Autonomous Agents and Multi-Agent Systems (2021) 35:36
https://doi.org/10.1007/s10458-021-09522-w

Revenue maximizing markets for zero-day exploits

Mingyu Guo[1] · Guanhua Wang[1] · Hideaki Hata[2] · Muhammad Ali Babar[1]

Accepted: 30 June 2021 / Published online: 7 July 2021

Abstract
Markets for zero-day exploits (software vulnerabilities unknown to the software vendor) have a long history and a growing popularity. We study these markets from a revenue-maximizing mechanism design perspective. We first propose a theoretical model for zero-day exploits markets. In our model, one exploit is being sold to multiple buyers. There are two kinds of buyers, which we call the defenders and the offenders. The defenders are buyers who buy vulnerabilities in order to fix them (*e.g.*, software vendors). The offenders, on the other hand, are buyers who intend to utilize the exploits (*e.g.*, national security agencies and police). We study the problem of selling one zero-day exploit to multiple defenders and offenders. Our model has a few unique features that make it different from single-item auctions. First, an exploit is a piece of information, so one exploit can be sold to multiple buyers. Second, buyers have externalities. If any defender wins, then the exploit becomes worthless to the offenders. Third, if the auctioneer discloses the details of the exploit to the buyers before the auction, then they may leave with the information without paying. On the other hand, if the auctioneer does not disclose enough details, then the buyers cannot determine how valuable the exploit is. Considering the above, our proposed mechanism discloses the details of the exploit to all offenders at the beginning of the auction. The defenders will receive the information slightly delayed. The offenders bid to prolong the delay and the defenders bid to shorten the delay. We derive the optimal mechanism for single-parameter valuations. For general valuations, we propose three numerical solution techniques. One is based on iterative linear programming and the other two are based on neural networks and evolutionary computation.

Keywords Revenue maximization · Bug bounty · Automated mechanism design · Mechanism design via neural networks · Mechanism design via evolutionary computation

✉ Mingyu Guo
mingyu.guo@adelaide.edu.au

Guanhua Wang
guanhua.wang@adelaide.edu.au

Hideaki Hata
hata@shinshu-u.ac.jp

Muhammad Ali Babar
ali.babar@adelaide.edu.au

[1] School of Computer Science, University of Adelaide, Adelaide, Australia

[2] Faculty of Engineering, Shinshu University, Nagano, Japan

1 Introduction

1.1 Background: zero-day exploit markets

A zero-day exploit refers to a software vulnerability that has not been disclosed to the public, and is unknown to the software vendor. Information of new vulnerabilities gives cyber attackers free passes to attacking targets, while the vulnerabilities remain undetected. The trading of zero-day exploits has a long history, and selling them by security researchers as a "legitimate source of income" is accepted by the security community [6].

Zero-day exploit markets are not necessarily black markets where the buyers are potential cyber criminals. It is common for software vendors to buy exploits via in-house or community-run bug bounty reward programs. National security agencies and police also buy exploits. It is widely reported that government agencies utilize zero-day exploits to track criminals or for other national security reasons. Financial industries buy exploits to prevent attacks (these companies are willing to pay significant amounts to buy information regarding any newly discovered attack methods, as such information helps implementing counter measures). There are legitimate venture capital backed security companies whose business model is to sell exploits *for profit*. For example, ZeroDium is a zero-day acquisition firm, who buys high-risk vulnerabilities with premium rewards, then resells them to mostly government clients [8]. Another similar company is Vupen, who offers a subscription service for its clients, providing vulnerability data and exploits for zero days and other bugs [8].

Greenberg presented a price list of zero-day exploit sale, ranging from $5000–$30,000 to $100,000–$250,000 [9]. It is widely reported that both Apple and ZeroDium offered (at least) 1 million USD reward for iOS exploits. These prices are so high because it is generally difficult for the software vendor to independently discover these vulnerabilities [2], hence the exploits are expected to be alive for long periods of time.

1.2 A summary of results

In this paper, we study markets for zero-day exploits from a revenue-maximizing mechanism design perspective. We summarize all results of this paper as follows:

- We propose a theoretical mechanism design model for zero-day exploit markets. We identify the unique features of such markets. First, an exploit is a piece of information, so one exploit can be sold to multiple buyers. Second, buyers have externalities. We divide the buyers into two categories: the defenders (*e.g.*, software vendors) and the offenders (*e.g.*, national security agencies and police). Once a defender "wins" an exploit, the exploit becomes worthless for the offenders. Third, if the auctioneer discloses the details of the exploit to the buyers before the auction, then they may leave with the information without paying. On the other hand, if the auctioneer does not disclose the details, then it is difficult for the buyers to come up with their private valuations.
- We propose the *straight-forward (SF)* mechanism property, which requires that the mechanism discloses the full details of the exploit to the offenders before they submit their bids. In Proposition 1, we show that for the purpose of designing revenue-maximizing mechanisms, if *strategy–proofness (SP)*, *individual rationality (IR)*, and

straight-forwardness (SF) are required, then it is without loss of generality to focus on mechanisms that works as follows. In the beginning of the auction, the mechanism discloses the full details of the exploit to all offenders, and then based on the bids from both the offenders and the defenders, the mechanism specifies a slightly delayed time to reveal the information to the defenders, which we call the *ending time* (for the exploit), because once the exploit is revealed to the defenders, the exploit becomes worthless to the offenders. The offenders bid to keep the exploit alive, while the defenders bid to end the exploit earlier.

In our model, the outcome space is a continuous time interval and the outcome (the ending time) is a point within the interval. Our model is similar to the *cake-cutting* problem [3, 4]. For example, the interval before the ending time is the part enjoyed by the offenders, and the interval after the ending time is the part enjoyed by the defenders.

- For a simplified single-parameter model, we modify and apply Myerson's classic technique for designing optimal single-item auction [15]. Our technique involves one extra optimization step on top of Myerson's original approach. Instead of choosing an outcome based on which agent has the highest virtual valuation, we pick the ending time that maximizes the agents' combined virtual valuations. Under our model, this outcome selection rule still guarantees the monotonicity constraint required by Myerson's technique. As a result, we are able to derive an optimal mechanism that maximizes the expected revenue for the single-parameter model.
- For the general model, we adopt the computationally feasible automated mechanism design approach [10]. Instead of optimizing over all mechanisms, we focus on a family of parameterized mechanisms, and tune the parameters in order to obtain a good mechanism. We focus on the Affine Maximizer Auctions (AMA) mechanisms used for designing revenue-maximizing combinatorial auctions [18]. To identify a good AMA mechanism, we propose three numerical techniques. One is based on iterative linear programming and the other two are based on neural networks and evolutionary computation. Experimentally we show that all techniques can solve for near optimal mechanisms. For theoretical guarantees, we show that if there are only two agents (one offender and one defender), and if the defender's valuation is much lower than the offender's valuation (typically true for zero-day exploit markets), then the optimal AMA mechanism is near optimal among all mechanisms, which implies that focusing on the AMA mechanisms is justified (under the above conditions).

