法分析系统的安全风险等级。Mkpong 等人[12]利用专家经验量化漏洞相关属性对系统的影响程度，提出了一种基于漏洞属性的风险量化模型。上述 2 种方法虽然计算简单，但主要依赖领域知识，具有很强的主观性，仅适用于分析已知漏洞的风险值。付志耀等人[13]设计了一种基于粗糙集理论的属性约简及漏洞影响评估方法，避免了漏洞关联属性集选择时，过分依赖先验的专家经验。Houmb 等人[14]利用通用漏洞系统（common vulnerability scoring system，CVSS）的标准数据集训练贝叶斯信念网络，并结合先验漏洞信息，推算漏洞被利用的可能性及威胁影响，能在一定程度上度量未公开漏洞的潜在风险。在实际应用方面，周亮等人[15]提出了基于关联网络的漏洞风险评估模型，利用层次分析法，对电力调度管理系统的安全漏洞风险进行了综合评估，类似应用还涉及安卓移动应用终端[16]，Web 应用程序[17]，电信通信设备[18]和电力工业控制系统[19]。不难看出，现有漏洞风险评估侧重静态分析，具有计算量小，操作简单的优点，但随着信息系统的复杂化和大规模化，宏观的漏洞风险随生命周期动态演化，例如在漏洞公布阶段，随着公众开始了解漏洞信息，潜在的攻击者数量不断增加，网络系统风险呈现动态上升趋势。

通过上述分析，目前研究在以下 3 个方面有待改进：1) 现有安全漏洞模型侧重于漏洞发现和数量预测，缺

乏对漏洞生命周期整体的考虑；2) 需结合漏洞利用历史信息，对各漏洞生命周期过程的漏洞利用概率进行分析，从而更加科学、真实地反映时间概率（时间维度上的概率值）的变化规律；3) 如何客观地衡量攻击威胁对网络系统的影响，动态评估漏洞生命周期过程中的时间风险（时间维度上的风险值），给出合理准确的风险评估方法。

基于上述分析，考虑当前安全漏洞风险度量的研究缺乏对漏洞生命周期整体的考虑，仅从统计学角度进行规律发现，缺乏细致科学的状态转移概率计算。而吸收 Markov 链适用于时间序列建模，被广泛应用于状态转移概率的计算，在经济学预测领域应用较多，且漏洞生命周期中各状态间转移在时间上具有无后效性，存在至少 1 个终止状态，切合吸收 Markov 链的特征，因此借助吸收 Markov 链可以有效提高解决上述问题的效率，更加真实、准确地反映现实世界安全中漏洞演化的基本规律。本文利用吸收 Markov 链描述漏洞状态的转移过程，通过在时间轴上进行推演，建立漏洞生命周期的时间随机模型；在此基础上结合国际权威漏洞披露组织 CVE 的漏洞数据库，利用先验的历史漏洞信息作为模型输入，计算漏洞生命周期中各状态的时间概率；然后通过模型训练分析漏洞利用规律，并对未来趋势进行预测，实例分析验证了模型及方法的实用性和有效性。

本文的主要贡献有：1）建立了一种基于吸收 Markov 链的安全漏洞生命周期模型，实现了漏洞生命周期中各状态演化的定量刻画，为总结漏洞状态的演化规律提供了理论基础；2）提出了一种漏洞利用及修复的期望概率计算方法，为安全漏洞的主动防御提供了定量数据参考；3）提出了一种安全漏洞时间维度风险度量方法，一定程度上解决了安全漏洞风险状况动态测度难的问题。

6 预备知识

为方便描述，文中所用符号及部分表达式含义如表 1。

Table 1 Primary symbols and their descriptions

表 1 主要符号及其含义

符号	含义	符号	含义
“Creation”	漏洞产生阶段	“Discovery”	漏洞发现阶段
“Exploit”	漏洞利用阶段	“Disclosure”	漏洞公布阶段
“Patched”	漏洞修复阶段	P	状态转移概率矩阵
Q	过渡状态间的转移矩阵	R	过渡状态到吸收状态的转移矩阵
I	单位矩阵	n	所有状态的数量
$x_i \to x_j$	状态 x_i 转移到 x_j	$p_{i,j}$	$x_i \to x_j$ 的一步转移概率
State i	第 i 个状态	X	随机序列
Vendo	厂商名称	H_t	t 时刻的状态发生概率向量
$H_t(i)$	t 时刻漏洞处于 *State i* 的概率	*Vuln*	漏洞名称
r	吸收状态的数量	d	过渡状态的数量
ExpSco	CVSS 可利用性得分	*ImpSco*	CVSS 威胁影响得分
t	漏洞生命周期的 t 时刻（天）	$ExpPro(Vuln)$	漏洞 *Vuln* 的可利用性概率的期望值
$TimeRisk(t)$	网络系统在 t 时刻的风险值	$ImpSco(Vuln)$	漏洞 *Vuln* 的威胁影响得分
Level	漏洞攻击难度等级	$Weight_x$	第 x 个漏洞的权重
H_t	t 时刻的状态发生概率向量	$TimPro(Vuln, State\ i, t)$	t 时刻漏洞 *Vuln* 处在状态 *State i* 的概率

6.1 漏洞生命周期

漏洞是信息系统在硬件、软件、固件或者协议的需求、设计、实现、配置和运行过程中有意或无意产生的一个或若干个能够被利用的缺陷，它会导致信息系统处于风险之中[21]，受到广泛认可的漏洞生命周期的几个阶

段[9]及其风险状况如图 1 所示，相关定义如下。

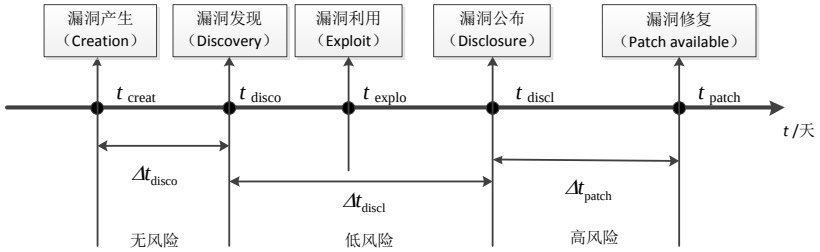

Fig.1　Stages of vulnerability lifecycle

图 1　漏洞生命周期各阶段

定义 1 漏洞产生 Creation. 指系统缺陷或者错误在设计过程中产生。例如软件开发过程中编程语言的 bug，从 Creation 到 Discovery 时间段内漏洞处于“Creation”状态。

定义 2 漏洞发现 Discovery. 指产品测试和运行过程中，漏洞被首次发现。发现者可能是软件开发商（如 Micorsoft, IBM, Sun 等）、漏洞研究机构（如 CERT/CC，NIST，Symantic 等）或者黑客组织（如 Anonymous，Lizard Squad，Lulzsec 等）。从 Discovery 到（Exploit/Disclosure/Patched）时间段内漏洞处于“Discovery”状态。

定义 3 漏洞利用 Exploit. 指漏洞首次被恶意攻击者利用。例如 0-day 漏洞表示即被发现即被利用，因此安全威胁程度高。从 Exploit 到 Patched 时间段内漏洞处于“Exploit”状态。

定义 4 漏洞公开 Disclosure. 指权威机构（如 CERT，CVE，CNVD 等）发布漏洞公告，披露漏洞的详细信息。从 Disclosure 到（Exploit/Patched）时间段内漏洞处于“Disclosure”状态。

定义 5 漏洞修复 Patched. 指软件开发商首次发布漏洞补丁程序。Patched 之后漏洞处于“Patched”状态。

与上述定义相关的几点说明：

1. 对于“Creation”状态，漏洞尚未被发现或者利用，因此该阶段安全漏洞无时间风险。

2. 对于“Discovery”状态，当漏洞刚被发现时，对漏洞的发掘和利用处于探索阶段，并且掌握漏洞信息的人员数量少，因此该阶段漏洞的时间风险低。

3. 对于“Exploit”状态，随着攻击脚本的传播，更多的攻击者掌握了漏洞的利用方法，从宏观上看，漏洞的时间风险不断增长。

4. 对于“Disclosure”状态，由于漏洞处于公开但未提供补丁的状态，随着公众开始了解漏洞信息，越来越多的潜在攻击者开始关注此安全漏洞，同时伴随着漏洞存在时间的增加，攻击者的攻击方式也会发生改变，例如攻击由简单的代码供给转变为自动攻击工具，使低技能的攻击者开始尝试漏洞利用，因此时间风险持续升高。

