arXiv:2301.09892v1 [cs.GT] 24 Jan 2023

具有切换成本的移动目标防御有效策略学习

Vignesh Viswanathan*
马萨诸塞大学阿默斯特分校，美国阿默斯特 vviswanathan@umass.edu

Megha Bose*
海得拉巴国际信息技术学院，印度
megha.bose@research.iiit.ac.in

Praveen Paruchuri
印度海得拉巴国际信息技术学院
praveen.p@iiit.ac.in

摘要

移动目标防御（MTD）已成为各种安全应用中的一项关键技术，因为它剥夺了攻击者通过侦察来利用系统漏洞的能力。然而，该领域现有的大多数研究在制定移动目标防御策略时，都假设能够不切实际地获取有关攻击者动机和／或行为的信息。许多现有方法还假设掌握系统漏洞及攻击者如何利用这些漏洞的完整知识。在本研究中，我们旨在创建无需依赖攻击者先验知识、能够生成有效移动目标防御策略的算法。我们假设，防御者获取自身奖励信息的唯一途径，是在重复博弈环境中与攻击者互动。根据互动中能够获取的信息量，我们采用多臂老虎机建模设计了两种不同的算法，以识别高效策略。随后，我们使用从国家漏洞数据库中挖掘的数据评估这些算法，结果表明，尽管所用信息量少得多，它们的性能仍可与最先进技术相媲美。

关键词 移动目标防御 · 自适应策略 · 重复博弈

1 引言

随着网络系统日益复杂，事先了解特定系统中的所有漏洞变得越来越困难。攻击者通常有充足的时间探查这些漏洞，这使安全分析师面临的情况变得相当复杂。失败造成的后果不仅限于经济损失，还可能导致隐私泄露或拒绝服务，对用户造成严重伤害。移动目标防御 已成为应对 Web 应用程序 [1, 2], 操作系统 [3], 云服务 [4, 5],等领域所面临威胁的一种潜在解决方案。分析人员不再维持单一配置，而是可以维护系统的多种配置并在它们之间切换。这种方法的核心思想是剥夺攻击者所拥有的优势，即随着时间推移针对静态配置进行侦察。近年来，研究界对这种方法表现出浓厚兴趣，已有不少研究（见 [6]）探讨其可行性，并提出在不同配置之间切换的方法。

有些研究将该问题自然地建模为分析者（作为防御者）与多个攻击者之间的博弈，并据此采用博弈论方法来确定切换策略，即在什么阶段切换到哪种配置。然而，许多现有研究假设人们对系统中的漏洞以及攻击者的能力和动机拥有不切实际的大量先验知识。有效的切换策略应当在假设先验知识量合理的同时，令攻击者难以预测。此外，频繁切换或许并不可取，因为防御者每次切换都要根据当前配置和下一配置承担相应成本。

*贡献相同。本文的一个较短版本，题为'不确定性下 *Web* 应用的移动目标防御' 以扩展摘要的形式发表于第*21*届自主代理与多智能体系统国际会议论文集（*AAMAS 2022*），*2022*年*5*月*9*日至*13*日，线上。

1.1 我们的贡献

该领域的一项重要研究思路受到强斯塔克尔伯格均衡 解概念的启发，并对其加以改进[1, 7]。事实上，强斯塔克尔伯格均衡已被证明是在贝叶斯斯塔克尔伯格博弈[8]中采取的最佳策略。然而，计算强斯塔克尔伯格均衡需要完全了解攻击者奖励；在大多数情况下，这一假设并不现实。因此，我们采用了不同的方法，改编多臂老虎机文献中的算法。这种方法有助于我们设计出无需了解攻击者奖励、同时能够帮助我们学习良好策略的算法。

我们提出了两种可扩展的算法，用于为移动目标防御应用生成切换策略——具体采用哪种算法取决于可获取的系统配置相关信息量。第一种算法 *FPL-MTD*（第4.1节）假设对系统中的漏洞以及攻击者实施的漏洞利用一无所知。对于假设拥有此类先验知识的情况，我们提出了 *FPL-MaxMin* （第4.2节），该算法利用有关系统漏洞以及攻击者的知识。我们在多个数据集上评估了所提出的方法，以展示尽管使用的信息少得多，其性能仍可媲美最先进的方法。

1.2 相关工作

我们的工作与 [1], 的研究密切相关，并在其基础上展开。该研究首次提出了 Web 应用程序移动目标防御问题，并提出了一种基于强斯塔克尔伯格解概念的切换策略。该研究的后续论文还探讨了一种基于强化学习的方法，用于生成有效的切换策略 [7]。其他研究通过遗传算法 [9, 10] 提高已部署实现的多样性，并通过强化学习 [11] 尽可能降低总损害，从而计算有效的切换策略。 [11] 也认识到信息不确定性问题，并假设拥有与我们在第4.1节中相同数量的防御者奖励函数相关信息。据我们所知，此前没有研究采用受多臂老虎机文献启发的算法来生成移动目标防御的切换策略。我们在第5节中对所提出的方法与上述所有效用最大化方法进行了实证比较。

其他与移动目标防御相关的研究关注防御者应在何时部署新的实现 [12, 13, 14] ，以及防御者如何创建多个实现以便在它们之间切换 [15, 16, 17, 18, 19]。另一类研究探讨如何在现实场景中应用移动目标防御 [20, 21]。所有这些研究领域都与我们的工作相互独立且互为补充。

2 预备知识

对于任意正整数 t，我们用 $[t]$ 表示集合 $\{1, 2, \ldots, t\}$。附录A中的表格列出了本节引入的所有符号。

博弈模型： 我们将移动目标防御场景建模为由两名不同参与者进行的重复贝叶斯博弈： (a) 部署系统并致力于保护系统的防御者 （记作 θ），以及 (b) 致力于利用漏洞的攻击者（记作Ψ）。这类博弈中通常只有一名防御者，即部署系统的组织（可能由一组分析人员组成）。然而，攻击者可能有多个，他们可以是个人，也可以是由个人组成的团体；这些攻击者各有不同的目标和动机，并试图利用已部署的系统。按照贝叶斯博弈中的常见做法 [8], ，我们将这些不同的攻击者表示为单个攻击者的不同攻击者类型。更正式地说，我们考虑一个包含两名参与者的博弈，即一名防御者和一名攻击者。攻击者有 τ 种类型；我们用集合 $\Psi = \{\psi_1, \psi_2, \ldots, \psi_\tau\}$表示这些类型。我们进一步假设，攻击者集合上存在概率分布 P。防御者可能并不总是知道这一分布。

防御者拥有一组 $C = \{c_1, c_2, \ldots c_n\}$ 可部署的 n 系统配置。系统的各个方面都可以进行更改，以构建不同的系统配置。考虑一个 Web 应用程序示例：其数据库有两种不同的实现方式，一种使用 MongoDB，另一种使用 MySQL。如果 Web 应用程序的其他部分都只有一种实现方式，那么该应用程序将有两种不同的配置：一种使用基于 MongoDB 的数据库，另一种使用基于 MySQL 的数据库。此外，假设该应用程序使用多种编程语言实现——例如，一个版本使用 Python，另一个版本使用 Java。根据兼容性，防御者现在最多可以拥有四种可在其间切换的可部署配置 —— 每种配置分别对应数据库实现方式和编程语言的一种可能组合。这个例子还可以扩展到技术栈的每个部分都有多种实现方式的情况，其中每种实现方式组合都会产生一种新的可部署配置。对于由多个可调架构层组成的设备 [22], ，通过在每一层进行类似的重新配置，可以生成不同的系统配置。在 [23], 所描述的基于网络的系统中，其目标是选择节点的一个 k 子集来部署攻击检测器，不同的系统配置指的是每一个可能的 k 元组节点组合。

正如其字面定义一样，漏洞指配置中可能被攻击者利用、从而破坏系统完整性的不同方面。它们通常取决于系统的硬件和软件特性。每种配置 $c \in C$ 都有一组漏洞 V_c。我们将所有漏洞的集合定义为 $V = \bigcup_{c \in C} V_c$。每种配置的漏洞集不一定互不相交。它们可能并非总能预先得知；不过，我们假设不存在完美的配置，即每种配置都存在某些漏洞。我们将漏洞利用定义为一种方法（或一种算法），攻击者可以利用它来利用某个漏洞，并使防御者付出代价。由于每个漏洞都有相应的漏洞利用方式，因此我们交替使用这两个术语。

奖励：对于每个配置 $c \in C$、每种攻击者类型 $\psi \in \Psi$ 和每个漏洞 $v \in V$，我们定义防御者在第 t 轮的奖励 θ，记作 $r_\theta^t(\psi, v, c)$。如果攻击者类型 ψ 成功利用已部署配置中的某个漏洞，则奖励 $r_\theta^t(\psi, v, c) \in [-1, 0)$；具体值取决于漏洞利用造成的损害。如果攻击者类型 ψ 未能利用已部署配置中的漏洞，防御者获得的奖励为 0。类似地，我们为每种攻击者类型 $\psi \in \Psi$ 定义奖励，记作 $r_\psi^t(v, c)$。$r_\psi^t(v, c) \in (0, 1]$ 如果
攻击者类型 ψ 成功利用已部署配置中的某个漏洞；否则为 $r_\psi^t(v, c)$。由于本文大部分内容讨论的是防御者奖励，因此在提及防御者奖励时，我们省略下标 θ，仅记作 $r^t(\cdot)$。

不同的攻击者类型可能会使防御者获得不同的效用，因为攻击者能力各不相同。例如，某些类型可能不具备实施特定攻击所需的专业技能，或者造成的损害可能不如其他类型严重。需要注意的是，奖励不必始终保持不变；它们可以是随机的，甚至具有对抗性。

重复博弈中有 T 轮。每一轮代表某种攻击者类型尝试进行一次漏洞利用，也可以代表一个固定的时间间隔，间隔结束后防御者会更改其配置。我们定义映射 $f\colon [T] \mapsto [\tau]$，该映射利用攻击者集合上的概率分布 P，将每一轮映射到在该轮发起攻击的攻击者类型。在第 t 轮，防御者选择要部署的配置（记为 $d_t \in C$），攻击者类型 $\psi_{f(t)}$ 选择要利用的漏洞（记为 $a_t^{f(t)}$）。该轮结束时，防御者获得该轮的奖励 $r_\theta^t(\psi_{f(t)}, a_t^{f(t)}, d_t)$，以及其他一些特定于问题的信息。