1.3 Practical benefits of the proposed markets

- *Benefits for offenders* On existing zero-day exploit trading platforms such as ZeroDium and Vupen, buyers are not allowed to view the details of the exploits before committing to buy. Our mechanisms satisfy the *straight-forward* property: our mechanisms disclose the full details of the exploit to the offenders before they submit their bids.
- *Benefits for defenders* Under existing bug bounty programs (markets where the defenders purchase zero-day exploits), pricing of zero-day exploits is difficult. It is up to the defenders to assign prices. If the prices are too low, then the bug bounty program cannot attract bug submissions. Under our mechanisms, prices are determined by the market. Furthermore, when there are multiple defenders, our mechanism splits the payment among the defenders in a way that those with high valuations for the exploit pay more.

– Benefits for the auctioneer: For single-parameter settings, our proposed mechanism has the optimal revenue. For other settings, experimentally, we show that our proposed mechanisms have near-optimal revenue.

2 Model description

In this section, we propose a mechanism design model for zero-day exploit markets. Our aim is to create a model with minimal assumptions, and draw a parallel between our model and existing classic mechanism design models.

There is one exploit being sold to multiple game-theoretically strategic buyers. The seller is the mechanism designer, who wants to maximize her revenue. For example, the seller is a cyber security company that sells exploits for profit.[1]

Assumption 1 One exploit is sold over a time frame [0, 1]. 0 represents the moment the exploit is ready for sale. 1 represents the exploit's end of life (*e.g.*, it could be the end of life of the affected software, or the release of a major service pack).

Assumption 2 The buyers consist of *defenders* and *offenders*.

– A defender is a buyer who buys exploits in order to fix them. Given a specific exploit, usually there is only one defender. For example, suppose the exploit attacks the Chrome browser, then Google is the defender, who would, for example, buy the exploit via a bug bounty reward program. Our model allows multiple defenders, but we assume that as soon as any defender gets hold of an exploit, the exploit gets immediately fixed. That is, if t is the earliest moment any defender receives the information about the exploit, then the exploit becomes worthless to all offenders from time t, and all defenders can enjoy a "protected" time interval from time t to 1.
– An offender is a buyer who intends to utilize the exploit.

Let $t_i \in [0, 1]$ be the moment the exploit is disclosed to buyer i. The above assumption basically says that it is without loss of generality to assume that all defenders receive the information at the same time, denoted by t_{end}, where t_{end} represents the earliest moment any defender receives the exploit information. In other words, t_{end} is the ending time of the exploit.

It is also without loss of generality to assume that if i is an offender, then $t_i \in [0, t_{end}]$, because for an offender, receiving the information after t_{end} is equivalent to receiving it exactly at t_{end}.[2]

Buyer i's type is characterized by a nonnegative function $v_i(t)$, where $v_i(t)$ is i's instantaneous valuation at time t. If i is an offender who receives the exploit at t_i and the exploit gets fixed at t_{end}, then i's valuation equals

$$\int_{t_i}^{t_{end}} v_i(t)dt$$

[1] Example such companies include ZeroDium and Vupen [8].
[2] Both mean the same thing – this offender does not get to use the exploit.

Similarly, if i is a defender, then her valuation is determined by the time interval where she is protected, which equals

$$\int_{t_{end}}^{1} v_i(t)dt$$

Basically, the offenders wish to keep the exploit alive for as long as possible (they prefer a larger t_{end}), while the defenders wish to fix the exploit as early as possible (they prefer a smaller t_{end}).

A mechanism takes as input the valuation functions $v_i(t)$ for $i = 1, 2, \ldots, n$, and produces an outcome $(t_1, t_2, \ldots, t_n)$ and a payment vector $(p_1, p_2, \ldots, p_n)$, where p_i is the amount i pays under the mechanism. A buyer's utility equals her valuation minus her payment. We focus on mechanisms that are strategy-proof and individually rational.

Definition 1 *Strategy-proofness (SP)*: For every buyer i, by reporting $v_i(t)$ truthfully, her utility is maximized.

Definition 2 *Individually rationality (IR)*: For every buyer i, by reporting $v_i(t)$ truthfully, her utility is nonnegative.

Besides SP and IR, we introduce another mechanism property specifically for zero-day exploit markets. One thing we have been ignoring is that an exploit is a piece of information. As a result, if the auctioneer discloses the exploit's details to the buyers before the auction, then they may simply walk away with the information for free. If the auctioneer does not describe what it is being sold, then it is difficult for the buyers to come up with their valuation functions.

Assumption 3 We assume that there are two ways for the seller to describe an exploit: either describe the full details, or describe what can be achieved with the exploit (*e.g.*, with this exploit, anyone can seize full control of a Windows 10 system remotely).

- We assume that it is safe for the seller to disclose what can be achieved with the exploit. That is, the buyers will not be able to derive "how it is done" based on "what can be achieved".
- If the seller only discloses what can be achieved, then it is difficult for an offender to determine whether the exploit is new, or something she already knows. (For example, a national security agency may already possess an arsenal of many exploits.) It is therefore difficult for the offenders to come up with their valuation functions in this kind of situation.[3]
- We assume that the defenders are able to come up with valuation functions just based on what can be achieved. This is because all zero-day exploits are by definition unknown to the defenders. This assumption is consistent with practise. For bug bounty programs, vulnerabilities are generally classified into different levels of severities, and vendors pay depending on these classification [1]. For example, Google Chrome provides guidelines to classify vulnerabilities into critical, high,

[3] They may come up with expected valuation functions by estimating how likely the exploit is new, but this may then lead to regret after the auction.

medium, and low severities, and pay accordingly [17]. Google pays not based on "how it is done", but based on "what can be achieved".

The above assumption leads to the following mechanism property:

Definition 3 *Straight-forwardness (SF)*: A mechanism is straight-forward if the mechanism reveals the full details of the exploit to the offenders, before asking for their valuation functions.

It should be noted that SF does not require that the exploit details be revealed to the defenders before they bid. If the seller does this, then the defenders can simply fix the exploit and bid $v_i(t) \equiv 0$. Due to IR, the defenders can get away without paying.

Offenders are given the details before they bid, but they cannot simply bid $v_i(t) \equiv 0$ to get away without paying. Our mechanism's key idea is to use the defenders as "threat". That is, if the offenders bid too low, then the auctioneer discloses the exploit to the defenders earlier, which renders the exploit worthless. Essentially, the offenders need to bid/pay to keep the exploit alive. The higher they bid/pay, the longer the exploit remains alive.