5. 对于“Patched”状态，随着补丁释放，安装用户增多，安全风险不断降低。由于不同漏洞的复杂度差别较大，对于另外一些未能提供补丁的漏洞，由于漏洞已逐步完成扩散，安全风险处于稳定水平。

6. 有些文献包含漏洞消亡状态[4]，事实上对应本文“Patched”状态，对于另外一些划分更多阶段的方法，通过阶段融合，都能转化为本文这 5 个阶段。

可以看出，整个漏洞生命周期过程中，漏洞风险是动态变化的，不但与补丁是否提供相关，并且和漏洞的状态和所处时间都密切相关，因此研究漏洞的时间风险具有重要意义。

6.2　通用漏洞评分标准

CVSS 由美国国家基础建设咨询委员会发布[22]，是漏洞评估的行业公开标准，其制定的如图 2 所示的漏洞可利用性评分标准全面衡量了漏洞的可利用性得分 *ExpSco* 和威胁影响得分 *ImpSco*，通过查询美国国家漏洞数据库(national vulnerability database, NVD)[24]可以得到具体值。

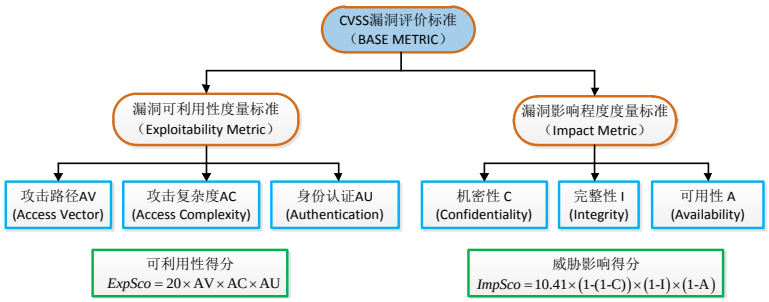

Fig.2　Framework of common vulnerability scoring system

图 2　CVSS 标准框架

由图 2 可知，漏洞 *Vuln* 的可利用性得分 $ExpSco(Vuln)=20\cdot AV\cdot AC\cdot AU$，$0<ExpSco(Vuln)\le 10$，通过查询 NVD 数据库可以计算 *ExpSco* 的具体值，得分越高表明漏洞利用越容易，状态转移成功概率越大。漏洞的利用难度等级用 Level 表示，其中 $Level\in\{High,Medium,Low\}$，依据 CVSS 给出的漏洞利用难度等级的划分标准，当 $ExpSco\in(0,4]$时，$Level=High$；当 $ExpSco\in(4,8]$这时，$Level=Medium$；当 $ExpSco\in(8,10]$时，$Level=Low$。例如，对于漏洞 CVE-2016-0135，通过查询 NVD 数据库可知 AV=0.395,AC=0.71,AU=0.704，因此 ExpSco=3.9，$Level=Low$。

国际权威漏洞披露组织 CVE 提供了漏洞字典表[23]，对漏洞名称进行统一标记（例如 CVE-2016-0728 代表了一种 Linux 内核漏洞），并按软件厂商，威胁得分对漏洞数量进行统计和分类。NVD 数据库公布了超过 7.5 万个已知漏洞的具体得分，通过查询 NVD 可以得到不同漏洞各项指标的具体分值。OSVDB 是一个开源的漏洞数据库[25]，提供漏洞生命周期的全部详细信息，包括时间、厂商和补丁信息。本文以 NVD，OSVDB 和 CVE 作为数据源，统计早期报道的先验漏洞信息，例如不同年份不同厂商发布的漏洞数量在生命周期不同时间段上的分布等，作为第 2 节模型输入，研究漏洞风险随时间的变化规律，以增强研究的通用性，详细方法见第 3 节。

6.3　吸收Markov链

Markov 模型被广泛应用于分析离散系统的状态转移规律，通过分析现时状态预测未来状态的变化趋势，是一种典型基于时间序列的分析方法。

定义 6 Markov 链[26]. 对于一个离散的包含有限数量状态的随机序列 $X=\{x_1,x_2,\cdots,x_n\}$，如果每个状态值仅与前一个相邻的状态值有关，而与再之前的状态序列无关，称为 Markov 链

$$p(x_{i+1}\mid x_i,x_{i-1},\cdots,x_1)=p(x_{i+1}\mid x_i).$$

定义 7 状态发生概率向量 *H*. 表示系统处于不同状态的发生概率，对于 $h_i\in H$，h_i 表示系统处于状态. x_i. 的概率。

定义 8 状态转移矩阵 *P*[26]. 将 Markov 链中的状态转移概率用邻接矩阵 *P* 表示，其中 $P_{i,j}$ 表示状态 $x_i\to x_j$ 的一步转移概率，若 $x_i\to x_j$ 不可达，令 $P_{i,j}=0$，矩阵 *P* 满足

$$\sum_{j=1}^{n}P_{i,j}=1，\text{其中 } 0\le P_{i,j}\le 1,1\le i,j\le n.$$

定义 9 吸收状态[26]. 表示系统的终止状态，该状态节点只有流入边，没有流出边。

定义 10 过渡状态[26]. 除去吸收状态的其他状态统称为过渡状态。

定义 11 吸收 Markov 链[26]. 含有吸收状态的 Markov 链称为吸收 Markov 链，对于一个包含 *r* 个吸收状态和 *d* 个过渡状态的吸收链，设状态总数 $n=d+r$，其状态转移矩阵的标准形式为

$$P=\begin{bmatrix}Q & R\\ 0 & I\end{bmatrix}.$$

其中，*P* 是 $n\times n$ 的矩阵，表示吸收 Markov 链中全部状态间的转移概率矩阵；*Q* 是 $d\times d$ 的矩阵，表示过渡状态

间的转移概率；R 是 $d\times r$ 的非零矩阵，表示过渡状态到吸收状态的转移概率；0 是 $r\times d$ 的零矩阵；I 是 $r\times r$ 的单位矩阵。

7 基于吸收 Markov 链的漏洞生命周期时间模型

由定义 11 可知，对漏洞生命周期进行 Markov 建模首先需要满足如下 2 个前提条件：

一方面，漏洞生命周期中各状态间转移具有无后效性，即漏洞在某一时刻的状态一旦确定，则此后状态演变不再受此前状态的影响，如 1.1 节所述。例如当漏洞处于状态“Disclosure”时，接下来往“Exploit”或者“Patched”演化仅与当前状态相关，而与“Discovery”无关；同时，文献[5]对超过 2000 个漏洞的统计数据表明，各状态间转移在时间维上拟合指数关系，因此漏洞生命周期的状态转移符合离散系统的 Markov 性质。

另一方面，由于漏洞演化始终存在至少 1 个终止状态，对应“Exploit”或者“Patched”，与吸收 Markov 链的吸收态相符。Beattie 等人[27]分析了 136 条 CVE 记录，发现 92 个补丁能够获得，20 个补丁有问题，24 个补丁未发布。考虑某些漏洞始终未能提供有效补丁（例如弱口令漏洞与用户设置习惯相关），本文模型包含“Exploit”和“Patched”2 种吸收态。

其次，利用吸收 Markov 链进行漏洞生命周期建模需要解决的 3 点关键技术如下：建立时间维度上的漏洞生命周期状态转移模型；计算 Markov 链模型中各状态间的转移概率；利用 Markov 链实现漏洞后续状态的推导预测。

基于上述分析，首先建立基于吸收 Markov 链的漏洞生命周期时间模型，绘制漏洞生命周期时间模型的状态转移图如图 3 所示，其中，State 1~State 3 对应“Creation”、“Discovery”和“Exploit”这 3 种过渡状态，State 4~State 5 对应“Exploit”和“Patched”这 2 种吸收状态，借助该模型推导生命周期内漏洞状态的演化规律。

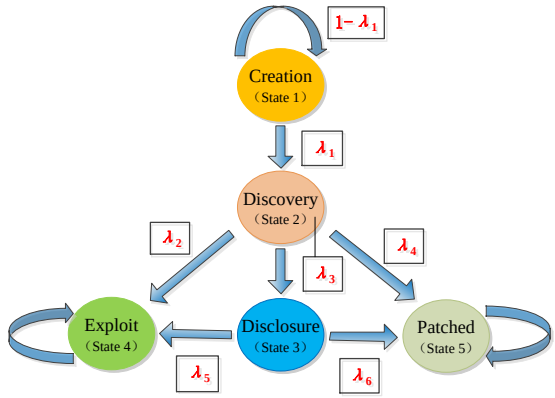

Fig.3 State transition graph of vulnerability life cycle based on absorbing Markov chain