如果防御者切换其已部署配置，即 $d_t \neq d_{t-1}$，则会产生额外的固定切换成本。该成本由映射 $s\colon C \times C \mapsto [0, 1]$ 给出；从配置 c 切换到配置 c' 所产生的成本为 $s(c, c') \in [0, 1]$。由于此类成本可以在系统测试期间确定，因此我们假设这些成本是先验已知的。对于 Web应用程序，切换成本可以表示两个配置之间存在差异的技术栈组件数量，因为在二者之间切换时需要替换这些组件。对于具有多个架构层的设备 [22],，切换成本可以表示更改某一架构层相较于更改其他架构层的开销（以延迟和/或功耗计），因此，在架构层不同的配置之间进行切换时，如果差异出现在不同的架构层，所产生的成本也会不同。这种方式也可以轻松扩展，以表示多个层同时发生变化的情况。对于前文所述的网络示例 [23],，切换成本可能取决于需要从一个 k 元组重新部署到另一个 k 元组的检测器数量，和/或相关节点的特征。
我们不对攻击者策略作任何假设，因为攻击者可能并非完全理性。我们也不假设了解攻击者效用，因为这些值很难观察到，更难以事先获知。另一方面，假设攻击者能够获取前几轮部署的配置，并利用这些信息决定尝试哪种漏洞利用，这使攻击者更为强大。在多臂老虎机文献中，这通常称为自适应对手。

策略：我们将防御者策略组合定义为一个长度为 T 的元组：$(d_1, d_2, \ldots\ldots, d_T)$，其中 d_t 对应于时间步 t 部署的配置。我们允许 d_t 依赖于前几轮观测到的奖励及其他信息。

类似地，我们将攻击者策略配置定义为一个长度为 T 的元组：$(a_1^{f(1)}, a_2^{f(2)}, \ldots\ldots, a_T^{f(T)})$，其中 $a_t^{f(t)}$ 对应于攻击者类型 $\psi_{f(t)}$ 在时间步 t 尝试的漏洞利用。然而，与防御者不同，攻击者类型可能事先就知道自己的奖励。每种攻击者类型在每轮观测到的唯一额外信息，是防御者在前几轮使用的配置。有时，将 $a_t^{f(t)}$ 定义为函数 $a_t^{f(t)}(d_1, d_2, \ldots\ldots, d_{t-1})$ 会有所帮助，该函数依赖于此前所有的防御者策略。不过，根据上下文能够明确含义时，我们会省略表示漏洞利用的函数参数，仅用 $a_t^{f(t)}$ 表示。我们有时也会省略表示第 t 轮漏洞利用的 $f(t)$，仅用 a_t 表示。

目标：我们研究这样一类情形：防御者不具备完整的先验信息，必须在重复博弈的每一轮结束时根据观察结果进行学习。在此类情形下，我们旨在创建能够生成策略组合的算法，使防御者承担的总成本最小化。该成本可以建模为重复博弈每一轮的奖励与切换成本之和。更正式地说，给定策略 $D = (d_1, d_2, \ldots, d_T)$，

我们将 D 的总效用定义如下

$$TU(D) = \sum_{t=1}^{T}(r^t(\Psi_{f(t)}, a_t^{f(t)}, d_t) - s(d_{t-1}, d_t))$$

其中， r^t 表示第 t 轮的奖励，而 s 表示切换成本。本文的目标是开发能够输出策略组合、使上述表达式最大化的算法。然而，如果我们事先不知道每种可能的 $f(t)$所对应的函数 $a_t^{f(t)}(d_1, \ldots, d_{t-1})$，就无法最大化这样的函数。因此，我们采用经验方法，开发在现实问题实例上具有良好经验性能的算法。

3 多臂老虎机与遗憾

当每一轮的奖励由对手选择时，很容易看出，移动目标防御可以建模为对抗性多臂老虎机问题。每种配置都可以视为多臂老虎机问题实例中的一条臂；与多臂老虎机问题一样，我们在部署一种配置（拉动一条臂）后观察奖励。本节讨论多臂老虎机文献中的理论结果，这些结果适用于我们的问题。关于老虎机算法及其理论保证的更详细总结，请参阅 [24]。

将我们的问题表述为多臂老虎机问题后，便可以使用 *FPL+GR*[25] 或 Exp3 [26] 等高效算法，获得具有良好外部遗憾 保证的策略。策略组合的外部遗憾可以定义为该策略组合的总效用与事后最佳固定策略的效用之差；在我们的语境中，固定策略指永久部署单一配置。*FPL+GR* 和 Exp3 算法都保证输出具有 $O(\sqrt{T})$ 期望外部遗憾的策略；当 T 足够大时，此类策略的总效用会收敛到事后最佳固定策略的总效用。事实上，这些结果也适用于博弈。对于零和博弈，众所周知，老虎机策略的平均奖励会收敛到博弈的值 [24, 第 9] 章。 [27] 表明，略作修改的*FPL+GR*算法能够在重复安全博弈中最小化外部遗憾。这些结果表明，老虎机算法能够生成效率极高的策略。

然而，将切换成本纳入问题后，老虎机算法便失去了理论保证。主要问题在于，算法本身并未考虑切换成本；因此，要将这些算法用于我们的场景，就必须将切换成本纳入奖励函数。经过这一修改，我们得到如下遗憾保证：

$$\max_{d \in C}\left(\sum_{t=1}^{T}(r^t(\Psi_{f(t)}, a_t^{f(t)}(d_1, d_2, \ldots, d_{t-1}), d) - s(d_{t-1}, d))\right) - \left(\sum_{t=1}^{T}(r^t(\Psi_{f(t)}, a_t^{f(t)}, d_t) - s(d_{t-1}, d_t))\right) \leq O(\sqrt{T})$$

从上述表达式可以看出，正如 [28], 所论证的那样，在存在切换成本时，外部遗憾这一概念毫无意义。对于存在切换成本的问题实例， [28] 提出了一种称为策略遗憾的替代遗憾形式。该遗憾会考虑这样一种情况：如果防御者采用纯策略，攻击者会如何行动；随后将老虎机算法的总效用与事后表现最佳的纯策略的总效用进行比较。策略组合的策略遗憾由以下表达式给出：

$$\max_{d \in C}\left(\sum_{t=1}^{T} r^t(\Psi_{f(t)}, a_t^{f(t)}(d, d, \ldots, d), d)\right) - \left(\sum_{t=1}^{T}(r^t(\Psi_{f(t)}, a_t^{f(t)}(d_1, d_2, \ldots, d_{t-1}), d_t) - s(d_{t-1}, d_t))\right)$$

[28] 表明，面对能够利用先前轮次信息的对手，任何算法都无法保证次线性的策略遗憾。换言之，在存在自适应对手的情况下，所有算法的策略遗憾都将达到 $\Omega(T)$。由于策略遗憾可能达到的最大值为 $2T$，每种算法都能保证一个紧确的遗憾界$O(T)$。因此，无法根据遗憾保证的强弱来区分算法。

尽管缺乏有意义的理论保证，但鉴于这些方法在其他重复博弈场景中的表现（无论是理论上还是实证上）， [27], 我们相信，流行的老虎机方法在我们的实际问题设置中仍会表现良好。因此，我们借鉴这些算法，设计移动目标防御策略。

4 个效用最大化启发式方法

本节提出两种受多臂老虎机文献中算法启发的算法。我们的两种算法都不需要攻击者奖励信息。我们出于以下两个原因确保这一点：

（a）在许多情况下，防御者对攻击者类型及其奖励函数几乎没有先验知识。例如，某些软件提供以前可能从未提供过的新颖服务，就属于这种情况。这类应用通常几乎没有关于自身漏洞的数据，因此无法推断出可能试图利用该应用的不同类型攻击者。除此之外，动态应用程序可能没有固定不变的漏洞集，随着应用程序的发展，还会不断出现新的漏洞；防御者无法对这类应用程序的漏洞做出任何假设。

（b）其次，即使事先已知每种配置的漏洞和攻击者奖励，非理性攻击者的行动也未必会以任何有意义的方式反映这些奖励。例如，非理性攻击者可能不会在每一轮都遵循强斯塔克尔伯格均衡策略。在这种情况下，最好不要假设攻击者会采取任何特定策略，而是通过观察来学习攻击者的策略。

4.1 FPL-MTD

我们的第一个算法适用于上文（a）点所述的应用场景。该算法不假设任何关于可部署配置中的漏洞或试图利用系统的不同攻击者的先验知识。算法1给出了*FPL-MTD*的步骤。

FPL-MTD 为每种配置维护一个奖励估计，该估计大致对应于防御者部署该配置时可获得的预期奖励。在每一轮中，算法要么以一定概率随机选择一个臂进行探索，要么选择奖励估计最高的臂。

算法1 FPL-MTD

1：输入：超参数 η **和** γ**2：** $\hat{r}_c^1 \leftarrow 0 \quad \forall c \in C$**3：对于** t**，从 1 到** T**，执行 4：采样** $q \sim$ **伯努利分布(**γ**) 5：如果** $q = 1$**，则 6：令** d_t **为均匀随机采样的配置 7：否则 8：采样** $z_c \sim$ **exp(**η**)** $\forall c \in C$**9：** $u_c \leftarrow \hat{r}_c^t - z_c \quad \forall c \in C$**10：** $d_t \leftarrow$ **max**$_{c \in C}$**(**$u_c - s$**(**d_{t-1}, c**)) 11：结束 if 12：未知类型的对手** $\psi_{f(t)}$ **执行未知操作** a_t**，使防御者获得奖励** r^t**(**$\psi_{f(t)}, a_t, d_t$**) 13：运行 GR-MTD 以获得** K**(**d_t**) 14：** $\hat{r}_d^{t+1} \leftarrow \frac{1}{t}\big[($ $t-1$**)**$\hat{r}_d^t + K$**(**d_t**)**r^t**(**$\psi_{f(t)}, a_t, d_t$**)**$\mathbb{I}\{d = d_t\}\big]$**15：结束 for**