From now on, we focus on mechanisms that are SP, SF, and IR. We prove the following characterization result:

Proposition 1 *Let M be a mechanism that is strategy-proof, individually rational, and straight-forward.*

We can easily construct M' based on M, so that M' is also strategy-proof, individually rational, and straight-forward. M' and M have the same revenue for all type profiles. M' has the following form:

- At time 0, the seller reveals the exploit in full details to all offenders, and reveals what can be achieved with the exploit to all defenders.
- Collect valuation functions from the buyers.
- Pick an outcome and a payment vector based on the bids. The outcome is characterized by a single value t_{end}, which is the ending time for the exploit. All offenders have $t_i = 0$ and all defenders have $t_i = t_{end}$.

The above proposition implies that for our goal of designing revenue-maximizing mechanisms, it is without loss of generality to focus on mechanisms with the above form.

Proof Given M, we modify it and construct M' as follows: for each i that is an offender, we move t_i to 0. For each i that is a defender, we do not change t_i. For every type profile, we keep M's payment vector. That is, M' has the same revenue for every type profile. M' is by definition SF.

Now we show M' is still SP and IR. It is easy to see that the defenders' valuations are not changed, so M' is still SP and IR for the defenders. Offenders' valuations are changed. For offender i, originally under M, she receives the information at time t_i. Under M', she receives the information at time 0. It should be noted that because M is SF, that means t_i

is not dependent on i's own report. Therefore, the valuation increase for i, which equals $\int_0^{t_i} v_i(t)dt$, is independent of i's own report. Hence, this increase of valuation does not change i's strategy. M' is still SP and IR for the offenders as well. □

3 Comparing against classic models

To summarize our model, there are two types of agents (offenders and defenders). Agent i's type is characterized by her valuation function $v_i(t)$. The outcome $t_{end}(v_1, v_2, \dots, v_n) \in [0, 1]$ is chosen based on the type profile. The exploit is active between $[0, t_{end}]$, during which period all offenders can utilize the exploit. The exploit becomes worthless from t_{end}. High bids (high valuation functions) from the offenders would push t_{end} toward 1, while high bids from the defenders would push t_{end} toward 0.

Our model is very similar to both the *cake-cutting* problem and (to a much less extent) the *single facility location* problem.

Cake-cutting: The time frame [0, 1] can be viewed as the cake. t_{end} cuts the cake into two halves. There are a few differences between our model and cake-cutting. For one thing, our model is more like *group* cake cutting, as both sides involve multiple agents. Secondly, the offenders are bound to the left-hand side ($[0, t_{end}]$) while the defenders are bound to the right-hand side ($[t_{end}, 1]$).

Single facility location: t_{end} can also be viewed as the position of the facility in a single facility location problem. The defenders are all positioned at 0, so they prefer t_{end} to be closer to 0 (which enlarges the interval $[t_{end}, 1]$). The offenders are all positioned at 1, so they prefer t_{end} to be closer to 1 (which enlarges the interval $[0, t_{end}]$). Our model is different from single facility location because the agents' locations are only one of the two ends.

Furthermore, most cake-cutting and facility location literatures focus on money-free settings, so previous results do not apply to our model.

4 Optimal single-parameter mechanism

In this section, we study a simplified single-parameter model, and derive an optimal mechanism that maximizes the expected revenue. Results in this section are based on Myerson's technique on optimal single item auction [15], with one extra optimization step introduced to fit our model.

Assumption 4 Single-parameter model: Agent i's valuation function $v_i(t)$ is characterized by a single parameter $\theta_i \in [0, \infty)$:

$$v_i(t) = \theta_i c_i(t)$$

Here, $c_i(t)$ is a publicly known nonnegative function. That is, i's type is characterized by a single parameter θ_i.

For example, consider an offender i, if $c_i(t)$ represents the number of software users i may attack using the exploit at time t, and θ_i is agent i's valuation for attacking one user, then we have $v_i(t) = \theta_i c_i(t)$. Similarly, for defenders, it'd be *saving* instead of attacking.

For the single-parameter model, $t_{end}(\theta_1, \theta_2, \ldots, \theta_n)$ determines the outcome (ending time). For presentation purposes, fixing θ_{-i} (the other agents' types) and drop it from the notation, when agent i reports θ_i, we denote the outcome by $t_{end}(\theta_i)$.

$p(\theta_1, \theta_2, \ldots, \theta_n)$ is the payment vector. For mechanisms that are SP and IR, the function p is completely determined by the allocation function t_{end}, as illustrated below.

Proposition 2 *If i is an offender, then we define*

$$x_i(\theta_i) = \int_0^{t_{end}(\theta_i)} c_i(t)dt$$

If i is a defender, then we define

$$x_i(\theta_i) = \int_{t_{end}(\theta_i)}^{1} c_i(t)dt$$

A mechanism is SP and IR if and only if for all i, $x_i(\theta_i)$ is nondecreasing in θ_i, and agent i's payment equals exactly

$$\theta_i x_i(\theta_i) - \int_0^{\theta_i} x_i(z)dz$$

Proof Suppose $x_i(\theta_i)$ is nondecreasing in θ_i and i pays according to the above expression. By reporting θ_i, i's utility equals

$$\theta_i x_i(\theta_i) - \theta_i x_i(\theta_i) + \int_0^{\theta_i} x_i(z)dz = \int_0^{\theta_i} x_i(z)dz \geq 0.$$

The above implies IR. We then show SP. By reporting θ_i', i's utility equals

$$\theta_i x_i(\theta_i') - \theta_i' x_i(\theta_i') + \int_0^{\theta_i'} x_i(z)dz.$$

We subtract the above from i's utility when reporting truthfully, the difference equals

$$\int_0^{\theta_i} x_i(z)dz - \theta_i x_i(\theta_i') + \theta_i' x_i(\theta_i') - \int_0^{\theta_i'} x_i(z)dz.$$

If $\theta_i > \theta_i'$, then the above equals

$$\int_{\theta_i'}^{\theta_i} x_i(z)dz - (\theta_i - \theta_i')x_i(\theta_i') \geq \int_{\theta_i'}^{\theta_i} x_i(\theta_i')dz - (\theta_i - \theta_i')x_i(\theta_i')$$

The right-hand side equals 0. Hence, under-reporting is never beneficial. Similarly, we can show over-reporting is never beneficial.

For the other direction, suppose the mechanism under discussion is SP and IR. We use $p_i(\theta_i)$ to represent i's payment. By SP, we have

$$\theta_i x_i(\theta_i) - p_i(\theta_i) \geq \theta_i x_i(\theta_i') - p_i(\theta_i')$$
$$\theta_i' x_i(\theta_i') - p_i(\theta_i') \geq \theta_i' x_i(\theta_i) - p_i(\theta_i)$$

Combining these two inequalities, we get

$$(\theta_i - \theta_i')x_i(\theta_i) \geq (\theta_i - \theta_i')x_i(\theta_i').$$

Therefore, x_i must be nondecreasing.