图 3 基于吸收 Markov 链的漏洞生命周期状态转移图

结合定义 8 和定义 11 构造图 3 的状态转移概率矩阵 P、Q、R 如下，例如 $P_{1,2}$ 表示漏洞由 State 1 一步转移到 State 2 的概率 λ_1，其中 $\lambda_1,\lambda_2,\lambda_3,\lambda_4,\lambda_5,\lambda_6\in(0,1)$，满足 $\lambda_2+\lambda_3+\lambda_4=1$，$\lambda_5+\lambda_6=1$。

$$P=\begin{bmatrix}1-\lambda_1 & \lambda_1 & 0 & 0 & 0\\ 0 & 0 & \lambda_3 & \lambda_2 & \lambda_4\\ 0 & 0 & 0 & \lambda_5 & \lambda_6\\ 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 1\end{bmatrix},\quad Q=\begin{bmatrix}1-\lambda_1 & \lambda_1 & 0\\ 0 & 0 & \lambda_3\\ 0 & 0 & 0\end{bmatrix},\quad R=\begin{bmatrix}0 & 0\\ \lambda_2 & \lambda_4\\ \lambda_5 & \lambda_6\end{bmatrix}.$$

其次，按照漏洞利用难度、厂商两个维度对漏洞生命周期的状态转移概率进行定义，可以区分同一厂商的不同种漏洞的差别。设 CVE 数据库披露了软件供应商 *Vendo* 近 y 年发布的所有漏洞信息，第 k 年发现的等级为 *Level* 且处于状态“*State i*”漏洞数量 $Num(Vendo, Level, State\ i)_k$，相比前一年新增的该状态的漏洞数量为 $NumAdd(Vendo, Level, State\ i)_k$，第 k 年由状态“*State i*”一步转移到“*State j*”的漏洞数量 *Num*(*Vendo*, *Level*, *State*

$i \to State\ j)_k$，$k=1,2,\cdots,y$。状态间的一步转移概率计算方法如表 2 所示，对于第 2 行 λ_1，具体计算时将第 k 年发现的新增漏洞数量作为第 k-1 年“Creation”但未“Discovery”漏洞数量，由 $\lambda_2+\lambda_3+\lambda_4=1$可知，第 4 行成立，由 $\lambda_5+\lambda_6=1$ 可知，第 6 行成立。

Table 2 State transition probabilities information of vulnerability life cycle

表 2 漏洞生命周期的状态转移概率信息

编号	概率描述	概率计算
λ_1	软件供应商 *Vendo* 的难度为 *Level* 的漏洞由状态“Creation”转移到“Discovery”的概率	$\sum_{k=1}^{y}\left(\frac{NumAdd(Vendo,Level,Disco)_k}{Num(Vendo,Level,Disco)_{k-1}}\right)/y$
λ_2	软件供应商 *Vendo* 的难度为 *Level* 的漏洞由状态“Discovery”未经“Disclosure”直接转移到“Exploit”的概率	$\sum_{k=1}^{y}\left(\frac{Num(Vendo,Level,(Disco\to Explo))_k}{Num(Vendo,Level,Disco)_k}\right)/y$
λ_3	软件供应商 *Vendo* 的难度为 *Level* 的漏洞由状态“Discovery”转移到“Disclosure”的概率	$1-\lambda_2-\lambda_4$
λ_4	软件供应商 *Vendo* 的难度为 *Level* 的漏洞由状态“Discovery”未经“Disclosure”直接转移到“Patched”的概率	$\sum_{k=1}^{y}\left(\frac{Num(Vendo,Level,(Disco\to Patch))_k}{Num(Vendo,Level,Disco)_k}\right)/y$
λ_5	软件供应商 *Vendo* 的难度为 *Level* 的漏洞由状态“Disclosure”直接转移到“Exploit”的概率	$1-\lambda_6$
λ_6	软件供应商 *Vendo* 的难度为 *Level* 的漏洞由状态“Disclosure”直接转移到“Patched”的概率	$\sum_{k=1}^{y}\left(\frac{Num(Vendo,Level,(Discl\to Patch))_k}{Num(Vendo,Level,Discl)_k}\right)/y$

8 面向漏洞生命周期时间模型的安全度量

基于第 2 节建立的漏洞生命周期时间模型，本节借助该模型，推导漏洞状态转移满足的若干引理和定理，实现漏洞后续状态的量化预测。在此基础上给出 3 种评估方法，用于分析漏洞生命周期各状态的时间概率规律；计算漏洞最终停留在“Patched”和“Exploit”状态的期望概率值；结合漏洞“Exploit”状态的时间概率，分析漏洞的真实利用率，通过融合网络系统的所有安全漏洞，量化系统安全的时间风险，为准确、科学评估网络安全状态提供依据。

8.1 漏洞生命周期各状态的时间维度发生概率度量方法

以时间 t（单位为天）为度量标准，由于初始时刻漏洞处于“Creation”状态，状态发生概率向量 $H_0=[1\ \ 0\ \ 0\ \ 0\ \ 0]$，记 H_t 表示 t 天后的漏洞状态发生概率向量，$H_t(i)$ 表示经过 t 天后漏洞处于状态 i 的概率。当 $t=1$ 时，$H_1=H_0\times P$；当 $t=2$ 时，$H_2=H_1\times P=H_0\times P^{(2)}$；当 $t=3$ 时，$H_3=H_0\times P^{(3)}$；因此 t 天的漏洞状态发生概率向量 $H_t=H_0\times P^{(t)}$，通过计算 $P^{(t)}$ 可以得到 H_t。

定理 1 若 P 是满足定义 11 的吸收 Markov 链的状态转移矩阵，大小为 5×5，$P_{i,j}$ 表示漏洞由状态 i 转移到 j 的概率，令 $P_{i,j}^{(t)}$ 表示经过 t 天后，漏洞由状态 i 转移到状态 j 的概率，则

$$P^{(t)}=\left[P_{i,j}^{(t)}\right]=\begin{bmatrix}Q^{(t)} & \sum_{k=0}^{t-1}Q^{(k)}\times R\\ 0 & I\end{bmatrix}.$$

证明：利用数学归纳法进行证明：

1) 当 $t=2$ 时，由下式可知结论成立

$$P^{(2)}=\begin{bmatrix}Q & R\\ 0 & I\end{bmatrix}\cdot\begin{bmatrix}Q & R\\ 0 & I\end{bmatrix}=\begin{bmatrix}Q\times Q & Q\times R+R\times I\\ 0 & I\cdot I\end{bmatrix}=\begin{bmatrix}Q^{(2)} & (Q+Q^{(0)})\times R\\ 0 & I\end{bmatrix}=\begin{bmatrix}Q^{(2)} & \sum_{k=0}^{1}Q^{(k)}\times R\\ 0 & I\end{bmatrix}.$$

2) 假设当$t=t-1$时结论成立，即 $P^{(t-1)}=\begin{bmatrix} Q^{(t-1)} & \sum_{k=0}^{t-2}Q^{(k)}\times R \\ 0 & I \end{bmatrix}$，则满足

$$P^{(t)}=P^{(t-1)}\cdot P=\begin{bmatrix} Q^{(t-1)} & \sum_{k=0}^{t-2}Q^{(k)}\times R \\ 0 & I \end{bmatrix}\cdot\begin{bmatrix} Q & R \\ 0 & I \end{bmatrix}=\begin{bmatrix} Q^{(t-1)}\times Q & Q^{(t-1)}\cdot R+\sum_{k=0}^{t-2}Q^{(k)}\times R \\ 0 & I^2 \end{bmatrix}=\begin{bmatrix} Q^{(t)} & \sum_{k=0}^{t-1}Q^{(k)}\times R \\ 0 & I \end{bmatrix}$$

因此假设成立，由 1) 2)可知定理 1 成立。

结合表 2 和定理 1，设计漏洞生命周期各状态的时间概率计算方法，具体流程如下。

方法 1 漏洞生命周期各状态的时间概率度量方法

输入　漏洞 *Vuln* 的软件供应商 *Vendo*，CVE 数据库中近 *y* 年关于 *Vendo* 的漏洞数据统计量

输出　*t* 时刻漏洞 *Vuln* 处于状态 *State i* 的时间概率 *TimPro*(*Vuln, State i, t*)