配置选择： 受*FPL+GR* 算法的启发，我们的算法采用经典的跟随扰动领袖（Follow the Perturbed Leader）形式来选择配置 [29, 30] 。*FPL+GR* 算法为每个配置 c 和时间步 t维护一个估计奖励 $\hat{r}_c^t$。在每个时间步 t，算法都会对这些奖励进行扰动，并选择扰动后奖励最高的配置。我们的方法与*FPL+GR* 的关键区别在于，我们考虑了切换成本。*FPL+GR* 仅选择使预期奖励最大化的配置，而我们选择的是使估计平均奖励与防御者将承担的切换成本之差最大化的配置（第 10 行）。

奖励估计： 一种自然的防御者奖励估计 $\hat{r}_c^t$ 是：对每个配置 c，采用其在此前所有时间步中部署时获得的平均奖励。更正式地说，在时间步 t，我们希望将奖励估计为 $\hat{r}_c^t$，即

$$\hat{r}_c^t = \frac{1}{t-1} \sum_{i \in [t-1]} r^i(\psi_{f(i)}, a_i, c) \tag{1}$$

其中，r^i 表示时间步 i 的奖励函数。由于我们会在不同配置之间切换，因此无法观测未部署配置的奖励。因此，我们无法直接使用这个值。不过，通过对

动作集合进行随机化，我们可以使奖励估计成为式（1）中该值的无偏估计。更具体地说，奖励估计

$$\hat{r}_c^t = \frac{1}{t-1} \sum_{i \in [t-1]} \frac{r^i(\psi_{f(i)}, a_i, c)}{\Pr[d_i = c]} \mathbb{I}\{d_i = c\} \tag{2}$$

是（1）中所需奖励的无偏估计，其中 $\Pr[d_i = c]$ 表示将 c 选作策略 d_i 的概率，而 $\mathbb{I}\{d_i = c\}$ 是一个指示函数：当 $d_i = c$ 时取值为1，否则取值为0。

然而，为了对每种配置都获得良好的估计，我们需要在算法中引入随机性，使得在任意给定的时间步都可能选择不同的配置；换句话说，我们需要进行探索。我们通过两种不同的方式实现这一点。首先，在每个时间步，我们使用均值为 η的指数分布对奖励估计进行扰动。其次，在每个时间步，我们以概率 γ随机选择一种配置。可以通过超参数 η 和 γ来控制引入的随机性程度。

此外，由于算法的复杂性，无法高效地计算概率分布 $\Pr[d_t]$ 的闭式表达。因此，我们用其无偏估计替代值$^1{}_{\Pr[d_t=c]}$。可以使用一种经过充分研究的方法[25]来计算该估计，即几何重采样（见算法2）。这种方法的核心直觉是：通过反复模拟算法1中的 FPL 步骤，直到观察到相同的配置，我们可以模拟一个期望值为$^1{}_{\Pr[d_t=c]}$的几何分布。因此，模拟终止所需的迭代次数 ——— 是一个无偏估计值 $^1{}_{\Pr[d_t=c]}$。~~该子程序~~的详细描述见算法2。

算法2 GR-MTD

$K(d_t) \leftarrow M$
对于 k，**从** 1 **到** M，**执行**
　按照算法1中的第4至11行生成　　$\tilde{d}$ 作为 d_t的模拟
　if $\tilde{d} = d_t$ **则**
　　$K(d_t) \leftarrow \min(K(d_t), k)$
　结束 if
结束 for
返回 $K(d_t)$

理论上，几何重采样可能需要无限长的时间，但实际上，我们只能运行有限的 M 次。不过，由于我们的算法包含均匀探索机制，选择任意配置的概率都有一个不低于 $\frac{\gamma}{|C|}$ 的下界。因此，对于 $M \geq \binom{|C|T}{\gamma}$，几何重采样子程序将以至少 $1 - e^{-T}$ 的指数级高概率终止。（形式化陈述及其证明已放在附录中。）

4.2 FPL-MaxMin

虽然算法1假设我们对可部署配置的漏洞和攻击者类型没有任何先验信息，但可能存在一些应用能够获得有关系统配置漏洞的先验信息。我们提出第二种算法 *FPL-MaxMin* ，适用于可能较旧且经过充分研究的应用；在这类应用中，系统中的漏洞集合以及攻击者数量及其分布 P 均事先已知。该算法假设防御者除了观察奖励外，还能观察到每个时间步由哪名攻击者发起攻击，以及该攻击者试图利用哪个漏洞。该算法唯一无法观察到的信息是每轮的攻击者奖励。

我们在算法 3 中给出了*FPL-MaxMin* 的步骤。与*FPL-MTD* 类似，我们维护估计值，并根据估计值选择配置。不过，*FPL-MaxMin* 维护的是每个漏洞—攻击者类型对的估计值，随后使用最大最小策略选择配置。这样做主要是为了确保鲁棒性；通过采用最大最小策略，该算法可以防范最坏情况下的攻击者策略，因此无论攻击者利用哪个漏洞，都能保证获得合理的奖励。

奖励估计：与前一个算法类似，我们为每个漏洞和攻击者类型对（v， ψ）定义一个奖励估计：

算法 3 FPL-MaxMin

输入： 超参数 η 和 γ
$\hat{r}^1_{v,\psi} \leftarrow 0 \quad \forall v \in V, \psi \in \Psi$
对于 t，**从**1 **到** T，**执行**
  采样 $q \sim$ 伯努利分布(γ)
  如果 $q = 1$，**则**
    令 d_t 为均匀采样的配置
  else
    为 $v \in V, \psi \in \Psi$采样 $z_{v,\psi} \sim \exp(\eta)$
    $u_c \leftarrow \sum_{\psi \in \Psi} \min_{v \in V_c} P_\psi(\hat{r}^t_{v,\psi} - z_{v,\psi}) \quad \forall c$
    $d_t \leftarrow$ 最大值$_{c \in C}(u_c - s_{d_{t-1},c})$
  结束 if
  类型为 $\psi_{f(t)}$ 的对手执行 a_t，使防御者获得奖励 $r^t(\psi_{f(t)}, a_t, d_t)$
  $p_{v,\ \psi_{f(t)}} \leftarrow P_{\psi_{f(t)}} \frac{|i \in [t]:\ f\ (i)\ = \psi_{f(t)} \wedge a_i = v|}{|i \in [t]:\ f\ (i)\ = \psi_{f(t)}|} \quad \forall v \in V$
  $n_v \leftarrow |\{i :\ v \in V_{d_i}\}| \quad \forall v \in V$
  $\hat{r}^{t+1}_{v,\ \psi_{f(t)}} \leftarrow \frac{1}{n_v} \sum_{i \in [t]} \frac{r^i(\psi, v}{p_{v,\psi_{f(t)}}} \mathbb{I}\{a_i, \psi_{f(i)}{}_{,d_i}) = v, \psi_{f(t)}\}$
结束循环

$$\hat{r}^t_{v,\psi} = \frac{1}{|\{i \in [t-1] : v \in V_{d_i}\}|} \sum_{i \in [t-1]} r^i(\psi, v, d_i) \tag{3}$$

这对应于这样一种平均奖励：如果攻击者类型 ψ 每一轮都利用漏洞 v，而防御者部署的是具有漏洞 v 的配置，那么防御者将获得这一平均奖励。请注意，当 v 不是 d_i 的漏洞时，$r^i(\psi,\ v,\ d_i) = 0$。这种方法很有吸引力，因为它利用了多个配置可能具有相同漏洞这一事实，因此我们可以结合针对不同配置的攻击信息，来估计该特定漏洞的奖励。

由于这 个值无法准确计算，因此可以按如下方式计算该值的无偏估计：

$$\hat{r}^t_{v,\psi} = \frac{1}{|\{i \in [t-1] : v \in V_{d_i}\}|} \sum_{i \in [t-1]} \frac{r^i(\psi, v, d_i)}{\Pr[a_i, \psi_{f(i)} = v, \psi]} \mathbb{I}\{a_i, \psi_{f(i)} = v, \psi\} \tag{4}$$

其中，$\Pr[a_t, \psi_{f(t)} = v, \psi]$ 表示攻击者类型为 ψ 的攻击者在时间步 t 尝试利用漏洞 v 的概率。$\mathbb{I}\{a_i, \psi_{f(i)} = v, \psi\}$ 是一个指示函数：如果攻击者类型 ψ 在时间步 i 利用漏洞 v，则取值为 1，否则取值为 0。这就是我们使用的奖励估计。

与上一节类似，$\Pr[a_t, \psi_{f(t)} = v, \psi]$ 无法精确计算。不过，原因有所不同。式 (2) 依赖于算法随机化所决定的概率，而式 (4) 依赖于各攻击者类型所采用的随机化，以及对攻击者类型集合进行的随机化。

由于这一差异，我们无法使用几何重采样，因为无法从攻击者那里采样漏洞利用，以推断其混合策略。因此，我们用攻击者类型迄今为止采用的经验混合策略来近似其混合策略。更正式地说，我们将 $\Pr[a_t, \psi_{f(t)} = v, \psi]$ 近似为

$$\Pr[a_t, \psi_{f(t)} = v, \psi] \approx P_\psi \frac{|i \in [t] : \psi_{f(i)} = \psi \wedge a_i = v|}{|i \in [t] : \psi_{f(i)} = \psi|} \tag{5}$$

利用关于 a_t 和 $\psi_{f(t)}$的信息，我们还使用上述表达式更新所有轮次 $i < t$ 的 $\Pr[a_i, \psi_{f(i)} = v, \psi]$ 估计值。根据这些更新后的概率估计，我们在每轮结束时使用 (4) 重新计算奖励估计。请注意，当某种攻击者类型不进行随机化而采用纯策略时，使用 (5) 计算的奖励估计会反映这一纯策略；也就是说，只有该攻击者类型所采用的纯策略对应的奖励估计非零。其他奖励无关紧要，因为该攻击者类型不会采用其他策略。

配置选择： 当我们只有一种攻击者类型（例如 ψ）时，可以说，使用针对每个漏洞的奖励估计来选择配置的最稳健方法是最大最小策略——将每个配置的值估计为该配置中奖励最低的漏洞所对应的奖励。然后，在切换成本的约束下，选择值最高的配置。

将配置的奖励定义为该配置中奖励最低的漏洞的奖励。然后，在考虑切换成本的情况下，选择值最高的配置。更正式地说，这等价于评估为

$$d_t = \arg\max_{c \in C} \min_{v \in V_c} \left(\hat{r}^t_{v,\psi} - s(d_{t-1}, c) \right) \tag{6}$$