By reporting θ_i', i's utility equals $\theta_i x_i(\theta_i') - p_i(\theta_i')$. This is maximized when $\theta_i' = \theta_i$. Also, θ_i is arbitrary. We have $zx_i'(z) = p_i'(z)$. Integrating both sides from 0 to θ_i, we get that i's payment must be as described in the proposition.[4] □

Proposition 2 directly builds on the classic Myerson's characterization of optimal single-item auction [15]. Myerson's characterization is as follows: We use $x_i(\theta_i)$ to represent agent i's allocation (fraction of the single item won) when bidding θ_i. The auction is strategy-proof and individually rational if and only if $x_i(\theta_i)$ is monotone in θ_i and the payment equals $\theta_i x_i(\theta_i) - \int_0^{\theta_i} x_i(z)dz$. We proved that this characterization still holds for our model. The only difference is that the function x_i is defined differently in this paper. Under Myerson's optimal auction, the resulting x_i function must satisfy that it is either 0 or 1 (*i.e.*, whether i wins the item or not). Under our model, we no longer have this restriction. The x_i function values are based on the final outcome t_{end}. As a result, we need an extra step for deciding t_{end}, which needs to be optimal in terms of the revenue, and t_{end} needs to simultaneously ensure that the x_i are monotone. The details are presented below:

For agent i, we assume θ_i is drawn independently from 0 to an upper bound H_i, according to a probability density function f_i (and cumulative density function F_i). Agent i's virtual valuation $\phi_i(\theta_i)$ is defined as

$$\phi_i(\theta_i) = \theta_i - \frac{1 - F_i(\theta_i)}{f_i(\theta_i)}$$

We also need the *monotone hazard rate condition* required by Myerson's technique: the virtual valuation functions are nondecreasing (which is generally true for common distributions).

Given the payment characterization result, the expected payment from agent i equals $E_{\theta_i}(\phi_i(\theta_i)x_i(\theta_i))$. That is, given a type profile, to maximize revenue, we pick t_{end} to maximize $\sum_i(\phi_i(\theta_i)x_i(\theta_i))$. This optimization step is a new step on top of Myerson's technique, which is required by our model. Under our model, x_i is not necessarily bounded between 0 and 1 (for single-item auction, the proportion won by an agent is between 0 and 1). Also, the sum of the x_i is not necessarily bounded above by 1 (for single-item auction, the total proportion allocated is at most 1). Without these bounds, picking the x_i becomes more difficult compared to Myerson's original approach. Fortunately, for our model, an outcome is characterized by a single value, so we can simply find the outcome via one-dimensional numerical optimization. It should be noted that when an agent increases her bid, her virtual valuation also increases according to the monotone hazard rate condition. This leads to

[4] The extra constant term in the integration result equals 0, because whenever an agent's type equals 0, her payment must be 0.

higher value for x_i under our model. That is, the above rule for picking an outcome ensures that the x_i are monotone.

Given the allocation rule described above, the payments are then calculated according to Proposition 2. The resulting mechanism maximizes the expected revenue.

Example 1 We present an example mechanism with two agents:

- Agent 1 is an offender, whose valuation function is $v_1(t) = \theta_1 c_1(t)$. Here θ_1 is agent 1's type, drawn according to a probability density function f_1 (and cumulative density function F_1). c_1 is a publicly-known function. According to Proposition 2,

$$x_1(\theta_1) = \int_0^{t_{end}} c_1(t)dt$$

  It should be noted that t_{end} here depends on θ_1 (and the second agent's type θ_2).
- Agent 2 is a defender, whose valuation function is $v_2(t) = \theta_2 c_2(t)$. Here θ_2 is agent 2's type, drawn according to a probability density function f_2 (and cumulative density function F_2). c_2 is a publicly-known function. According to Proposition 2,

$$x_2(\theta_2) = \int_{t_{end}}^1 c_2(t)dt$$

Given a specific type profile (θ_1, θ_2), we pick t_{end} by *numerically* maximizing the following expression, which is a function of t_{end}.

$$\phi_1(\theta_1)x_1(\theta_1) + \phi_2(\theta_2)x_2(\theta_2)$$

where

$$\phi_i(\theta_i) = \theta_i - \frac{1 - F_i(\theta_i)}{f_i(\theta_i)}$$

Agent 1 receives the exploit information at time 0. t_{end} obtained above determines when agent 2 receives the exploit information. According to Proposition 2, the agents' payments are

$$\theta_i x_i(\theta_i) - \int_0^{\theta_i} x_i(z)dz$$

This mechanism is strategy-proof, individually rational, straight-forward, and maximizes the revenue among all mechanisms with the above properties.

5 General model and affine maximizer auctions

In this section, we return to the general model where an agent's type is characterized by a valuation function instead of a single parameter.

Myerson [15]'s optimal auction is surprisingly elegant, but unfortunately, Myerson's technique does not generalize beyond single-parameter settings. For example, when it comes to combinatorial auctions (auctions where multiple items are for sale, and the agents

bid on bundles of items), revenue maximizing mechanism design remains an open problem. Despite the difficulty of revenue maximization in general, for restricted domains, well performing mechanisms have been obtained based on a variety of revenue-boosting techniques [7, 11, 12, 18]. Here we name a few general-purpose revenue-boosting techniques. For one, we may artificially increase the winning chance of lower bidders, in order to drive up the competition faced by the higher bidders. As another example, we may artificially discourage or outright ban certain outcomes, in order to prevent low-revenue outcomes or force the agents to pay more to achieve the discouraged outcomes. The above techniques form the basis of a family of mechanisms called the *affine maximizer auctions (AMA)*. Lavi *et al.* [13] conjectured that a combinatorial auction is truthful if and only if it is an AMA mechanism, subject to technical conditions. Sandholm and Likhodedov [18] studied revenue maximizing combinatorial auction design by optimizing within the family of AMA mechanisms. The idea of optimizing within the AMA family is a general approach that can be applied to many different mechanism design settings, including ours.

- The AMA mechanisms are generally well defined (*e.g.*, whenever the VCG mechanism is well defined, chances are that the AMA family is well defined).
- The AMA family contains a large number of mechanisms. By optimizing within the AMA family, there is a good chance of reaching a well-performing mechanism in terms of revenue.
- Every AMA mechanism is characterized by a list of parameters. By focusing on the AMA mechanisms, we turn mechanism design into a value optimization problem, where we only need to adjust the parameters.[5]

However, the issue with optimizing within the AMA family is that every AMA mechanism is characterized by $|O| + n$ parameters, where $|O|$ is the size of the outcome space O, and n is the number of agents. For combinatorial auctions, $|O|$ is exponential in the number of items, which makes it computationally impractical. In [18], combinatorial auctions with only two items were considered. With two items, the number of parameters is small enough for the authors to conduct optimization via grid-based gradient descent.[6]

In this paper, we propose three numerical techniques that speed up the parameter tuning for AMA. Our techniques are not overly model specific so they have the potential to be applied to other mechanism design settings. One technique uses linear programming to identify a near optimal local perturbation on the parameters. The second technique is based on neural networks. The third technique is based on evolutionary computation. All three techniques can easily handle AMA mechanisms with hundreds of parameters. All three techniques lead to near optimal mechanisms in experiments.