步骤 1 依据图 3 构造吸收 Markov 链漏洞生命周期状态转移图；

步骤 2 查询 CVE 数据库近 *y* 年供应商 *Vendo* 的漏洞处于生命周期各状态的数量；

步骤 3 结合表 2 计算 $\lambda_1,\lambda_2,\lambda_3,\lambda_4,\lambda_5,\lambda_6$；

步骤 4 利用定义 11 和漏洞生命周期状态转移图，构造 5×5 的状态转移概率矩阵 P，3×3 的矩阵 Q，和 3×2 的矩阵 R；

步骤 5 计算 *Vuln* 在 *t* 时刻处于状态“Creation”的概率 $H_t(1)=Q_{1,1}^{(t)}$，“Discovery”的概率 $H_t(2)=Q_{1,2}^{(t)}$，“Disclosure”的概率 $H_t(3)=Q_{1,3}^{(t)}$，“Exploit”的概率 $H_t(4)=(\sum_{k=0}^{t-1}Q^{(k)}\times R)_{1,\cdot}$，“Patched”的概率 $H_t(5)=(\sum_{k=0}^{t-1}Q^{(k)}\times R)_{1,\cdot}$；

步骤 6 输出 $H_t(1),H_t(2),H_t(3),H_t(4),H_t(5)$，算法结束。

8.2 漏洞利用及修复的期望概率值度量方法

考虑“Exploit”状态的时间概率影响攻击者对漏洞的真实利用率，进而对网络资产的安全风险产生影响，而 Exploit”和“Patched”状态密切相关，本节对“Exploit”和“Patched”状态的时间概率进行深入分析，预测漏洞演化稳定后，该状态的概率期望值，辅助安全分析和控制。

引理 1 设漏洞当前处于过渡态 *i*，在吸收前停留在过渡态 *j* 的天数期望值用 $N_{i,j}$ 表示，其中 $1\le i,j\le3$，设 N 是大小为 3×3 的矩阵，将 $N_{i,j}$ 赋值给矩阵 N 中位置 (i,j) 上的元素，则 $N=(1-Q)^{-1}$。

证明：1) 漏洞从状态 *i* 开始演化，在吸收前分别经 $t(t=0,1,2,\cdots)$ 天，转移到状态 *j* 的天数之和. $N=\left[N_{i,j}\right]=\left[\sum_{t=0}^{\infty}\left((1-Q_{i,j}^{t})\times0+Q_{i,j}^{t}\times1\right)\right]=\left[\sum_{t=0}^{\infty}Q_{i,j}^{t}\right]=\sum_{t=0}^{\infty}\left[Q_{i,j}\right]^{(t)}=\sum_{t=0}^{\infty}Q^{(t)}$..

2) 由于 $\sum_{k=0}^{t-1}Q^{(k)}=I+Q+\cdots+Q^{(t-1)}=(I-Q)^{-1}\times(I-Q^{t})$，且演化稳定后过渡态间的转移概率为 0，满足 $\lim_{t\to\infty}Q^{(t)}=0$，因此 $\lim_{t\to\infty}(I-Q^{(t)})=I$，故 $\sum_{t=0}^{\infty}Q^{(t)}=(I-Q)^{-1}$。

综合 1) 2)可知，引理 1 成立。

定理 2 漏洞由过渡态 *i* 演化到吸收态 *j* 的概率期望值为 $B_{i,j}$，$1\le i\le3$，$4\le j\le5$。设 B 是大小为 3×2 的矩阵，将 $B_{i,j}$ 赋值给矩阵 B 中位置 $(i,j-3)$ 元素，称 B 为期望概率矩阵，则 $B=N\times R$。$B_{1,1}$ 表示漏洞处于“Exploit”的期望概率，$B_{1,2}$ 表示漏洞处于“Patched”的期望概率。

证明：计算漏洞从状态 *i* 演化到状态 *j* 的概率，即求解分别经 $t(t=0,1,2,\cdots)$ 天，由不同过渡态 $l=1,2,3$，到达吸收态 *j* 的概率之和，故

$$B=\left[B_{i,j}\right]=\left[\sum_{t=0}^{\infty}\left(Q_{i,1}^{t}\times R_{1,j}+Q_{i,2}^{t}\times R_{2,j}+Q_{i,3}^{t}\times R_{t,j}\right)\right]=\sum_{t=0}^{\infty}Q^{(t)}\times R\,.$$

由引理 1 可知 $\sum_{t=0}^{\infty} Q^{(t)} = (1-Q)^{-1} = N$，故 $B = N \times R$，定理 2 成立。

结合表 2 和定理 2，计算漏洞 *Vuln* 演化到“Exploit”和“Patched”状态的期望概率值 $ExpPro(Vuln)$ 和 $PatPro(Vuln)$，具体流程如下。

方法 2 漏洞利用及修复概率的期望值计算方法

输入　漏洞 *Vuln* 的软件供应商 *Vendo*，CVE 数据库中近 *y* 年关于 *Vendo* 的漏洞数据统计量

输出　漏洞 *Vuln* 被利用和修复的期望概率 *ExpPro*(*Vuln*)和 $PatPro(Vuln)$

步骤 1 依据图 3 构造吸收 Markov 链漏洞生命周期状态转移图；

步骤 2 查询 CVE 数据库得到近 *y* 年供应商 *Vendo* 的漏洞处于生命周期各状态的数量；

步骤 3 结合表 2 计算 $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6$；

步骤 4 利用定义 11 和漏洞生命周期状态转移图，构造 5×5 的状态转移概率矩阵 *P*，3×3 的矩阵 *Q*，和 3×2 的矩阵 *R*；

步骤 5 计算 3×3 期望概率矩阵 $B = (1-Q)^{-1} \times R$；

步骤 6 输出 $ExpPro(Vuln) = B_{(1,1)}$，$PatPro(Vuln) = B_{(1,2)}$，算法结束。

8.3 安全漏洞时间维度风险度量方法

为分析安全漏洞的风险随时间的变化规律，借鉴 CVSS 标准计算漏洞的威胁影响得分和可利用性得分，并结合漏洞“Exploit”状态的时间概率分析漏洞的真实利用率，在此基础上依据网络系统的漏洞权重，度量网络风险。CVSS 给出了一种基于机密性、完整性、可用性 3 个评价指标的漏洞威胁得分标准，用于衡量单个漏洞对网络的影响，如 1.2 节图 2 所示，其威胁得分为

$$ImpSco(Vuln) = 10.41 \times \left(1 - (1-C) \times (1-I) \times (1-A)\right).$$

其中 *C*、*I*、*A* 分别是机密性、完整性、可用性的威胁影响得分，通过查询 NVD 数据库可以得到具体值。

对于一个实际网络系统，安全风险和漏洞数量、漏洞威胁、漏洞的重要性权重以及漏洞的真实利用率相关。设 *q* 表示网络中漏洞数量，第 *x* 个漏洞 $Vuln_x$ 的重要性权重为 $Weight_x$，CVSS 威胁影响得分为 $ImpSco(Vuln_x)$，CVSS 可利用性得分 $ExpSco(Vuln_x)$，$Vuln_x$ 的产生时刻 $t_{creat} = t_x$，其中 $1 \le x \le q$，则系统安全风险的计算步骤如下。

方法 3 网络系统的安全漏洞时间风险度量方法

输入　漏洞 $Vuln_x$ 的软件供应商 $Vendo_x$，CVE 数据库中近 *y* 年关于 $Vendo_x$ 的漏洞数据统计量，漏洞产生时间 t_x，漏洞权重 $Weight_x$，NVD 数据库提供的 $Vuln_x$ 的 CVSS 得分，漏洞数量 *q*

输出　*t* 时刻的风险值 *TimeRisk*(*t*)

步骤 1 依据图 3 构造吸收 Markov 链漏洞生命周期状态转移图；

步骤 2 查询 CVE 数据库得到近 *y* 年供应商 *Vendo* 的漏洞处于生命周期各状态的数量；

步骤 3 结合表 2 计算 $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6$；

步骤 4 利用定义 11 和漏洞生命周期状态转移图，构造 5×5 的状态转移概率矩阵 *P*，3×3 的矩阵 *Q*，和 3×2 的矩阵 *R*；

步骤 5 计算 *t* 天后 *Vuln* 处于状态“Exploit”的发生概率 $H_t(4) = (\sum_{k=0}^{t-1} Q^{(k)} \times R)_{1,1}$；

步骤 6 查询 NVD 数据库得到漏洞 *Vuln* 的得分信息，依据 1.2 节图 2 计算威胁影响得分 $ImpSco(Vuln_x)$ 和可

利用性得分 $ExpSco(Vuln_x)$；

步骤 7 计算单漏洞 $Vuln$ 在 t 时刻的时间风险，公式为：

$$TimeRisk(Vuln,t)=(\sum_{k=0}^{t-1}Q^{(k)}\times R)_{1,1}\times ExpSco(Vuln_x)\times ImpSco(Vuln_x)$$；