当存在多种攻击者类型且已知这些类型的概率分布时，我们可以利用这一信息，将配置的值估算为该配置对每种类型所能提供的最低奖励漏洞的加权平均值。我们根据攻击者在给定轮次发动攻击的概率，对每种攻击者类型的估算奖励赋予权重。由此得到以下配置选择策略：

$$d_t = \arg\max_{c \in C} \left(\left[\sum_{\psi \in \Psi} P_\psi \min_{v \in V_c} \hat{r}^t_{v,\psi} \right] - s(d_{t-1}, c) \right) \tag{7}$$

这是主要的配置选择方法 。我们还保留了算法 3 中所用算法的随机化，以
进一步增加防御者策略的不确定性nty 关于防御者策略。

最后，我们指出，关于攻击者数量的先验知识这一假设可以放宽。如果防御者不知道攻击者的数量，或者无法观察到任意给定轮次中是哪位攻击者发动了攻击，我们仍可以在假设只有一名攻击者的情况下使用算法 3。

5 实验结果

本节中，我们将自己的算法与基于国家漏洞数据库（NVD）信息构建的数据集上的现有方法进行比较。NVD包含有关全球计算机系统所使用的不同硬件和软件组件中存在的漏洞的数据。这些数据可用于构建合成问题实例，以评估我们的算法。其他研究移动目标防御在 Web 应用程序中的应用的工作也采用了这种评估方法 [1, 7]。更具体地说，我们将自己的方法与以下**防御者策略**进行比较：

BSS-Q 是一种由 [7], 提出的基于强化学习（RL）的方法，它能够随时间学习一种混合策略，但假设已知每个时间步获得的攻击者奖励。由于我们假设不知道每个时间步的攻击者奖励，因此我们对该算法进行训练，共训练50个长度为10的回合，并使用输出策略进行评估。由于对于大型问题实例而言，这一训练过程非常缓慢，因此我们只训练少量回合；每个回合都需要求解10个混合整数二次规划问题。训练过程让该算法能够完全访问攻击者和防御者的奖励，并假设攻击者在给定防御者混合策略的情况下采用 ϵ-贪心策略。

S-OPT 是 [1] 提出的一种基于斯塔克尔伯格解的方法。该算法假设对攻击者和防御者奖励有完整的了解，并计算出一种在所有时间步中使用的混合策略。

RobustRL 是 [11], 提出的一种基于强化学习的方法，与我们的方法类似，无需任何关于攻击者或每种配置中漏洞的先验信息。

S-Exp3 是一种先进的老虎机算法，用于针对非自适应对手对切换成本进行建模 [28], ，同时假设没有关于攻击者或漏洞的先验信息。我们采用 [28], 提出的方法修改 Exp3 算法 [26] ，以便在存在切换成本且面对非自适应对手时，将策略遗憾降至最低。

FPL+GR 是一种多臂老虎机算法，仅通过获取样本来进行重要性加权。该算法假设没有关于攻击者或漏洞的先验信息，并从观察到的防御者奖励中学习 [25]。

BiasedASLR 是一种受地址空间布局随机化（ASLR）技术启发的随机化算法 [31]。基于 ASLR 的技术通常会最大化地址选择的熵，这在我们的场景中等同于配置选择；此外，我们还会偏向那些漏洞被利用次数较少的配置，使随机分布向这些配置倾斜。更正式地说，在第 t 轮，*BiasedASLR* 会以与某配置中的漏洞在前 $t-1$ 轮内被利用的次数成反比的概率，随机选择一个配置。

BSS-Q 和*S-OPT* 是在事先完全了解奖励和漏洞信息的情况下最先进的算法。然而，二者都做出了关于攻击者奖励和计算能力的不现实假设（都需要求解混合整数二次规划）。此外，二者都不是最优的：*BSS-Q* 需要不切实际的大量训练才能收敛到最优解，而*S-OPT*如 [7], 所述，由于将问题建模为单阶段正规形式贝叶斯斯塔克尔伯格博弈，因此是次优的。这种建模方式无法最优地刻画博弈的序贯性质。

据我们所知，*RobustRL* 是最少信息场景下最先进的技术。*RobustRL*、*S-Exp3*和*FPL+GR* 假设其能够获取的信息与 *FPL-MTD* 算法同样多。然而，它们都没有考虑切换成本；因此，在每个时间步，我们都会向其提供一个经过修改的奖励，该奖励考虑了上一次迭代的切换成本： $r^t(\Psi_{f(t)}, a_t, d_t) - s(d_{t-1}, d_t)$。*BiasedASLR* 是我们唯一一个完全不使用奖励的基线方法，它只统计攻击者尝试利用特定配置中漏洞的次数。我们针对以下**攻击者策略**评估上述每种算法：

最佳响应： 在每个时间步，攻击者都会根据防御者基于此前所有时间步中观察到的配置所采用的经验混合策略，选择最适合利用的漏洞。

FPL-UE： 在每个时间步，攻击者都会根据 FPL-UE 算法选择一种策略 [27].

斯塔克尔伯格： 在每个时间步，攻击者都会采用使用 DOBSS 算法计算出的最优斯塔克尔伯格解[8]。

随机： 在每个时间步，攻击者都会从其可利用的漏洞列表中均匀随机地选择一个漏洞进行利用。

量化响应（QR）： 在每个时间步，攻击者都会使用 QR 模型选择一个漏洞进行利用 [32]。在安全博弈文献中，QR 模型一直用于对人类（及有限理性）对手进行建模 [33, 34]。

偏置随机： 在时间步 t，攻击者会随机选择一个漏洞进行利用，选择概率与该漏洞利用在前 $t-1$ 轮中本可成功的次数成正比。这是 BiasedASLR 策略在攻击者方面的对应策略。

我们使用上述攻击者策略，在随机生成的数据集上评估这些防御者策略；这些数据集的奖励分布（大致而言）遵循国家漏洞数据库中现有漏洞的奖励分布。通过针对某一特定攻击者策略（例如最佳响应）进行评估，我们指的是：在所有攻击者类型均遵循最佳响应策略的实例中评估我们的算法。我们还研究了如何在附录 D 中使用我们的算法来决定修复哪些漏洞。我们也在随机生成的问题实例以及附录 C 中 [1] 使用的问题实例上评估了我们的策略。

我们使用上述攻击者策略，在以下数据集上评估这些防御者策略：(a) 由 [1], 创建的玩具数据集，该数据集使用国家漏洞数据库（NVD）中的漏洞生成奖励（第 5.2 节）；(b) 更大的随机生成数据集，其奖励分布大致遵循基于国家漏洞数据库中现有漏洞的通用漏洞评分系统（CVSS）的奖励分布（第 5.3 节）。

通过采用特定的攻击者策略（例如最佳响应）进行评估，我们考察算法在所有攻击者类型都遵循最佳响应策略的实例中的表现。由于防御者奖励为负，为便于说明，我们绘制所讨论算法的总效用与均匀随机策略的总效用之差。我们将该值称为算法的性能；直观而言，它表示该算法相较于均匀随机选择部署配置的策略好多少。请注意，均匀随机策略是一种最大熵策略，因此也可作为一种基线 ASLR 技术。

5.1 超参数选择

为了选择超参数 η 和 γ，我们生成 5 个随机零和博弈实例，以测试不同的超参数组合；由于篇幅限制，省略数据集生成的具体细节。我们考虑 $\gamma \in \{0.001, 0.002, 0.003, \ldots, 0.02\}$ 和 $\eta \in \{0.01, 0.02, \ldots, 0.1\}$的所有组合。

对于每种超参数组合，我们在全部 5 个零和数据集上运行算法，每个数据集运行 500 轮，共重复 5 次，并计算平均总效用。通过这种筛选方法，我们确定*FPL-MTD* 的理想参数组合为$\gamma_{\text{MTD}} = 0.007$ 和 $\eta_{\text{MTD}} = 0.1$。同样，我们确定 *FPL-MaxMin* 的理想参数组合为 $\gamma_{\text{MaxMin}} = 0.006$ 和 $\eta_{\text{MaxMin}} = 0.03$。

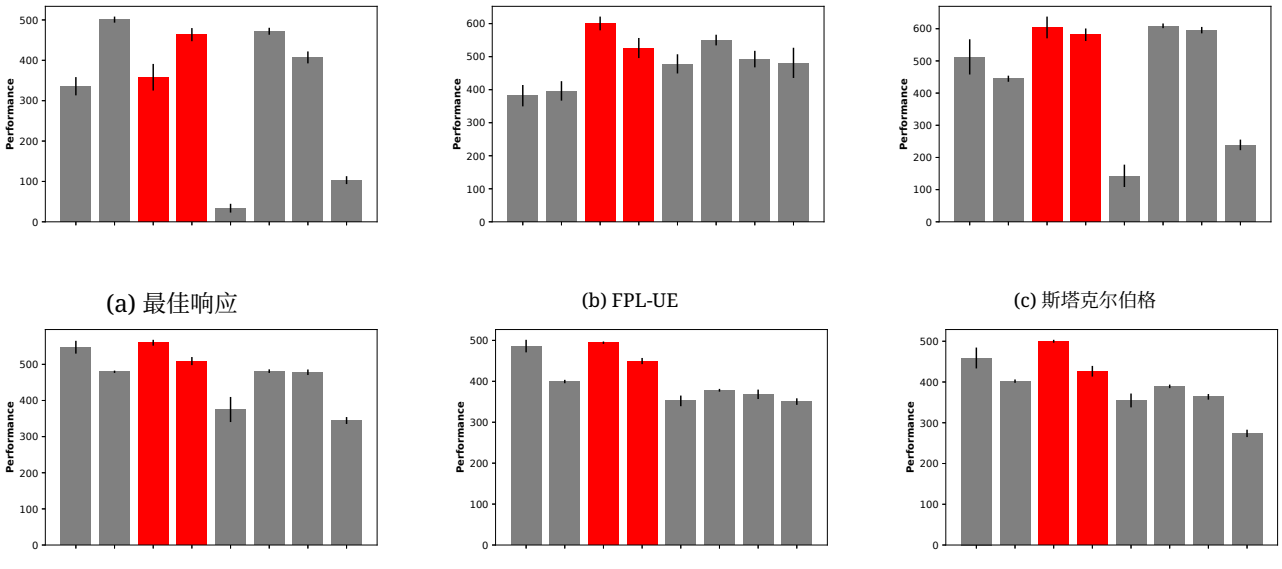
5.2 小型 NVD 数据

此数据集由 [1], 创建，包含 4 种可部署配置、3 种攻击者类型以及略少于 300 个漏洞的奖励。漏洞及每个漏洞对应的奖励均根据国家漏洞数据库的通用漏洞评分系统生成（更多信息请参见 [1]）。