Besides evaluating our techniques via numerical experiments, we also show that if there are only two agents (one offender and one defender), and if the defender's valuation is much lower than the offender's (typically true for zero-day exploit markets), then optimizing within the AMA family guarantees to lead to near optimal mechanisms.

[5] [10] summarized the computationally feasible automated mechanism approach which transforms mechanism design into optimization within parameterized mechanism families.

[6] The authors also proposed a restricted version of AMA called the VVCA mechanisms. A VVCA mechanism is only characterized by $2n$ parameters, which makes it much easier to optimize over. On the other hand, due to the fact that the VVCA family is only a tiny subset of the whole AMA family, we lose revenue by focusing only on it.

That is, under these conditions, it is without loss of generality to focus on the AMA mechanisms.

Next, we present the formal definition of the AMA mechanisms. We use O to denote the outcome space. We use Θ_i to denote agent i's type space. We use $v_i(\theta_i, o)$ to denote agent i's valuation for outcome $o \in O$ when her type is $\theta_i \in \Theta_i$. Under our model, the outcome space is [0, 1]. An outcome $o \in [0, 1]$ represents when the exploit is killed off (revealed to the defenders). In order to run the techniques proposed in this paper, we require the outcome space to be finite. So for this technical reason, we set the outcome space to be $\{0, \frac{1}{k}, \frac{2}{k}, \ldots, 1\}$. That is, we will only reveal the exploit at these discrete moments. The size of the outcome space $|O| = k + 1$.

The family of AMA mechanisms is defined as follows:

AMA Mechanisms

– Given a type profile θ, the outcome picked is the following:

$$o^* = \arg\max_{o \in O} \left(\sum_{i=1}^{n} u_i v_i(\theta_i, o) + a_o \right)$$

– Agent i's payment equals:

$$\frac{\max_{o \in O} \left(\sum_{j \neq i} u_j v_j(\theta_j, o) + a_o \right) - \sum_{j \neq i} u_j v_j(\theta_j, o^*) - a_{o^*}}{u_i}$$

In the above definition, the u_i and the a_o are mechanism parameters (*i.e.*, constants that need to be adjusted in order to locate a high-revenue AMA mechanism).

Example 2 Using the same setting from Example 1, we present an example AMA mechanism with two agents:

– Agent 1 is an offender, whose valuation function is $v_1(t) = \theta_1 c_1(t)$. Here θ_1 is agent 1's type. c_1 is a publicly-known function.
– Agent 2 is a defender, whose valuation function is $v_2(t) = \theta_2 c_2(t)$. Here θ_2 is agent 2's type. c_2 is a publicly-known function.

Given a specific type profile (θ_1, θ_2), the AMA mechanism picks the following outcome o^*. o^* is the time moment when we release the exploit to the defender (agent 2).

$$\begin{aligned} o^* &= \arg\max_{o \in [0,1]} \left(u_1 v_1(\theta_1, o) + u_2 v_2(\theta_2, o) + a_o \right) \\ &= \arg\max_{o \in [0,1]} \left(u_1 \theta_1 \int_0^o c_1(t)dt + u_2 \theta_2 \int_o^1 c_2(t)dt + a_o \right) \end{aligned}$$

Agent 1' payment is then

$$\frac{\max_{o\in[0,1]}\left(u_2\theta_2\int_o^1 c_2(t)dt + a_o\right) - u_2\theta_2\int_{o^*}^1 c_2(t)dt - a_{o^*}}{u_1}$$

Agent 2' payment is then

$$\frac{\max_{o\in[0,1]}\left(u_1\theta_1\int_0^o c_1(t)dt + a_o\right) - u_1\theta_1\int_0^{o^*} c_1(t)dt - a_{o^*}}{u_2}$$

An AMA mechanism is very similar to the VCG mechanism, where the only difference is that the agents' valuations undergo a linear transformation: agent i's valuation for outcome o is transformed from $v_i(\theta_i, o)$ to $u_i v_i(\theta_i, o) + a_o$, where the u_i and the a_o are constant parameters. The idea behind the AMA mechanisms is that, if agent i is likely to lose according to the prior distribution, then by assigning a larger coefficient u_i, we increase the competition, therefore increases revenue. Also, if an outcome o is frequently chosen and the agents have high surplus on this outcome, then by assigning a negative a_o, the agents are forced to pay more for this outcome.

In the above description, the u_i must satisfy $u_i \geq 1$ for all i. The a_o are unrestricted. In total, there are $n + |O|$ parameters. For every assignment of the parameters, the corresponding AMA mechanism is strategy-proof. However, not every AMA mechanism is individually rational. If we further assume that $\forall i, \theta_i, o$, we have $v_i(\theta_i, o) \geq 0$, then every AMA mechanism is individually rational. To show this, we only need to show that an agent's valuation is always at least her payment. That is,

$$v_i(\theta_i, o^*) \geq \frac{\max_{o\in O}\left(\sum_{j\neq i} u_j v_j(\theta_j, o) + a_o\right) - \sum_{j\neq i} u_j v_j(\theta_j, o^*) - a_{o^*}}{u_i}$$

$$\iff \sum_j u_j v_j(\theta_j, o^*) + a_{o^*} \geq \max_{o\in O}\left(\sum_{j\neq i} u_j v_j(\theta_j, o) + a_o\right)$$

The right-hand side is less than or equal to the left-hand side if every agent's valuation for every outcome is nonnegative.

In our model, we do have that every agent's valuation for every outcome is nonnegative, so in our model, every AMA mechanism is individually rational.

6 Optimizing affine maximizer auctions via linear programming

In this section, we introduce a linear programming based technique for approximating the parameter gradient when we optimize within the AMA family.

We recall that an AMA mechanism is characterized by $n + |O|$ parameters (u_i for every agent i, and a_o for every outcome o). For presentation purpose, we define $Z = n + |O|$ and use $p_1, p_2, \ldots, p_Z$ to refer to the parameters. Let $M(p_1, p_2, \ldots, p_Z)$ be the AMA mechanism characterized by p_1 to p_Z. The task of optimizing within the AMA family is simply to optimize over the parameters:

$$\max_{p_1,p_2,\ldots,p_Z} ER(M(p_1,p_2,\ldots,p_Z))$$

Here, $ER(M)$ represents mechanism M's expected revenue. We have analytical characterization of the AMA payments, so the revenue of M given a specific type profile can be calculated accordingly. Unfortunately, there is no known short-cut for calculating the expected revenue. Given a prior distribution of the θ_i, we need to draw large amount of sample profiles to calculate the expected revenue. For example, if for every agent i, we draw 100 samples for θ_i, then altogether the number of type profiles is 100^n. For this reason, in this paper, we focus on cases where n is small.[7]

Sandholm and Likhodedov [18] used a grid-based gradient descent approach for optimizing the parameters. Under this approach, suppose we start from a grid point $(p_1, p_2, \ldots, p_Z)$, we have to examine all neighbouring points $(p_1 + \delta_1 h, p_2 + \delta_2 h, \ldots, p_Z + \delta_Z h)$, where $\delta_i \in \{-1, 0, 1\}$ and h is the grid size. We need to examine 3^Z points. So this approach requires that both n and $|O|$ be tiny. For example, if $Z = 100$, then this approach is impractical. For our techniques, we still require tiny n, but Z is allowed to be large.