步骤 8 计算网络系统在 t 时刻的时间风险值：

$$TimeRisk(System,t)=\sum_{x=1}^{q}\left((\sum_{k=0}^{t-t_x-1}Q^{(k)}\times R)_{1,1}\times ExpSco(Vuln_x)\times ImpSco(Vuln_x)\times Weight_x\right)$$；

步骤 9 输出 $TimeRisk(Vuln,t)$ 、 $TimeRisk(System,t)$，算法结束。

网络安全风险求解一般是计算攻击发生可能性和严重程度的乘积，但本质上是一种静态的分析方法。本文借鉴漏洞可利用性得分衡量漏洞利用难易程度，利用漏洞影响得分衡量漏洞威胁程度，并结合“Exploit”状态的时间概率，分析随时间变化的漏洞难易程度，通过融合以上参数，实现了安全漏洞风险状况的动态量化测度。

9 实例与分析

APT 攻击已成为当前网络系统遭受最严重的攻击之一，具有隐藏性高，持续时间长，威胁性高的特点，其攻击过程具有重要的分析意义。本节以典型 APT 攻击场景中“WannaCry”勒索病毒的漏洞利用过程为例，通过对漏洞生命周期各状态发生概率的演化规律的总结和分析。验证了模型及方法的合理性和有效性。

9.1 “WannaCry”勒索攻击基本信息

勒索病毒攻击作为一种新型 APT 攻击，其目标是 Windows 操作系统，该病毒在看似正常的电子邮件附件中伪装，一旦收件人打开附件，就会在本地磁盘上加密文件和映射网络磁盘。最近，一种新的称为“WannaCry”的蠕虫式勒索病毒在全球范围内肆意传播，由黑客组织利用美国国家安全局 NSA 泄露的危险漏洞“EternalBlue”（永恒之蓝）进行传播，在 2017 年 5 月 12 日全球大爆发，至少 150 个国家、30 万名用户遭受攻击，造成巨大的经济损失。本节结合 APT 攻击场景，以勒索病毒“WannaCry”为例，分析其攻击过程，动态、真实地预测漏洞利用时间概率和时间风险，对于主动防御和风险控制具有重要意义。

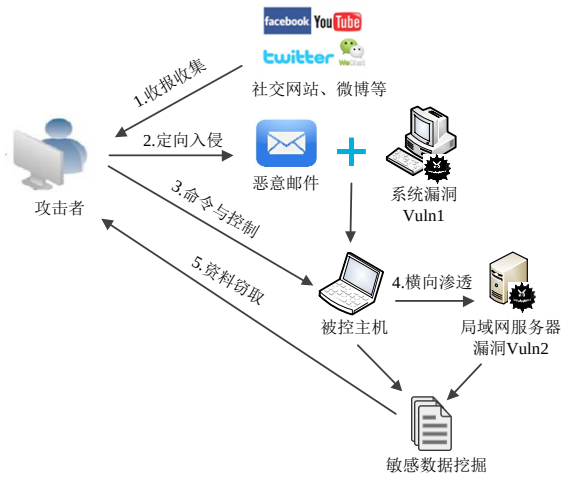

Fig.4 Attacking process of “WannaCry”

图 4 “WannaCry”勒索攻击过程

“WannaCry”勒索攻击过程如图 4，通过查询 NVD 数据库可以得到攻击利用漏洞的基础信息如表 3 所示，病毒传播过程主要利用到 Windows 的 2 个漏洞，攻击者首先锁定企业、机构、政府部门的资源，通过互联网搜

集、主动扫描被攻击目标的情报，利用漏洞 Vuln1（CVE-2017-0199）发送钓鱼邮件，将病毒添加在 Office 附件里，PC 一旦打开附件，勒索病毒释放成可执行文件，第一个传播的源头被感染成功；第二步是借助“永恒之蓝”漏洞攻击工具，利用漏洞 Vuln2（CVE-2017-7494）攻击局域网内所有开放了 TCP445 端口的 Windows 系统主机，完成横向渗透，并提升权限实现文件信息共享。

Table 3 Vulnerability exploits relevant information of “WannaCry”

表 3 “WannaCry”勒索攻击利用漏洞的基础信息

漏洞编号	目标主机信息	漏洞名称	漏洞描述	漏洞时间	漏洞年龄	*ExpSco*	*ImpSco*	*Level*	*Weight*
Vuln1	Microsoft Office 2010 SP2	CVE-2017-0199	执行任意代码漏洞	04/12/2017	33	1.8	5.9	Low	0.5
Vuln2	Windows 8.1	CVE-2017-7494	远程权限提升漏洞	04/05/2017	65	3.9	5.9	Low	0.5

9.2 漏洞生命周期各状态的时间维度发生概率规律总结

为了简化讨论，先不考虑软件厂商的区别，仅以一般意义上的单个漏洞为例，分析漏洞生命周期各状态的时间概率。对于图 4 中漏洞 *Vuln*1，由 3.1 节分析可知，$\lambda_1,\lambda_2,\lambda_3,\lambda_4,\lambda_5,\lambda_6$ 的大小直接影响到各状态的时间概率，下面通过一般化的参数枚举，总结时间概率规律。

1) 设 $\lambda_1=0.1,\lambda_2=0.3,\lambda_3=0.3,\lambda_4=0.4,\lambda_5=0.4,\lambda_6=0.6$，得到状态转移概率矩阵如下，利用方法 1 计算不同时刻的状态发生概率如表 4，依据表 4 绘制各状态在时间维上的概率分布如图 5 所示。

$$P=\begin{bmatrix}0.9 & 0.1 & 0 & 0 & 0\\ 0 & 0 & 0.3 & 0.3 & 0.4\\ 0 & 0 & 0 & 0.4 & 0.6\\ 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 1\end{bmatrix},\quad Q=\begin{bmatrix}0.9 & 0.1 & 0\\ 0 & 0 & 0.3\\ 0 & 0 & 0\end{bmatrix},\quad R=\begin{bmatrix}0 & 0\\ 0.3 & 0.4\\ 0.4 & 0.6\end{bmatrix}$$

Table 4 Time probability of each state in vulnerability life cycle

表 4 漏洞生命周期各状态的时间概率

t / 天	“Creation”	“Discovery”	“Disclosure”	“Exploit”	“Patched”
1	1.0000	0.0000	0.0000	0.0000	0.0000
2	0.9000	0.1000	0.0000	0.0000	0.0000
3	0.8100	0.0900	0.0300	0.0300	0.0400
4	0.7290	0.0810	0.0270	0.0690	0.0940
5	0.5905	0.0656	0.0219	0.1357	0.1863
6	0.5314	0.0590	0.0197	0.1641	0.2257
7	0.4783	0.0531	0.0177	0.1897	0.2611
8	0.4305	0.0478	0.0159	0.2127	0.2930
9	0.3874	0.0430	0.0143	0.2335	0.3217
10	0.3487	0.0387	0.0129	0.2521	0.3475
15	0.2059	0.0229	0.0076	0.3209	0.4427
20	0.1216	0.0135	0.0045	0.3615	0.4989
40	0.0148	0.0016	0.0005	0.4129	0.5701

80	0.0002	0.0000	0.0000	0.4199	0.5799
93	0.0001	0.0000	0.0000	0.4200	0.5799
94	0.0000	0.0000	0.0000	0.4200	0.5800
…	…	…	…	…	…

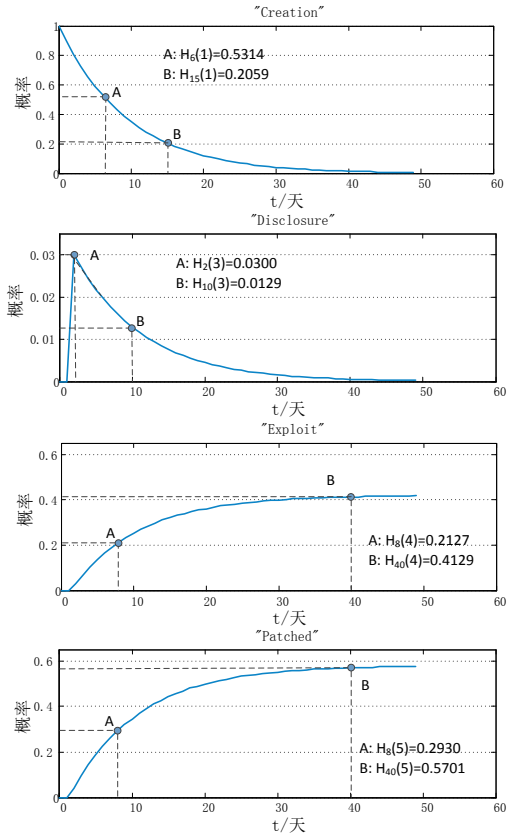

Fig.5 Time probability of each state in vulnerability life cycle

图 5 漏洞生命周期各状态的时间概率

1. 对于“Creation”状态，在漏洞产生后的第 1 天时，其发生概率为 1，随着时间的推进，漏洞状态向“Discovery”逐渐演化，因此“Creation”状态的概率不断降低，在前 20 天内，曲线的斜率较陡，说明概率下降的速度快，到第 6 天时，概率值已经降低为 0.5314，到第 15 天时，概率值降为 0.2059，说明漏洞维持在该状态的可能性较低，会迅速向后续状态演化。