由于该数据集相对较小，我们使用 100 回合训练*BSS-Q* 。我们让每种防御者策略分别对抗每种攻击者策略，运行 1000 个时间步，并在图 1 中绘出每种算法 10 次运行的平均性能。每个柱状图顶部的黑线表示性能的标准误差。

算法	漏洞	攻击者类型	防御者奖励（先验）	防御者奖励（通过观测）	攻击者奖励
BSS-Q	✓	✓	✓		✓
S-OPT	✓	✓	✓		✓
FPL-MaxMin	✓	✓		✓	
FPL-MTD				✓	
RobustRL				✓	
S-Exp3				✓	
FPL+GR				✓	
BiasedASLR				✓	

表 1：已实现算法的信息需求。



(a) 最佳响应　(b) FPL-UE　(c) 斯塔克尔伯格

(d) 随机　(e) 量化响应　(f) 偏置随机

图 1：算法在小型 NVD 数据上的性能。每张图中从左到右依次为：BSS-Q、S-OPT、FPLMaxMin、FPLMTD、RobustRL、S-Exp3、FPL+GR、BiasedASLR。性能定义为算法的总效用与均匀随机算法的总效用之差。

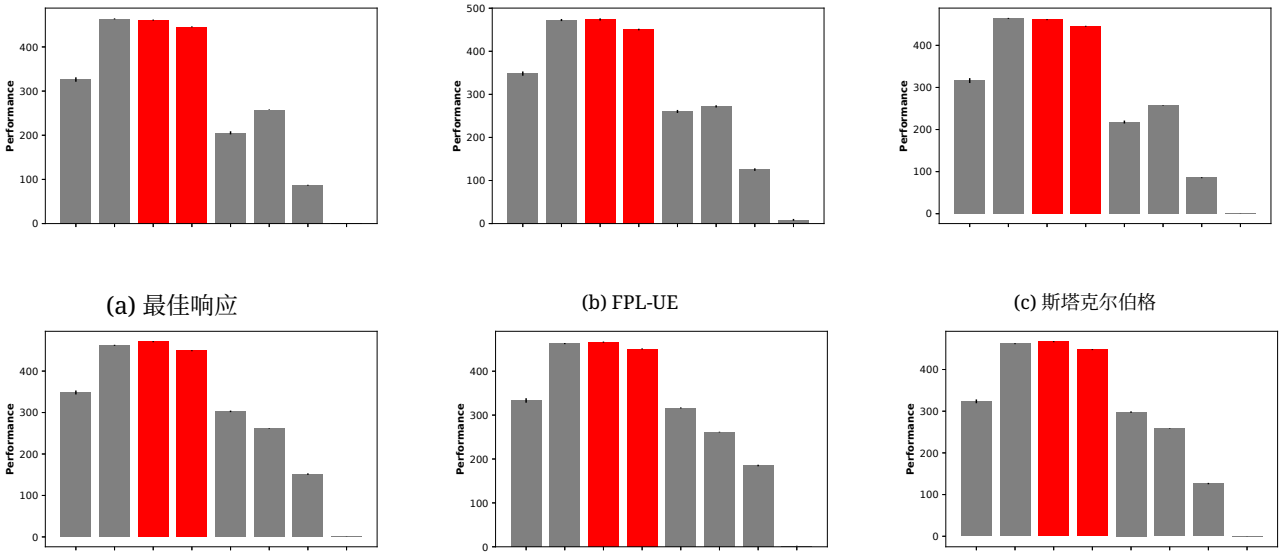
从图 1 的图表中可以看出，没有哪种防御者策略能够明确胜过其他所有防御者策略。在不需要先验知识的基线方法中，面对斯塔克尔伯格攻击者策略时，*S-Exp3* 胜过其他所有基线方法，包括我们的算法（见图 1c）。面对其他攻击者时，我们的算法性能相近，或优于现有的老虎机方法；其中，面对随机攻击者策略时，*FPL-MaxMin* 胜过其他所有防御者策略（见图 1d）。采用完全信息的最先进算法*S-OPT* 似乎表现不佳。进一步检查表明，这是由高昂的切换成本导致的。*S-OPT* 方法在 1000 个时间步中切换约 500 次，因此产生了高昂的切换成本。这导致*S-OPT* 性能不佳，正如图 1 所示。

基于使用该数据集进行的实验，我们发现，就漏洞数量而言，其中两种配置比另外两种更理想。此外，实例规模较小，仅包含 4 种配置。这两个因素使算法更容易进行学习并实现收敛。因此，下面我们使用规模更大的数据集展示更多结果。

5.3 基于大型 NVD 的数据

为了给算法设计更高难度的基准，我们创建了包含规模大得多的问题实例的数据集，其最优解需要在多个配置之间切换。由于涉及多个配置，规模更大的数据集能够支持更贴近实际的评估。构建这些数据时的主要限制是 *S-OPT* 和*BSS-Q* 方法的复杂性；计算这些策略需要指数级时间，因此对于规模大得多的博弈实例而言并不可行。

我们按如下方式生成数据集：配置数量在 10 到 20 之间均匀随机选取，攻击者数量在 3 到 6 之间均匀随机选取，漏洞数量在



(a) 最佳响应　(b) FPL-UE　(c) 斯塔克尔伯格

(d) 随机　(e) 量化响应　(f) 偏置随机

图 2：算法在基于大型 NVD 数据的数据集上的性能。每张图中从左到右依次为：BSS-Q、S-OPT、FPLMaxMin、FPLMTD、RobustRL、S-Exp3、FPL+GR、BiasedASLR。性能定义为算法的总效用与均匀随机算法的总效用之差。

500 到 800 之间均匀随机选取。每个漏洞的奖励从国家漏洞数据库为各漏洞给出的“评分”的经验分布中抽样。每个数据集对应一个合成问题实例，可用于评估我们的算法。数据集生成的详细流程见附录 E.1.1。

我们生成 20 个这样的数据集，并针对每个数据集，将我们的每种算法分别运行 10 次，每次运行 1000 个时间步，然后测量平均性能。图 2 展示了这 20 个数据集上的平均性能及其标准误差界。由于迭代次数很多，标准误差相当低，因此可能无法在所有算法的图中清晰看到。由此可见，我们对平均性能的估计接近真实的平均性能。

图 2 中的曲线显示，不同攻击者策略下存在明显的趋势。*S-OPT* 表现非常出色，优于*BSS-Q*；我们认为这是因为*BSS-Q* 收敛速度较慢，即使经过 50 个回合仍未能收敛到最优策略。在四种攻击者策略下，我们发现，尽管算法*FPL-MaxMin*不使用任何关于奖励的先验信息，而算法*FPL-MTD* 除了不使用关于奖励的先验信息外，还假设对系统漏洞没有先验知识，但二者的性能均与使用所有奖励和漏洞先验信息的*S-OPT* 相近。在少数情况下，我们的方法甚至略胜*S-OPT* 。我们的算法性能也显著优于其他基线方法*RobustRL*、*S-Exp3*、*FPL+GR* 和 *BiasedASLR*。具体而言，面对最佳响应攻击者策略时，*FPL-MTD* 的性能分别比 Robust-RL、*S-Exp3* 和*FPL+GR* 高 116%、72% 和 412%。面对其他攻击者策略时，我们也得到类似的百分比（如图 2 所示）。*BiasedASLR* 算法的性能接近 0，显著逊于所有其他算法；这表明该算法的平均总效用与均匀随机算法相同。

在现实场景中部署所提出的算法时，一个潜在的顾虑是，算法需要经过一定数量的交互才能学习到合适的奖励估计。在此期间，防御者奖励以及已部署配置可能会高度依赖切换成本。因此，已部署配置在试图避免切换时，可能会遭受更多攻击。可以通过一种实现选择来解决这一问题：在初始的若干时间步中，将 γ 设为较高的值，从而减轻初期对切换的抵触所带来的影响。

6 结论与未来工作

在本研究中，我们探讨了这样一个问题：当防御者无法全面了解系统漏洞以及不同

攻击者类型的策略时，如何为移动目标防御制定高质量的切换策略。基于经典的多臂老虎机算法 FPL，我们提出了两种可扩展的算法：*FPL-MTD*和*FPL-MaxMin*。这两种算法分别针对防御者可能预先掌握的两种不同程度的信息进行规划，并据此制定有效的切换策略。我们利用国家漏洞数据库中的数据展示了，与使用相同信息量的现有算法相比，我们的方法性能显著更优。尽管所用信息远少于*S-OPT*，它们的性能仍略优于当前最先进的方法*S-OPT*；这是因为我们无需预先了解漏洞和攻击者奖励，只需依据当前攻击影响观察到的防御者奖励。未来，我们计划为有限理性攻击者开发切换策略。在安全博弈领域，已有基于这一假设构建的算法，例如 SUQR [34]，我们计划将其应用于此处。未来研究的另一个方向是探索基于老虎机的方法在其他移动目标防御应用中的作用，例如入侵检测系统和网络安全。