A high-level description of our linear programming based technique is as follows:

- We initialize the algorithm with an initial AMA mechanism: *e.g.*, it could be based on random parameters, or we may start with the VCG mechanism.
- Given M_0 characterized by $p_1^0, p_2^0, \ldots, p_Z^0$, we use a linear program to (approximately) locate the best AMA mechanism *near* the starting point M_0. A mechanism M (characterized by $p_1, p_2, \ldots, p_Z$) is *near* M_0 if $\max_i |p_i - p_i^0| \leq \epsilon$ for a constant threshold ϵ. We repeat this step using the new mechanism as the starting point. Essentially, we are performing gradient descent, and the linear programs are there to approximate the gradient calculation.

Now we present the details of the linear program. We index the outcomes using $0, 1, \ldots, k$. We denote the initial mechanism as $M(u_1^0, u_2^0, \ldots, u_n^0, a_0^0, a_1^0, \ldots, a_k^0)$.

The following optimization model solves for the best AMA mechanism that is near this starting point:

Model 1
Variables: $u_1, u_2, \ldots, u_n, a_0, a_1, \ldots, a_k$
Maximize: $ER(M(u_1, u_2, \ldots, u_n, a_0, a_1, \ldots, a_k))$
Subject to:
For all i, $u_i \geq 1$ and $u_i^0 - \epsilon \leq u_i \leq u_i^0 + \epsilon$
For all t, $a_i^0 - \epsilon \leq a_i \leq a_i^0 + \epsilon$

Of course, the above model is not a linear program, as $ER(M)$ is not a linear combination of the variables. *We will approximate ER(M) using a linear combination of the variables.*

Let S be a large set of type profiles, we will approximate $ER(M)$ as follows:

[7] We have to emphasize that this is not an uncommon constraint when it comes to automated mechanism design.

$$ER(M) \approx \sum_{\theta \in S} P(\theta) \sum_i C_i(M, \theta)$$

Here, $C_i(M, \theta)$ is agent i's payment under M when the type profile is θ. One way to pick S is to discretize the type space and let S be the set of all grid points. Now what remains to be done is to approximate $C_i(M, \theta)$ using a linear combination of the variables.

$$C_i(M, \theta) = \frac{\max_{o \in O} \left(\sum_{j \neq i} u_j v_j(\theta_j, o) + a_o \right) - \sum_{j \neq i} u_j v_j(\theta_j, o^*) - a_{o^*}}{u_i}$$

Here, o^* is defined as

$$o^* = \arg\max_{o \in O} \left(\sum_{i=1}^{n} u_i v_i(\theta_i, o) + a_o \right)$$

We use the following heuristic to approximate $C_i(M, \theta)$: because the u_i and the a_o are close to the u_i^0 and the a_o^0, we will use the u_i^0 and the a_o^0 to calculate the outcomes mentioned in the above expressions. That is, we assume that for most type profiles, small perturbation in the parameters will not change the mechanism outcomes.

$$o^{*0} = \arg\max_{o \in O} \left(\sum_{i=1}^{n} u_i^0 v_i(\theta_i, o) + a_o^0 \right)$$

$$o^0 = \arg\max_{o \in O} \left(\sum_{j \neq i} u_i^0 v_i(\theta_i, o) + a_o^0 \right)$$

We replace o^* and o using o^{*0} and o^0, we have that

$$C_i(M, \theta) \approx \frac{\sum_{j \neq i} u_j v_j(\theta_j, o^0) + a_{o^0} - \sum_{j \neq i} u_j v_j(\theta_j, o^{*0}) - a_{o^{*0}}}{u_i}$$

We use c_j to denote $v_j(\theta_j, o^0)$ and c_j^* to denote $v_j(\theta_j, o^{*0})$. Both the c_j and the c_j^* are *constants*.

$$C_i(M, \theta) \approx \frac{\sum_{j \neq i} c_j u_j + a_{o^0} - \sum_{j \neq i} c_j^* u_j - a_{o^{*0}}}{u_i}$$

We then observe that for any x, $\frac{x}{u_i} = \frac{x}{u_i^0} - \frac{x(u_i - u_i^0)}{u_i u_i^0}$.

We can then rewrite $C_i(M, \theta)$ into:

$$\frac{\sum_{j \neq i} c_j u_j + a_{o^0} - \sum_{j \neq i} c_j^* u_j - a_{o^{*0}}}{u_i^0} - \frac{(\sum_{j \neq i} c_j u_j + a_{o^0} - \sum_{j \neq i} c_j^* u_j - a_{o^{*0}})(u_i - u_i^0)}{u_i u_i^0}$$

The first term is a linear combination of the variables, as u_i^0, the c_j, and the c_j^* are all constants.

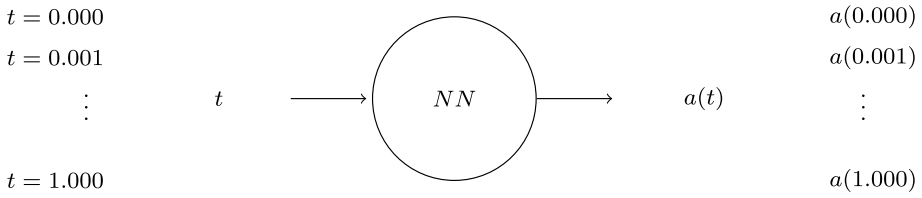


Fig. 1 Neural network representation of the a_i when $k = 1000$

The second term can be approximated as follows:

$$\frac{\left(\sum_{j\neq i} c_j u_j^0 + a_{o^0}^0 - \sum_{j\neq i} c_j^* u_j^0 - a_{o^{*0}}^0\right)(u_i - u_i^0)}{u_i^0 u_i^0}$$

The above is a linear function involving one variable (u_i). The idea of the approximation is that we replace several occurrences of u_i by u_i^0 since their values are close, but we leave the term $u_i - u_i^0$ untouched as this term is the most important component for the overall gradient.

Using the above heuristic-based approximation, we are able to turn Model 1 into a linear program involving $n + |O|$ variables. The number of constraints is $n + |O| + |S|$. Therefore, we can afford reasonably large $|O|$, as long as $|S|$ is not too large. By iteratively solving such linear programs, we are able to optimize within the AMA family.