2. 漏洞初始处于“Disclosure”状态的概率为 0，到第 3 天时，概率值陡增至 0.03，说明在漏洞产生之后，会迅速进入待公开状态，之后随着时间推进，漏洞开始由“Disclosure”状态向“Exploit”和“Patched”状态的过渡，此时“Disclosure”状态的概率值不断降低。同时，当漏洞到达稳定的“Exploit”或者“Patched”状态时，相对应时刻的“Disclosure”概率值为 0。

3. 对于“Exploit”和“Patched”状态，初始概率值为 0，随着时间推进，漏洞逐步被发现和披露，漏洞利用不断扩散，“Exploit”状态的发生概率不断增大，由于软件厂商努力研发修复补丁，加紧开发进度，因此漏洞演化到“Patched”的概率也不断增加。可以看出，“Exploit”概率值增速要小于“Patched”的增速，例如第 8 天时，被利用的概率为 0.2127，补丁发布的概率为 0.2930，说明此时漏洞被修复的概率要始终高于被利用的概率，通过调节参数可以改变漏洞的时间概率，为安全漏洞防御提供指导。

4. 由表 4 可以看出，在任意时间漏洞处于不同状态的概率之和为 1，同时随着时间推进，最终漏洞演化到稳定的吸收态，即“Exploit”的概率为 0.42，“Patched”的概率为 0.58，概率之和为 1。

2) 通过调节参数 $\lambda_1 \sim \lambda_6$ 可以分析参数变化对各漏洞状态时间概率的影响规律，由于涉及变量多，为简化讨论，考虑其他参数值不变，仅以调节 λ_1 为例说明。利用 3.1 节方法 1 计算 $\lambda_1 = 0.1$，0.3，0.5，0.7，0.9 这 5 种情况下的状态时间概率，并绘制概率值在时间维的分布如图 6 所示。

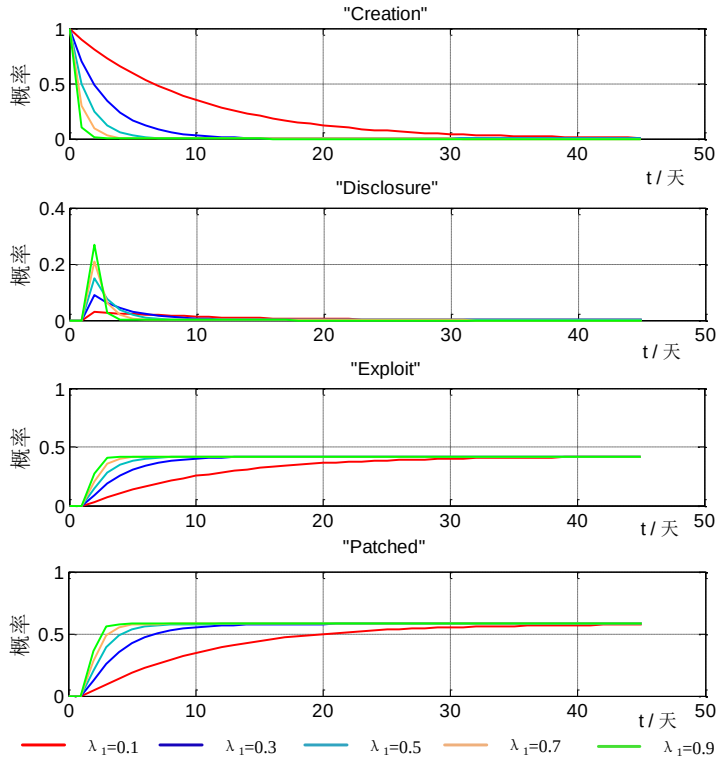

Fig.6 Time probability of each state in vulnerability life cycle with different λ_1

图 6 不同 λ_1 下的漏洞生命周期状态时间概率

从图 6 可以看出，参数 λ_1 取值对各状态发生概率在时间维的走势影响不大；对于所有状态，当 λ_1 的取值更大时，其发生概率的变化更迅速，能更快趋向于稳定状态，上述发现对于研究漏洞生命周期的时间长度具有重要意义，例如当新增漏洞的产生速度加快时，宏观上可以判定多数漏洞的生命周期在缩短，漏洞在较短时间内能演化到稳定状态，意味着攻击者利用漏洞的周期也在缩短，对于指导软件厂家和安全漏洞研究机构加快推进补丁研发速度，缩短研发周期具有参考价值。

9.3 漏洞利用及修复的期望概率值计算

接着 4.1 节，首先考虑一般意义上的单个漏洞，由 3.2 节方法 2 可以计算出期望概率矩阵 $B = \begin{bmatrix} \lambda_2 + \lambda_3 \times \lambda_5 & \lambda_4 + \lambda_3 \times \lambda_6 \\ \lambda_2 + \lambda_3 \times \lambda_5 & \lambda_4 + \lambda_3 \times \lambda_6 \\ \lambda_5 & \lambda_6 \end{bmatrix}$，则漏洞被利用的期望概率值是 $\lambda_2 + \lambda_3 \cdot \lambda_5$，被修复的期望概率值是 $\lambda_4 + \lambda_3 \cdot \lambda_6$。当 $\lambda_1 = 0.1, \lambda_2 = 0.3, \lambda_3 = 0.3, \lambda_4 = 0.4, \lambda_5 = 0.4, \lambda_6 = 0.6$ 时，漏洞被利用的期望概率 0.42，被修复的期望概率 0.58，与 4.1 节表 4 得出的稳定状态（$t = 93$ 天）概率结果一致，与预期相符，同时说明此时漏洞生命周期为 93 天。

定义漏洞被利用和修复的期望概率差值 $f = \left|(\lambda_2 - \lambda_4) + \lambda_3 \times (\lambda_5 - \lambda_6)\right|$，显然当 $\lambda_2 - \lambda_4$ 和 $\lambda_5 - \lambda_6$ 不变时，λ_3 越大，漏洞被利用和修复的概率之差越大，表明提高已“发现”漏洞被“公开”的概率，能同时提高漏洞被利用或修复的可能性。对于实际应用的指导意义是：当软件厂商的补丁研发水平提高时，应提高对发现漏洞的公开

概率，此举能从整体上提高漏洞被修复的概率，当软件研发能力受限时，应降低对发现漏洞的公开概率，避免利用方法扩散，导致漏洞被更容易演化到“Exploit”状态。

9.4 安全漏洞的时间维度风险分析

结合图 4，本节分析遭受“WannaCry”入侵的系统在时间维度上的风险变化趋势，由表 3 可知，“WannaCry”攻击利用漏洞均来自厂商 Microsoft，查询 CVE 数据库可知，截至 09/10/2017，供应商 Microsoft 的发布的漏洞总数 $Num(\text{Microsoft}, Disco)_1 = 4236$，近 1 年来（09/10/2016~09/10/2017）发布的不同 Level 漏洞的历史统计数据如表 5；然后利用第 2 节表 2 计算不同 Level 漏洞在其生命周期中的状态转移概率，如表 6 所示，可以看出，不同“Level”漏洞的生命周期时间模型中的状态转移概率不同；接着利用 3.1 节方法 1 计算厂商 Microsoft 不同“Level“漏洞处于生命周期“Exploit”状态的时间概率，记录在表 7 中，并绘制概率值在时间维度上的分布如图 7 所示。

从图 7 和表 7 可以看出，对于利用难度 High 的漏洞，稳定后到达“Exploit”状态的期望概率为 0.1287，漏洞生命周期为 32 天，对应图 7 中点 C；难度 Medium 的漏洞，“Exploit”状态的期望概率为 0.1536，漏洞生命周期为 26 天，对应图 7 中点 B；难度 Low 的漏洞，“Exploit”状态的期望概率为 0.1843，生命周期为 22 天，在图中用 A 标出。