参考文献

[1] Sailik Sengupta、Satya Gautam Vadlamudi、Subbarao Kambhampati、Adam Doupé、Ziming Zhao、Marthony Taguinod 和 Gail-Joon Ahn。面向 Web 应用程序移动目标防御的策略生成：一种博弈论方法。载于 第*16*届自主智能体与多智能体系统国际会议（*AAMAS*）论文集，第178—186页，2017年。[2] Michael Thompson、Marilyne Mendolla、Michael Muggler 和 Moses Ike。用于移动目标防御的动态应用轮换环境。载于*2016*年韧性周（*RWS*），第17–26页，2016年。[3] Joshua A. Lyle 和 Nathaniel Evans。软件定义网络多操作系统轮换环境。2020年4月。[4] Miguel Villarreal-Vasquez、Bharat Bhargava、Pelin Angin、Noor Ahmed、Daniel Goodwin、Kory Brin 和 Jason Kobes。面向云系统的基于移动目标防御的自适应韧性方法。载于 *2017*年*IEEE*第*10*届云计算国际会议（*CLOUD*），第723–726页，2017年。[5] Wei Peng、Feng Li、Chin-Tser Huang 和 Xukai Zou。针对异构动态攻击面的云服务移动目标防御策略。载于 *2014*年*IEEE*国际通信会议（*ICC*），第804–809页，2014年。[6] Jin-Hee Cho、Dilli P. Sharma、Hooman Alavizadeh、Seunghyun Yoon、Noam Ben-Asher、Terrence J. Moore、Dong Seong Kim、Hyuk Lim 和 Frederica F. Nelson。迈向主动自适应防御：移动目标防御综述。*IEEE*通信调查与教程，22：709–745，2020年。[7] Sailik Sengupta 和 Subbarao Kambhampati。用于自适应移动目标防御的贝叶斯斯塔克尔伯格马尔可夫博弈中的多智能体强化学习，2020年。[8] Praveen Paruchuri、J onathan P. Pearce、Janusz Marecki、Milind Tambe、Fernando Ordonez 和 Sarit Kraus。为安全而博弈：求解贝叶斯斯塔克尔伯格博弈的高效精确算法。载于 第*7*届自主智能体与多智能体系统国际会议（*AAMAS*）论文集，第895–902页，2008年。[9] Michael B. Crouse、E. Fulp 和 D. Cañas。改进基于遗传算法的移动目标方法的多样性防御。载于全国移动目标研究研讨会论文集，2012年。[10] Michael B. Crouse 和 Errin W. Fulp。使用遗传算法实现计算机配置移动目标环境。载于 第*4*届配置分析与自动化研讨会（*SAFECONFIG*）论文集，第1–7页，2011年。[11] Minghui Zhu、Zhisheng Hu 和 Peng Liu。针对 Heartbleed 的自适应网络防御强化学习算法。载于 第*1*届*ACM*移动目标防御研讨会（*MTD*）论文集，第51–58页，2014年。[12] Guilin Cai、Baosheng Wang、Wei Hu 和 Tianzuo Wang。移动目标防御：最新进展与特点。《信息技术与电子工程前沿》，17：1122–1153，2016年。[13] Quanyan Zhu 和 Tamer Ba.sar。反馈驱动的多阶段移动目标防御的博弈论方法。载于第*4*届安全决策与博弈论会议（*GameSec*）论文集，第246–263页，2013年。[14] Richard Colbaugh 和 Kristin Glass。针对自适应对手的面向可预测性的防御。第2721–2727页，2012年10月。[15] Jafar Haadi Jafarian、Ehab Al-Shaer 和 Qi Duan。OpenFlow随机主机变异：利用软件定义网络实现透明移动目标防御。载于 第*1*届软件定义网络热点专题研讨会论文集，第127–132页，2012年。[16] Ehab Al-Shaer、Qi Duan 和 Jafar Haadi Jafarian。用于移动目标防御的随机主机变异。载于 第*8*届通信网络安全与隐私国际会议论文集，2012年。

[17] Yulong Zhang、Min Li、Kun Bai、Meng Yu 和 Wanyu Zang。针对云中虚拟机共置攻击的激励相容移动目标防御。载于第*27*届信息、安全与隐私会议（*SEC*）论文集，第388–399页，2012年。[18] Noam Ben-Asher、James Morris-King、Brian Thompson 和 William Glodek。攻击者技能、防御者策略以及基于迁移的移动目标防御在网络系统中的有效性。载于 第*11*届国际战争与安全国际会议（*ICCWS*）论文集，第21–30页，2016年。[19] Michael Thompson、Nathaniel Evans 和 Victoria Kisekka。多操作系统轮换环境：一种已实现的移动目标防御。载于 第*7*届国际弹性控制系统研讨会（*ISRCS*）论文集，第1–6页，2014年。[20] Mariusz Rawski、Slawomir Kukli´nski、Piotr Sapiecha、Marek Pelka、Grzegorz Przytuła、Przemysław Wojslaw 和 Krzysztof Szczypiorski。MMTD：面向企业网络的基于Mano的移动目标防御。载于 第*5*届世界计算与通信技术大会（*WCCCT*）论文集，第79–87页，2020年。[21] Nico Saputro、Samet Tonyali、Abdullah Aydeger、Kemal Akkaya、Mohammad A. Rahman 和 Selcuk Uluagac。物联网应用的移动目标防御机制综述，第563–614页。2020年。[22] Valentina Casola、Alessandra De Benedictis 和 Massimiliano Albanese。一种用于保护资源受限分布式设备的移动目标防御方法。第263卷，第22–29页，2013年8月。[23] Sridhar Venkatesan、Massimiliano Albanese、George Cybenko 和 Sushil Jajodia。一种用于瓦解隐蔽僵尸网络的移动目标防御方法。第37–46页，2016年10月。[24] Aleksandrs Slivkins。多臂老虎机简介。《机器学习基础与趋势®》，12(1-2)：1–286，2019年。[25] Gergely Neu 和 Gábor Bartók。无需重要性权重的重要性加权：一种高效的组合半老虎机算法。《机器学习研究期刊》，17：1–21，2016年。[26] Peter Auer、Nicolò Cesa-Bianchi、Yoav Freund 和 Robert E. Schapire。非随机多臂老虎机问题。《*SIAM*计算杂志》，32：48–77，2003年。[27] Haifeng Xu、Long Tran-Thanh 和 Nicholas R. Jennings。在没有先验知识的情况下进行重复安全博弈。载于 第*15*届自主智能体与多智能体系统国际会议（*AAMAS*）论文集，第104–112页，2016年。[28] Ofer Dekel、Ambuj Tewari 和 Raman Arora。针对自适应对手的在线老虎机学习：从遗憾到策略遗憾。载于 第*29*届国际机器学习会议（*ICML*）论文集，2012年。[29] James Hannan。重复博弈中对贝叶斯风险的近似。《博弈论研究成果》，3：97–140，1958年。[30] Adam Kalai 和 Santosh Vempala。在线决策问题的高效算法。《计算机与系统科学杂志》，71：291–307，2005年。[31] PaX团队。PaX地址空间布局随机化，2003年。[32] Richard D. McKelvey 和 Thomas R. Palfrey。标准型博弈的量化响应均衡。《博弈与经济行为》，10：6–38，1995年。[33] Rong Yang、Christopher Kiekintveld、Fernando Ordóñez、Milind Tambe 和 Richard John。改进针对人类对手的安全博弈资源分配策略：扩展研究。《人工智能》，195：440–469，2013年。[34] Thanh H. Nguyen、Rong Yang、Amos Azaria、Sarit Kraus 和 Milind Tambe。分析安全博弈中对手建模的有效性。载于 第*27*届*AAAI*人工智能大会论文集（*AAAI*），第718–724页，2013年。

符号表

符号	说明
θ	防御者
Ψ	攻击者及攻击者类型集合
ψ_i/ψ	攻击者类型 *i*/一种攻击者类型
τ	攻击者类型数量
P	攻击者类型的概率分布
C	可部署配置集合
c_i/c	配置 *i*/一个配置
V_c	配置 *c*的漏洞集
v_i/v	漏洞 *i*/一个漏洞
V	完整漏洞集
T	轮次/时间步总数
t	特定时间步
r_θ^t/r^t	时间步 *t*的防御者奖励函数
r_ψ^t	类型 ψ 在时间步 *t*的攻击者奖励函数
$\psi_{f(t)}$	在时间步 *t*发起攻击的攻击者类型
I_t	每轮结束时随奖励一同收到的附加信息集
s	切换成本函数
$D/(d_1, d_2, \ldots, d_T)$	防御者策略组合
d_t	在时间步 *t*部署的配置
$(a_1^{f(1)} \quad a_2^{f(2)} \ldots, \quad a_T^{f(t)})$	攻击者策略配置
$a_t^{f(t)}/a_t$	攻击者类型为 $\psi_{f(t)}$ 的攻击者在时间步 t 尝试进行漏洞利用
$TU(D)$	防御者策略 *D*的总效用
$\hat{r}^t$	时间 *t*的防御者奖励估计
$\mathbb{I}(.)$	指示函数：当且仅当其中的表达式为真时取值为一
γ, η	FPL-MTD 和 FPL-MaxMin 的超参数
b	预算
p_v	修复漏洞的价格 v

表 2：所用符号说明汇总

B 第 4 节的缺失结果

命题1。 几何重采样过程至少以 $1-e^{-T}$ 的概率终止，其中 $M \geq \binom{|C|T}{\gamma}$，这里 $|C|$ 是可部署配置的数量，T 是时间步总数，γ 是 FPL-MTD （算法*1*）的探索参数

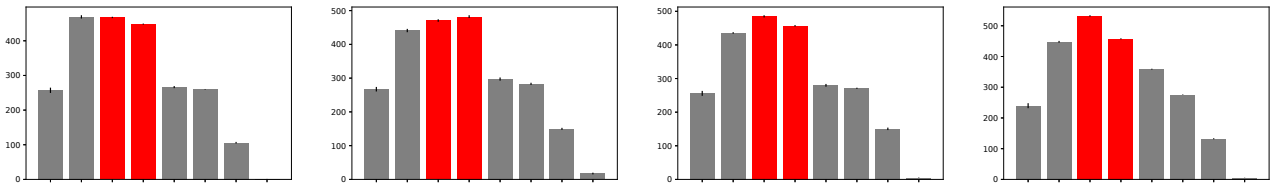
证明。 在每个时间步，算法1部署任意一个配置的概率至少为 $\frac{\gamma}{|C|}$，因为它以 $\tilde{}$ 的概率进行探索。几何重采样过程会在采样的配置 d 与该轮实际部署的配置 d_t 相同时终止。在任意给定轮次中，该事件发生的概率下界为 $\frac{\gamma}{|C|^\circ}$ 经过 $M \geq \binom{|C|T}{\gamma}$ 次迭代后，该事件仍未发生的概率上界为

$$\left(1-\frac{\gamma}{|C|}\right)^M \leq \left(1-\frac{\gamma}{|C|}\right)^{\frac{|C|T}{\gamma}} \leq e^{-\frac{\gamma}{|C|}\frac{|C|T}{\gamma}} \leq e^{-T}$$