7 Optimizing affine maximizer auctions via neural networks

An AMA mechanism $M(u_1, u_2, \ldots, u_n, a_0, a_1, \ldots, a_k)$ is characterized by $n + |O|$ parameters. As mentioned earlier, given a specific AMA mechanism, to evaluate its expected revenue, we resort to generating a large set of sample profiles S based on the prior distribution. For each sample profile $\theta \in S$, we evaluate the corresponding revenue and then take the average over all samples in S. Due to this, we require n to be tiny so that we can accurately evaluate the expected revenue using S. For large n, we cannot afford to evaluate a mechanism's expected revenue, not to mention carrying out mechanism optimization. In our linear programming based approach, the number of constraints is $n + |O| + |S|$. That is, realistically, the largest S we can work with is in the magnitude of "hundreds".

Mechanism design via neural networks has recently drawn significant attention in the algorithmic game theory community [5, 14, 19, 20]. In the context of our model, the high-level approach of tuning AMA parameters using neural networks is as follows:

- Treat the u_i and the a_i as model parameters.
- Initialize the model parameters randomly or start from a known mechanism such as the VCG mechanism. We can also run a supervised learning step to set the starting mechanism to be the mechanism produced by the linear programming based approach.
- In each learning step, we generate a batch of type profiles based on the prior distribution. We evaluate the current AMA mechanism's average revenue on this batch. Param-

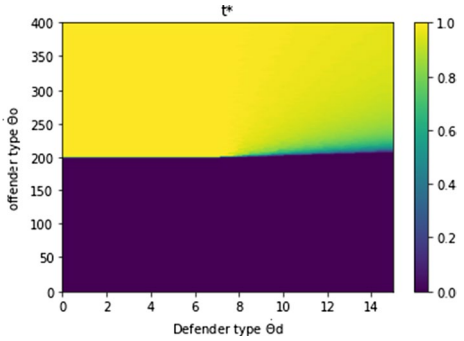


Fig. 2 Ending time t^* as a function of type profiles for the optimal mechanism

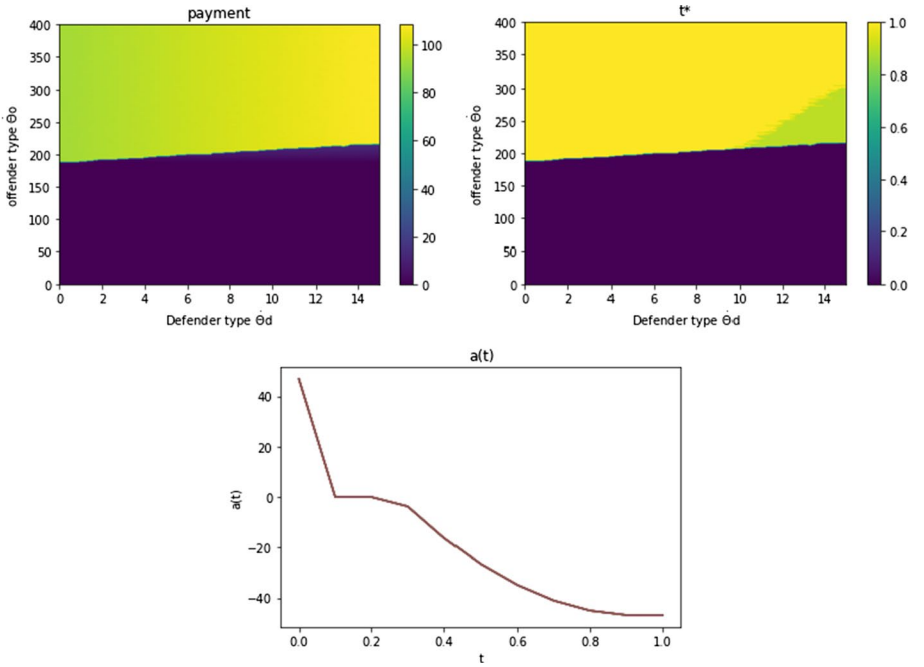


Fig. 3 Total payment, ending time t^*, and $a(t)$ as functions of type profiles for the AMA mechanism derived via LP

eter gradient is calculated based on this average. Model parameters are adjusted via gradient descent.

The *training data* are randomly sampled based on the prior distribution. In each learning step, we generate a fresh batch of type profiles. After we finish training, we generate another (much larger) fresh batch of type profiles to be the *testing data*. It should be noted that we never need to reuse any type profile, so there is a separation between training and testing data.

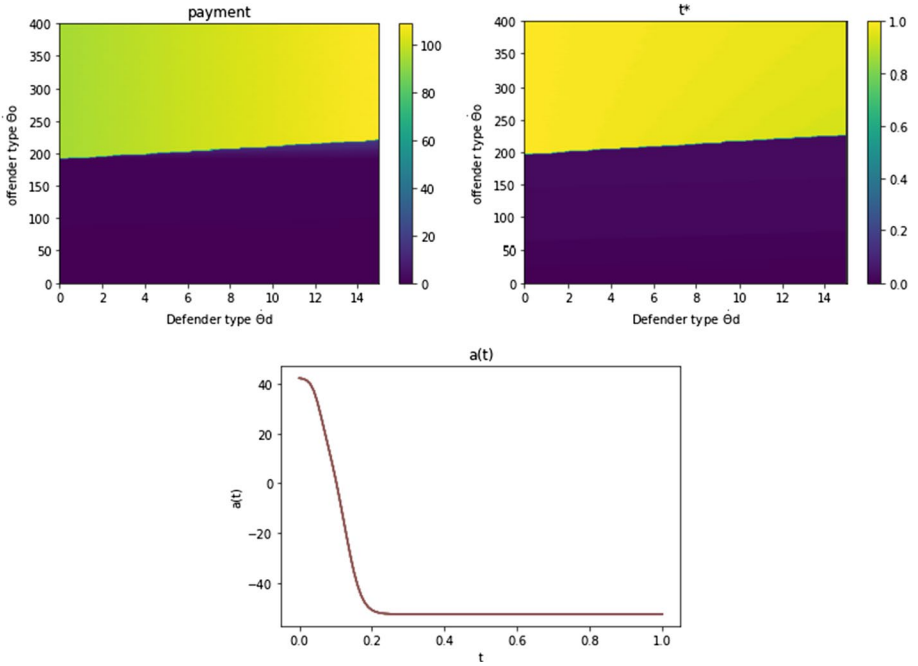


Fig. 4 Total payment, ending time t^*, and $a(t)$ as functions of type profiles for the AMA mechanism derived via NN

One limitation of the neural network approach is that the batch size in each learning step needs to small (*e.g.*, we set the batch size to be 16). If the batch size is very large, the time consumption of each learning step gets too long, and it actually hurts the learning performance. Obviously, it is insufficient to use only 16 type profiles to *accurately* estimate a mechanism's expected revenue, but generally speaking, we do not need every learning step to move toward the correct direction. We only need that in *most* learning steps we are moving toward the correct direction.

Due to the batch size limitation, for our neural network based approach, we focus on the case with only two agents (one defender and one offender). A batch of 16 type profiles can be generated by drawing 4 sample types for each agent (Figs. 1, 2, 3, 4).