Table 5 CVE History statistics information of Microsoft vulnerabilities

表 5 Microsoft 漏洞的 CVE 历史数据统计信息

Statistic number of vulnerability	Level = Low	Level = Medium	Level = High
$Num(Microsoft, Level, Disco)_1$	1742	1342	1152
$Num(Microsoft, Level, Discl)_1$	1537	1037	737
$NumAdd(Microsoft, Level, Disco)_1$	479	385	398
$Num\left(Microsoft, Level, (Disco \to Explo)\right)_1$	28	122	170
$Num\left(Microsoft, Level, (Disco \to Patch)\right)_1$	1390	1081	913
$Num\left(Microsoft, Level, (Discl \to Patch)\right)_1$	606	409	285

Table 6 Calculations of transition probabilities for different vulnerability levels of Microsoft

表 6 Microsoft 不同“Level”漏洞在其生命周期内的状态转移概率计算结果

Vulnerability level	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6
Low	0.2750	0.0161	0.1860	0.7979	0.6057	0.3943
Medium	0.2869	0.0909	0.1036	0.8055	0.6056	0.3944
High	0.3455	0.1476	0.0599	0.7925	0.6133	0.3867

Table 7 The probabilities of “Exploit” for different vulnerability levels of Microsoft in their life cycle

表 7 Microsoft 不同“Level“漏洞在生命周期内的“Exploit”概率

Day/Level	Low	Medium	High	Day/Level	Low	Medium	High
1	0	0	0	2	0.0044	0	0.0044
4	0.0971	0.0627	0.0634	6	0.1470	0.1074	0.0944
8	0.1683	0.1301	0.1107	10	0.1775	0.1417	0.1193
12	0.1814	0.1476	0.1238	14	0.1831	0.1505	0.1261
16	0.1838	0.1521	0.1274	18	0.1841	0.1528	0.1280
20	0.1842	0.1532	0.1284	**22**	**0.1843**	0.1534	0.1286

24	0.1843	0.1535	0.1287	**26**	0.1843	**0.1536**	0.1287
28	0.1843	0.1536	0.1287	30	0.1843	0.1536	0.1287
32	0.1843	0.1536	**0.1288**	34	0.1843	0.1536	0.1288
36	0.1843	0.1536	0.1288	38	0.1843	0.1536	0.1288
…	…	…	…	…	…	…	…

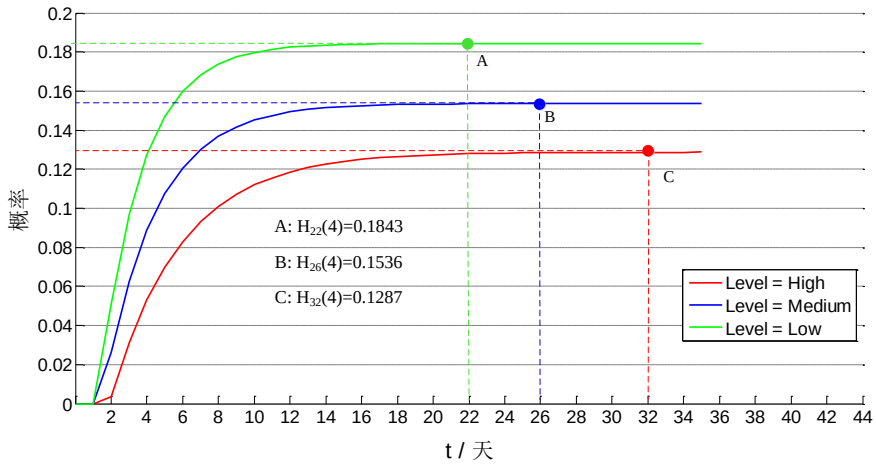

Fig.7　The probability distributions of “Exploit” for different vulnerability levels of Microsoft in their life cycle

图 7 Microsoft 不同“Level”漏洞在生命周期内的“Exploit”概率分布

上述结果从宏观角度预测了未来一段时间内，Microsoft 产品不同 Level 漏洞被利用的平均概率随时间的变化规律。不难看出在同一时刻，难度为 Low 的漏洞被利用的概率最高，而难度为 High 的漏洞被利用的概率最低，但后者到达稳定状态所需时间更长。该结论与实际漏洞利用攻击相符，对于难度低的漏洞，随着攻击脚本的传播，低水平的攻击者也能成功发动攻击，导致漏洞被利用概率增大，并且随着攻击方法的迅速传播，漏洞状态能更快趋于稳定，因此生命周期更短。

通过上述分析，对于新披露的漏洞，通过查询相应厂商的历史漏洞信息，并结合 CVSS 评估漏洞利用难度 Level，利用本文方法可以预测漏洞生命周期长度以及时间维度的利用概率，预测结果从整体上为安全漏洞的风险控制提供了参考。

下面分析图 4 网络系统在漏洞生命周期内的风险变化情况，对于“WannaCry”攻击利用的 2 个漏洞，结合表 3 中的漏洞基础信息，以漏洞 CVE-2017-7494 的公开时间（04/05/2017）为起始时间，设定该时刻 $t=0$，然后利用 3.3 节方法 3，计算被入侵系统中的单个漏洞及系统整体风险值，记录在表 8 中，并绘制漏洞时间风险变化如图 8 所示。

Table 8　Time risk values of “WannaCry” blackmail attacks

表 8　“WannaCry”勒索攻击的时间风险值

t / 天	Vuln1	Vuln2	System	*t* / 天	Vuln1	Vuln2	System	*t* / 天	Vuln1	Vuln2	System
0	0	0	0	2	0	0.889	0.444	4	0	1.873	0.936
6	0	2.390	1.195	8	0.047	2.662	1.354	10	0.673	2.804	1.739
12	1.003	2.880	1.941	14	1.176	2.919	2.048	16	1.267	2.940	2.103
18	1.315	2.951	2.133	20	1.340	2.956	2.148	22	1.353	2.960	2.156

24	1.360	2.961	2.160	26	1.363	2.962	2.163	28	1.366	2.962	2.164
30	1.366	2.963	2.164	**32**	**1.367**	**2.963**	**2.165**	34	1.367	2.963	2.165
…	…	…	…	…	…	…	…	…	…	…	…

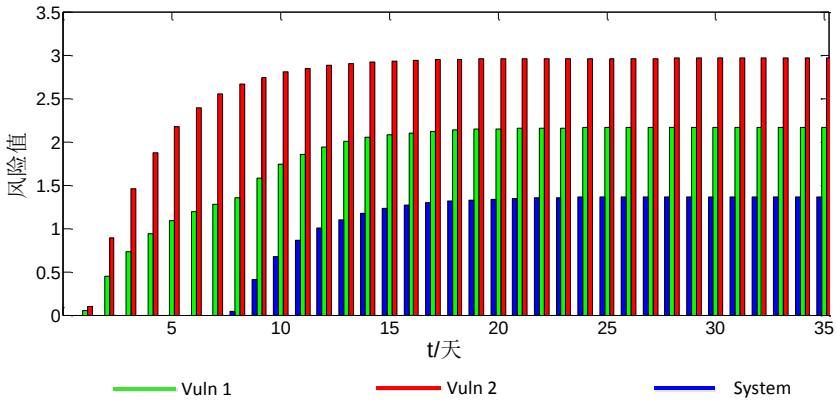

Fig.8 Security risk trend of "WannaCry" blackmail attacks

图 8 “WannaCry”勒索攻击的安全风险趋势

由图 8 可知，单个漏洞和整体系统的风险值在时间维度上均符合指数分布。在初始阶段，“Vuln1”会产生安全风险，而“Vuln2”不存在风险隐患，“System”风险处于较低水平。随着“Vuln2”的公开，“WannaCry”蠕虫开始利用此漏洞搜寻入侵目标主机，并结合“Vuln1”实现在局域网内横向渗透，随后几天内，受感染主机数量急剧增多，因此“System”风险迅速上升，由于“Vuln2”的 *ExpSco* 高于“Vuln1”，因此前者的单漏洞风险值更高。结合表 8 可知，在 $t=32$ 附近“System”风险达到一个较高的稳定值，表现为勒索病毒在全球范围大爆发，与著名软件供应商 Symante[28]提供的勒索病毒集中爆发时间为 12/05/2017 较为接近，验证了本文方法的有效性和准确性。