第二个不等式源于以下事实：对于任意实数值 x，$e^{-x} \geq 1-x$。代入 $x = \frac{\gamma}{|C|}$ 可得第三个表达式。因此，终止概率至少为 $1-e^{-T}$。 □

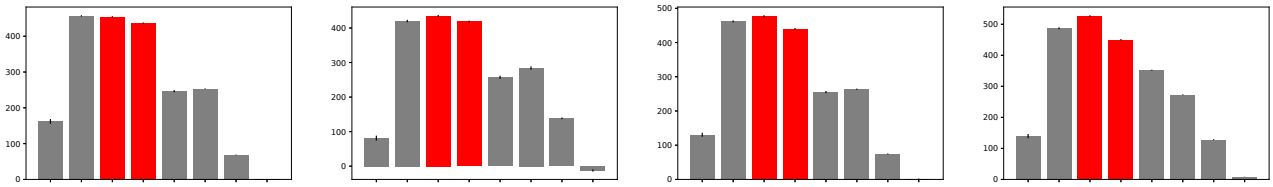
C 补充实验结果

我们还在合成生成的一般和数据集和零和数据集上评估了我们的算法。数据集生成所用的流程见附录E.1.2。



(a) 最佳响应　(b) FPL-UE　(c) 斯塔克尔伯格　(d) 随机

图 3：算法在一般和合成数据上的性能。每张图中从左到右依次为：BSS-Q、S-OPT、FPLMaxMin、FPLMTD、RobustRL、S-Exp3、FPL+GR



(a) 最佳响应　(b) FPL-UE　(c) 斯塔克尔伯格　(d) 随机

图 4：算法在零和合成数据上的性能。每张图中从左到右依次为：BSS-Q、S-OPT、FPLMaxMin、FPLMTD、RobustRL、S-Exp3、FPL+GR

我们生成了 20 个此类一般和数据集和零和数据集。对于每个数据集，我们将每种算法运行 10 次，每次运行 1000 个时间步，并测量平均性能。图 3 和图 4 展示了所有 20 个数据集上的平均性能及其标准误差界。

趋势与大型基于 NVD 的数据集相同。有关这些趋势的讨论，请参阅第 5.3 节。

D 关键漏洞识别

除了提供良好的切换策略以促进移动目标防御外，*FPL-MaxMin* 使用的奖励估计还可用于帮助防御者在资源不足以修复其配置中的所有漏洞时，决定修复哪些漏洞。更正式地说，本节研究以下问题：给定预算 b 和价格向量 $\vec{p}$，其中 p_v 表示修复漏洞 $v \in V$的价格，防御者应修复哪些关键漏洞，才能提升*FPL-MaxMin*未来的性能？请注意，*FPL-MTD* 无法做到这一点，因为它不会为每个漏洞维护奖励估计。尽管*FPL-MTD*的奖励估计可用于在决定重点关注哪些配置时提供指引，但不能用于识别关键漏洞。

令 $r_{v,\psi}$ 表示*FPL-MaxMin* 算法针对每个漏洞－攻击者类型对 (v, ψ) 得到的奖励估计。给定预算 b 和价格向量 $\vec{p}$，选择关键漏洞的一种方法是遵循*FPL-MaxMin* 的配置选择方法，选择漏洞以最大化由式（7）给出的防御者加权最大最小值。更正式地说，令 $S \subseteq V$为选定进行修复的漏洞集合。令 r^S 表示奖励估计，其中漏洞 S 已修复，防御者不再从该漏洞获得任何负奖励，即

$$r^S_{v,\psi} = \begin{cases} r_{v,\psi} & v \notin S \\ 0 & v \in S \end{cases}$$

上述方法的目标是选择一个集合 S，使其在满足预算约束的条件下，最大化博弈的加权最大最小值。这可以表示为以下优化问题：

$$\max_{S \subseteq V} \max_{c \in C} \left(\sum_{\psi \in \Psi} P_\psi \min_{v \in V_c} r^S_{v,\psi} \right)$$
$$\text{s.t.} \sum_{v \in S} p_v \leq b \tag{8}$$

其中， V_c 是配置 c的漏洞集合。
很容易看出，使上述优化问题达到最大值的集合 S 会极度偏向于已有较高最坏情况奖励的配置；这种方法会选择最坏情况奖励最高的配置，并将全部预算用于修复该配置中的漏洞。示例 D.1 更清楚地说明了这一点。

示例 D.1. 考虑这样一种情形：有 1 种攻击者类型、2 种配置 $\{c_1, c_2\}$ 和 3 个漏洞， $V = \{v_1, v_2, v_3\}$。设 c_1 有漏洞 $\{v_1, v_2\}$，而 c_2 有漏洞 $\{v_2, v_3\}$。奖励估计向量为 $r = (-1, -0.5, -0.1)$。由此可知， c_1 的最差漏洞的奖励为 -1，而 c_2 的最差漏洞的奖励为-0.5 。假设预算只够修复一个漏洞，且三个漏洞中的任意一个都可以修复。

很容易看出，(8) 会选择修复 v_2，将 c_2 的最差漏洞的奖励提高到 -0.1，同时使 c_1 的最差漏洞的奖励保持在 -1。

有效移动目标防御要求各配置所包含的漏洞数量大致相当。否则，当某个配置明显优于其他配置时，其他配置就不会经常部署，因而会被浪费。因此，我们需要调整方法，使各个配置得到均衡的发展。我们可以将对配置集合取最大值改为取最小值。直观而言，这会最大化最差配置中最严重的漏洞；这种方法会将预算分配到各个配置上，因为随着我们修复其中一些配置的漏洞、使其变得更好，“最差”的配置会不断变化。

更正式地说，我们有一个新的优化问题，称为*CHOOSE-VUL*:

$$\max_{S \subseteq V} \min_{c \in C} \left(\sum_{\psi \in \Psi} P_\psi \min_{v \in V_c} r^S_{v,\psi} \right)$$
$$\text{s.t} \sum_{v \in S} p_v \leq b \tag{9}$$

尽管上述形式化确实刻画了我们要解决的问题，但遗憾的是，即使只有一个配置，该问题也是 NP 困难的。

定理 D.1. 即使只有一个配置，求解 *CHOOSE-VUL* 的最优解也是 *NP* 困难的。

证明。 我们通过归约到 NP 完全问题（0-1）背包问题来证明这一点:

> 给定 n 个物品 $N = [n]$，每个物品的值 val_i 和重量 w_i均为非负数，以及两个非负数 W 和 K，是否存在一个子集 $S \subseteq N$，使得 S 的总重量小于W，且 S 的总值至少为 K?

给定一个 KNAPSACK 实例，按如下方式构造一个*CHOOSE-VUL* 实例：设只有一个配置，其中包含一组漏洞 n $\{v_1, v_2, \ldots, v_n\}$和一组攻击者类型 n $\{\psi_1, \psi_2, \ldots, \psi_n\}$，且每种类型发动攻击的概率相同。奖励估计构造如下:

$$r_{v_i,\psi_j} = \begin{cases} -val_i & i = j \\ 0 & i \neq j \end{cases}$$

直观地说，这相当于每种攻击者类型只能利用一个漏洞，且任意两种类型都不能利用同一个漏洞的情形。最后，令价格向量 $\vec{p}$等于重量向量 $\vec{w}$，预算 b 等于W。在这一实例化表述下，由式（9）给出的*CHOOSE-VUL* 可化简为

$$\max_{S \subseteq V} \left(\sum_{i \in [n]} \frac{1}{n} r^S_{v_i,\psi_i} \right)$$
$$\text{s.t} \sum_{v \in S} w_v \leq W \tag{10}$$

这是因为，假设只有一种配置，我们就可以去掉 $\min_{c \in C}$ 项。我们还可以将 $\min_{v \in V_c}$替换为攻击者类型 ψ_i 能够利用的唯一漏洞。

现在，我们用二进制向量 $\vec{x}$ 表示集合 S，其中 $x_i = 1$ 当且仅当 $i \in S$。我们可以将上述优化问题重写为

$$\max_{\vec{x}} \left(\sum_{i \in [n]} \frac{-val_i + val_i x_i}{n} \right)$$
$$\text{s.t} \sum_{i \in [n]} w_i x_i \leq W$$
$$x_i \in \{0, 1\}$$

去除目标函数中的所有常数项后，上述问题等价于以下问题

$$\max_{\vec{x}} \left(\sum_{i \in [n]} val_i x_i \right)$$
$$\text{s.t} \sum_{i \in [n]} w_i x_i \leq W$$
$$x_i \in \{0, 1\} \tag{11}$$

注意，这等价于背包问题的优化版本。容易看出，当且仅当原始背包问题实例的解为 YES 时，式 (11) 中的目标函数值至少为 K。此外，利用式 (10) 的解，可以在多项式时间内计算出式 (11) 的解。至此，归约完成。 □

上述归约要求存在多种攻击者类型。如果将问题空间限制为仅包含一种攻击者类型的实例，则可以用贪心算法在多项式时间内求解 *CHOOSE-VUL* 。算法的步骤已在算法 4 中描述。算法从空集开始，并在每次迭代中加入奖励估计值最低的漏洞。算法持续执行，直至再加入任何项目都会违反预算约束。

算法 4 单一攻击者类型下用于漏洞选择的贪心算法

$S \leftarrow \emptyset$**当1 执行**$v' = \arg\min_{v \in V} r^S_{v,\ \psi}$ ▷ 如果有多个，则选择价格最低的那个
如果 $\sum_{v \in S} p_v + p_{v'} \leq b$ **则** $S \leftarrow S \cup \{v'\}$ **否则跳出循环 结束如果 结束循环 返回** S

定理 D.2. 算　法 *4* 在只有一种攻击者类型时，返回 *CHOOSE-VUL* 的最优解。

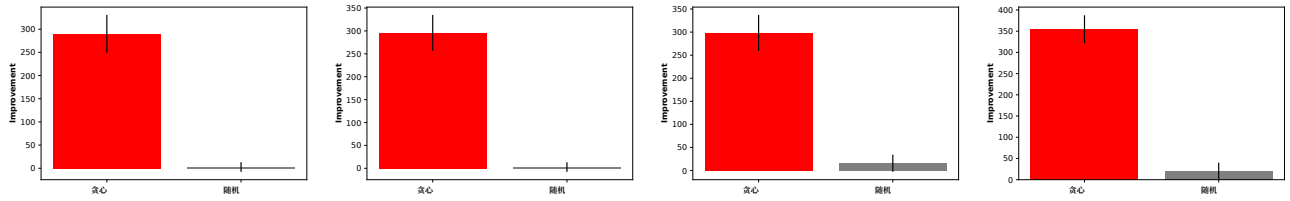
证明。当只有　一种攻击者类型（设为 ψ）时，*CHOOSE-VUL* 可化简为以下优化问题　:

$$\max_{S \subseteq V} \left(\min_{v \in V} r^S_{v,\psi} \right)$$
$$\text{s.t} \sum_{v \in S} p_v \leq b \tag{12}$$

这是因为，当只有一种攻击者类型时，我们可以将两个最小值函数合并，因为它们之间不会有概率项。具体来说，我们有 $\min_{c \in C} \min_{v \in V_c} = \min_{v \in V}$。

很容易看出，对于这类问题，最优解会在满足预算约束的前提下选取最差的漏洞。反设事实并非如此。令 S_g 为非最优的贪心解， S_{OPT} 为最优解。令贪心解的最终目标值为 O_g，最优解的最终目标值为 O_{OPT}。假设 $O_g < O_{OPT}$。

假设存在某个漏洞 $v \in S_g \setminus S_{OPT}$，其奖励估计为 r_v。注意，根据定义，贪心算法会选取所有奖励估计更差的漏洞，因此， $O_g \geq r_v$。另一方面，最优解的目标函数上界为 r_v。由此可得 $O_g \geq O_{OPT}$，这就产生了矛盾。



(a) 预算 $= 2$ (b) 预算 $= 5$ (c) 预算 $= 10$ (d) 预算 $= 15$

图 5：漏洞选择算法在小型基于 NVD 的数据集上的改进。左侧为贪心方法，右侧为随机方法。

因此，我们必须有 $S_g \subseteq S_{OPT}$。注意， S_g 已终止，因为由于预算约束，它无法将价格最低且奖励估计最小的漏洞加入集合。我们将这个漏洞称为 v，并用 r_v 表示其奖励估计。根据定义，我们有 $O_g = r_v$。此外，我们还有 $O_{OPT} = r_v$，因为 OPT 无法加入 v；但由于它是 S_g 的超集，它会加入所有比 v 更差的漏洞。由此可得 $O_g = O_{OPT}$，这又产生了矛盾。

因此， 贪心算法必须返回最优解 。 □

算法 5 适用于多种攻击者类型的漏洞选择贪心算法

$S \leftarrow \emptyset$**当** 1 **时执行** $S' =$ 取最大值$^{S:|S|=1}$ 最小值$_{c \in C}$ $\left(\sum_{\psi \in \Psi} P_\psi$ 最小值$_{v \in V_c} r_v^{S \cup S'}$, $_\psi \right)$ ▷ 如果有多个，则选择价格最低的那个**如果** $\sum_{v \in S(\cup S')} p_v \leq b$ **则** $S \leftarrow S \cup S'$ **否则跳出循环 如果结束 循环结束 返回** S

对于具有多种攻击者类型的更复杂情形，我们建议采用类似的贪心算法（见算法 5）。在每次迭代中，算法会考察所有可能的单元素漏洞集合，并选择使目标函数最大化的集合。当违反预算约束时，算法便停止。只有一种攻击者类型时，该算法就会退化为算法 4。虽然我们无法为该算法提供近似保证，但简单的模拟表明，只需修复少量漏洞，贪心方法就能显著提高*FPL-MaxMin*的总效用。

具体来说，我们在小型基于 NVD 的数据集（第 5.2 节）上，针对最佳响应攻击者测试了我们的方法。我们假设所有漏洞的单位价格相同，并通过遍历集合 $\{2, 5, 10, 15\}$ 中的所有预算值来改变总预算。请注意，当预算为 15 时，任何算法都只能选择约 5% 的漏洞进行修复。我们将我们的方法与随机方法进行了比较；随机方法会随机选择漏洞进行修复，直到超出预算为止。

为了比较这两种方法，我们首先运行了 *FPL-MaxMin* 1000 个时间步，然后创建了三个分支：一个使用贪心算法（算法 5）修复漏洞，一个使用随机方法修复漏洞，另一个则不修复任何漏洞。对于这三个分支，我们继续使用同一个*FPL-MaxMin* 实例再运行 1000 个时间步。我们进行了 50 次实验，并记录了总效用。

我们将改进定义为修复漏洞后算法的总效用与未修复漏洞时算法的总效用之差。我们绘制了各漏洞修复方法的平均改进及其标准误差界，如图 5 所示。

结果表明，贪心方法的表现显著优于随机方法；当预算为 5 时，其效果提升超过随机方法的 100 倍。我们还发现，当预算为 5 时，修复漏洞后的总效用平均降至未修复任何漏洞时总效用的 57.1%；这是好事，因为总效用始终为负。这表明，通过智能地选择要修复的漏洞，我们可以取得显著改善。

虽然 *CHOOSE-VUL* 是我们采用的问题形式化方法，但它只是选择漏洞时可最大化的众多目标之一。可以说，它并非在所有应用场景中都是最优的；但对于这个问题而言，何为最优本身就存在争议。我们将把对该问题及其各种方法的更深入研究留待未来开展。

E 实验的技术细节

E.1 数据集生成

E.1.1 大型 NVD 数据的生成

国家漏洞数据库（NVD）是一个公开目录，其中包含可能影响系统组件的通用漏洞与披露（CVE）信息。该目录每年都会更新一个新文件。对于每个漏洞，他们使用一种称为通用漏洞评分系统（CVSSv3）的评分系统，对漏洞的不同方面进行评分。在众多评分中，两个重要的评分是影响评分（IS），用于衡量成功进行漏洞利用所造成的影响；以及基础评分（BS），它综合了影响评分和漏洞的可利用难易程度。 [1] 使用这些评分在贝叶斯博弈中生成奖励，具体如下：对于每个漏洞 $v \in V$、每种攻击者类型 $\psi \in \Psi$ 以及每种配置 $c \in C$，他们都会生成一个奖励函数

$$r_\theta(\psi, v, c) = \begin{cases} -IS_v & v \in V_c \text{ and the attacker type } \psi \text{ can exploit } v \\ 0 & \text{Otherwise} \end{cases}$$
$$r_\psi(v, c) = \begin{cases} BS_v & v \in V_c \text{ and the attacker type } \psi \text{ can exploit } v \\ 0 & \text{Otherwise} \end{cases} \tag{13}$$

其中，IS_v 和 BS_v 分别是 v 的影响评分和基础评分。

我们使用相同的方法将漏洞评分转换为奖励；唯一的区别是，我们会缩小奖励，以确保其范围为 $[0, 1]$。我们从 2002 年至 2021 年的通用漏洞与披露（ CVE ）数据中提取数据，收集了一个包含 87180 个漏洞及其影响评分和基础评分的数据库。我们将这组漏洞称为全局漏洞集。

对于每个数据集，我们在 10 到 20 之间均匀随机选择配置数量，在 3 到 6 之间均匀随机选择攻击者数量，并在 500 到 800 之间均匀随机选择漏洞数量。我们从全局漏洞集中均匀随机抽取漏洞，组成每个数据集。

对于每种攻击者类型，我们从范围为 $[0, 1]$ 的截断正态分布中抽取一个技能水平。随后，我们使用均值与其技能水平成正比的截断正态分布，确定该攻击者能够利用的漏洞数量，并从中均匀随机选择这些漏洞。对于每个配置，我们生成一个漏洞集，使得每个漏洞以 0.01 的概率不是该配置的漏洞。

利用每个漏洞的基础评分和影响评分、每个配置的漏洞集，以及每个攻击者能够利用的漏洞集合，我们根据 (13) 为防御者和攻击者生成奖励函数。

E.1.2 合成通用数据和零和数据的生成

对于每个数据集，我们在 10 到 20 之间均匀随机选择配置数量，在 3 到 6 之间均匀随机选择攻击者数量，并在 500 到 800 之间均匀随机选择漏洞数量。

对于每种攻击者类型，我们从范围为 $[0, 1]$ 的截断正态分布中抽取一个技能水平。随后，我们使用均值与其技能水平成正比的截断正态分布，确定该攻击者能够利用的漏洞数量，并从中均匀随机选择这些漏洞。对于每个配置，我们生成一个漏洞集，使得每个漏洞以概率 0.05 不属于该配置。

对于每个漏洞，我们从 $U[-1, 0]$ 中抽取防御者奖励，并从 $U[0, 1]$ 中抽取攻击者奖励。总之，我们按如下方式创建奖励函数:

$$r_\theta(\psi, v, c) = \begin{cases} U[-1, 0] & v \in V_c \text{ and the attacker type } \psi \text{ can exploit } v \\ 0 & \text{Otherwise} \end{cases}$$
$$r_\psi(v, c) = \begin{cases} U[0, 1] & v \in V_c \text{ and the attacker type } \psi \text{ can exploit } v \\ 0 & \text{Otherwise} \end{cases}$$

nvd.nist.gov

参数	值
Seed	2022
攻击者类型数量	[3, 6]
漏洞数量	[500, 800]
配置数量	[10, 20]
T	1000
γ_{MaxMin}	0.006
η_{MaxMin}	0.03
γ_{MTD}	0.007
η_{MTD}	0.1
α_{BSSQ}	0.2
折扣因子 (γ_{BSSQ})	0.8
$\epsilon_{RobustRL}$	0.1

表 3：参数值

对于零和数据集，我们 仅对防御者奖励进行采样，并将攻击者奖励视为该防御者奖励的加法逆元。

E.2 计算资源

所有混合整数二次规划均使用 Gurobi 求解。所有实验均在配备 128GB 随机存取存储器的英特尔至强 E5-2680 v4 处理器上运行。

E.3 可复现性

为确保代码的可复现性，我们在所有程序中均使用随机种子值 2022 来初始化伪随机数生成器。所有使用随机化的函数都使用同一个生成器。代码中包含了所有使用的数据集以及超参数的值；这些值也已列于表 3 中。