9.5 方案综合比较

Table 9 Comprehensive comparisons among our method and others

表 9 本文方法与其他方法综合比较

类型	客观程度	未知漏洞	动态度量	风险值量化	时间维分析
文献[11]	低	否	否	否	否
文献[12]	低	否	否	是	否
文献[13]	中	否	否	否	否
文献[14]	高	是	否	是	否
文献[15]	低	否	否	否	否
本文方法	中	是	是	是	是

本文与典型相关研究的特点比较见表 9，从表中可以看出，文献[11]通过对漏洞威胁和利用可能性划分等级，结合风险矩阵计算系统的安全风险等级，但等级划分具有较强的主观性；文献[12]结合漏洞属性评估漏洞影响度，该过程依赖领域知识，分析未知漏洞的能力不足；文献[15]基于层次分析法，但系统安全指标体系难以建立，且未对风险结果进行量化；文献[13]结合粗集理论实现了漏洞属性约简，但只能分析已知漏洞风险；文献[14]结合先验的漏洞数据库，通过 CVSS 数据集训练贝叶斯信念网络，一种程度上能度量未知漏洞，且结果的客观性较高；但以上研究均未结合漏洞生命周期过程，缺乏时间维度上的动态度量。

相比之下，本文结合漏洞生命周期过程，从宏观角度分析了漏洞状态的一般演化规律，度量安全漏洞的风险水平，具备对未知漏洞风险预测的能力，给出了漏洞利用概率在时间维度上的变化函数，能够实时动态度量网络风险，更真实地反映系统安全漏洞的风险状况，为安全管理员的风险管理提供更科学的决策支持。

10 结束语

信息系统安全漏洞是系统设计本身的缺陷，它可能引起系统异常、瘫痪，甚至被黑客利用进行入侵，引起更大的损失，因此安全漏洞风险评估是信息系统安全研究的重要内容。目前关于安全漏洞风险研究大多的静态的分析方法，未考虑漏洞利用概率在漏洞产生时间维度上的分布规律，缺少对于漏洞生命周期整体的考虑。

本文从时间维度出发，利用吸收 Markov 链对漏洞生命周期中的状态迁移进行建模，以 CVE 的漏洞数据库为输入，依据历史先验漏洞信息，构造状态转移概率矩阵，通过矩阵推导，在时间维上对安全风险进行量化分析，真实、动态地呈现漏洞的时间风险。实例分析结果表明所提模型及方法符合实际应用，结果准确、有效。

References:

[1] Chen K, Feng DG, Su PE, Zhang XF. Multi-Cycle vulnerability discovery model for prediction[J]. Journal of Software, 2010, 21(9):2367-2375.

[2] Nie CJ, Zhao XF, Chen K, Han ZQ. An software vulnerability number prediction model based on Micro-parameters[J]. Journal of Computer Research and Development, 2011, 48(7):1279-1287.

[3] Gao ZW, Yao Y, Rao F, Liu YZ, Luo P. Predicting model of vulnerabilities based on the type of vulnerability security[J]. Acta Electronica Sinica, 2013, 41(9):1784-1787.

[4] Arbaugh W A, Fithen W L, Mchugh J. Windows of vulnerability: a case study analysis[J]. Computer, 2000, 33(12):52-59.

[5] Frei S, Security Econometrics: The Dynamics of (IN)Security, Ph.D. dissertation at ETH Zurich, 2009

[6] Kaaniche M, Marconato G V, Nicomette V. Security-related vulnerability life cycle analysis[C]// International Conference on Risk and Security of Internet and Systems. IEEE, 2013:1-8.

[7] Song MQ, Wang LL, Yu B. Research on time risk of security vulnerability based on lifecycle theory[J]. Computer Engineering, 2011, 37(1):131-133.

[8] Jumratjaroenvanit A, Teng-Amnuay Y. Probability of Attack Based on System Vulnerability Life Cycle[C]// International Symposium on Electronic Commerce and Security. IEEE Computer Society, 2008:531-535.

[9] Joh H, Malaiya Y K. A Framework for Software Security Risk Evaluation using the Vulnerability Lifecycle and CVSS Metrics[J]. Cvss Metrics Proc International Workshop on Risk & Trust in Extended Enterprises, 2010:430--434.

[10] Shahzad M, Shafiq M Z, Liu A X. A large scale exploratory analysis of software vulnerability life cycles[C]// International Conference on Software Engineering. IEEE Press, 2012:771-781.

[11] Jr C L. Some limitations of "Risk = Threat × Vulnerability × Consequence" for risk analysis of terrorist attacks.[J]. Risk Analysis An Official Publication of the Society for Risk Analysis, 2010, 28(6):1749-1761.

[12] Mkpong-Ruffin I, Umphress D, Hamilton J, et al. Quantitative software security risk assessment model[C]// ACM Workshop on Quality of Protection. ACM, 2007:31-33.

[13] Fu ZY, Gao L, Sun Q, Li Y, Gao N. Evaluation of vulnerability serverity based on rough sets and attributes reduction[J]. Journal of Computer Research and Development, 2016, 53(5):1009-1017.

[14] Houmb S H, Franqueira V N L, Engum E A. Quantifying security risk level from CVSS estimates of frequency and impact[J]. Journal of Systems & Software, 2010, 83(9):1622-1634.

[15] Zhao L, Li JE, Lu TB, Liu KP. Research on quantitative assessment model on vulnerability risk for information system[J]. Journal of Communication, 2009, 30(2):71-76.

[16] Li S, Tryfonas T, Russell G, et al. Risk Assessment for Mobile Systems Through a Multilayered Hierarchical Bayesian Network[J]. IEEE Transactions on Cybernetics, 2016, 46(8):1749-1759.

[17] Fonseca J, Seixas N, Vieira M, et al. Analysis of Field Data on Web Security Vulnerabilities[J]. IEEE Transactions on Dependable & Secure Computing, 2013, 11(2):89-100.

[18] Wiik J, Gonzalez J J, Lipson H F, et al. Dynamics of Vulnerability–Modeling the Life Cycle of Software Vulnerabilities[J]. 2004.

[19] Ciapessoni E, Cirio D, Kjølle G, et al. Probabilistic Risk-Based Security Assessment of Power Systems Considering Incumbent Threats and Uncertainties[J]. IEEE Transactions on Smart Grid, 2016, 7(6):2890-2903.

[20] Li ZJ, Zhang JX, Liao XK, Ma JX. Survey of software vulnerability detection techniques[J]. Chinese Journal of Computer, 2015, 38(4):717-732.

[21] Shizhong W U. Review and outlook of information security vulnerability analysis[J]. Journal of Tsinghua University, 2009, 49:2065-2072.

[22] Schiffman M. Common vulnerability scoring system (CVSS) version3.0. [EB/OL] https://www.first.org/cvss/specification-document.

[23] CVE. Common Vulnerabilities and Exposures [DB/OL] http://cve.mitre.org/.

[24] NIST. National vulnerability database [DB/OL]. https://nvd.nist.gov/.

[25] OSVDB. The Open Source Vulnerability Database [DB/OL] http://osvdb.org/.

[26] Dimitri P, Bertsekas J N. Introduction to Probability [EB/OL]. http://www.sciencedirect.com/science/book/9780128000410.

[27] Beattie S, Arnold S, Cowan C, et al. Timing the Application of Security Patches for Optimal Uptime[C]// Proceedings of the 16th USENIX conference on System administration, Berkeley, CA, 2002: 233-242.

[28] Symantec. [EB/OL] https://www.symantec.com/connect/ru/blogs/what-you-need-know-about-wannacry-ransomware.

附中文参考文献:

[1] 陈恺, 冯登国, 苏璞睿,等. 一种多周期漏洞发布预测模型[J]. 软件学报, 2010, 21(9):2367-2375.

[2] 聂楚江, 赵险峰, 陈恺,等. 一种微观漏洞数量预测模型[J]. 计算机研究与发展, 2011, 48(7):1279-1287.

[3] 高志伟, 姚尧, 饶飞,等. 基于漏洞严重程度分类的漏洞预测模型[J]. 电子学报, 2013, 41(9):1784-1787.

[7] 宋明秋, 王磊磊, 于博. 基于生命周期理论的安全漏洞时间风险研究[J]. 计算机工程, 2011, 37(1):131-133.

[13]付志耀, 高岭, 孙骞,等. 基于粗糙集的漏洞属性约简及严重性评估[J]. 计算机研究与发展, 2016, 53(5):1009-1017.

[15]周亮, 李俊娥, 陆天波,等. 信息系统漏洞风险定量评估模型研究[J]. 通信学报, 2009, 30(2):71-76.

[20]李舟军, 张俊贤, 廖湘科,等. 软件安全漏洞检测技术[J]. 计算机学报, 2015, 38(4):717-732.