With two agents, an AMA mechanism can be expressed as

$$M(u_{offender}, u_{defender}, a_0, a_1, \ldots, a_k)$$

It is without loss of generality to set $u_{offender} = 1$. Besides the model parameter $u_{defender}$, the only other mechanism parameters are the a_i. We recall that a_i represents the constant term in the AMA mechanism for outcome $\frac{i}{k}$. That is, the a_i can naturally be represented using a curve $a(t)$ with $t \in [0, 1]$, where $a(t)$ is the constant term for outcome t. The function $a(t)$ can be expressed using the following neural network:

In the context of the above representation, the AMA mechanism's allocation (the ending time) equals

$$t^* = \arg\max_{t\in[0,1]} \left(\sum_{i\in N} u_i v_i(\theta_i, t) + a(t) \right)$$
$$N = \{offender, defender\}$$

Agent i's payment equals ($i \in \{offender, defender\}$):

$$p_i = \frac{\max_{t\in[0,1]} \left\{ \sum_{j\neq i} u_j v_j(\theta_j, t) + a(t) \right\} - \left\{ \sum_{j\neq i} u_j v_j(\theta_j, t^*) + a(t^*) \right\}}{u_i}$$

To maximise the total revenue, the loss function is set to be:

$$minimise : loss = -(p_{offender} + p_{defender})$$

In training, we set $u_{defender}$ as an *autograd* [8] parameter, and we use a fully connected network to represent the function $a(t)$. We assume that we have the analytical form of the agents' valuation function $v_i(\theta_i, t)$, which is to facilitate automatic differentiation needed by gradient descent. We will present the experimental results in Sect. 10.

8 Optimizing affine maximizer auctions via evolutionary computation

As discussed in the above section, an AMA mechanism is characterized by a curve $a(t)$ with $t \in [0, 1]$.[9] In the previous section on neural networks, we used neural networks to model the curve $a(t)$. A natural idea is to consider other methods for expressing curves, such as:

- Piece-wise linear segments: We may join k straight-line segments to form a curve. The coordinates for the end points are $(\frac{j}{k}, c_j)$ for $j = 0, 1, \ldots, k$. To optimize for the best piece-wise linear segments, we just need to adjust the c_j.
- Polynomial:

$$a(t) = c_k t^k + c_{k-1} t^{k-1} + \ldots + c_1 t + c_0$$

  To optimize for the best polynomial representation of $a(t)$, again, we just need to adjust the c_j.
- Fourier series:

$$a(t) = \frac{c_0}{2} + \sum_{j=1}^{N} \left(c_j cos\left(\frac{2\pi}{p} jt\right) + c'_j sin\left(\frac{2\pi}{p} jt\right) \right)$$

  To optimize for the best Fourier series representation of $a(t)$, we need to adjust the c_j and the c'_j.

For all the above representation models, we use the following genetic algorithm to adjust the parameters:

[8] For an overview of automatic differentiation in PyTorch, please refer to [16].

[9] For two agents, besides the curve $a(t)$, we also have another model parameter $u_{defender}$, which is a single parameter that can be dealt with separately (using a naive for loop).

Algorithm 1: Genetic Algorithm

Step 1: Create an initial population of 60 curves (PopSize = 60). All initial curves are randomly generated. For piece-wise linear segments, the initial random curves are straight lines $a(t) = st + b$ where s, b are drawn randomly from $U(-100, 100)$. We choose a random slope from $U(-100, 100)$ so that the resulting straight lines have similar *vertical swings* to the curves obtained via linear programming and neural networks. For polynomials and Fourier series, all parameters are drawn randomly from $U(-10, 10)$.

Step 2: Evolution:

while *Have not reached the* 100*-th round* **do**

  for *Each Individual* **do**

    Randomly choose 200 type profiles to test fitness (revenue);

  Sort individuals according to fitness;

  Selection:

    Keep the top 20 individuals in terms of fitness (EliteSize = 20);

  Add new individuals through **Crossover** and **Mutation** until we reach Popsize;

  1. Crossover: Generate 20 new curves via standard two-point crossover.

  2. Mutation: Generate 20 new curves via perturbing existing curves (with 0.1 probability, a parameter is increased or decreased by $\delta = 0.5$).

Step 3: Testing:

Average over 10000 random type profiles to choose the best individual.

We will present the experimental results in Sect. 10.

9 Optimality of AMA: two agent setting

We introduced three techniques for optimizing within the AMA family. A natural question to ask is whether the obtained AMA mechanism's revenue is close to the optimal revenue under the globally optimal mechanism, which is not necessarily an AMA mechanism. In this section, we try to answer this question in a setting with only two agents: one offender and one defender. We derive a numerical revenue upper bound. According to experiments, both of our proposed techniques are able to identify AMA mechanisms whose revenues are close to the upper bound. Furthermore, when the defender's valuation is much lower than the offender's valuation, which is generally true for zero-day exploit markets in practise, our upper bound shows that optimizing within the AMA family does lead to near optimal mechanisms.

We use $EPO(M)$ to denote the offender's expected payment under mechanism M.

We use $EPD(M)$ to denote the defender's expected payment under mechanism M.

We still use $ER(M)$ to denote mechanism M's expected revenue.

Let F be the set of all straight-forward, strategy-proof and individually rational mechanisms.

Let M^* be the globally optimal mechanism.

$$M^* = \arg\max_{M \in F} (EPO(M) + EPD(M)) = \arg\max_{M \in F} ER(M)$$

Let MO^* be the optimal mechanism that maximizes the expected payment collected from the offender.

$$MO^* = \arg\max_{M \in F} EPO(M)$$

Let MD^* be the optimal mechanism that maximizes the expected payment collected from the defender.

$$MD^* = \arg\max_{M \in F} EPD(M)$$

By definition, we have the following proposition.

Proposition 3 *The optimal revenue $ER(M^*)$ has the following upper bound:*

$$ER(M^*) = EPO(M^*) + EPD(M^*) \leq EPO(MO^*) + EPD(MD^*) \quad (1)$$

Next we discuss how to calculate $EPO(MO^*)$ and $EPD(MD^*)$. In other words, we discuss how to derive mechanisms that focus on maximizing one agent's payment.

Proposition 4 *It is without loss of generality to assume that both MO^* and MD^* are posted-price mechanisms with the following form:*

- *Every outcome o is associated with a price a_o.*
- *One agent picks the outcome that maximizes her own utility.*
- *The other agent makes no decisions and pays 0.*

We use $PP(a_0, a_1, \ldots, a_k)$ to denote the posted-price mechanism with the parameters a_0 to a_k.

Proof Let MO^* be the mechanism that maximizes the offender's expected payment. Let $EPO(M, \theta_D)$ be the offender's expected payment under M when the defender bids θ_D. Because $EPO(MO^*) = \sum_{\theta_D} P(\theta_D) EPO(MO^*, \theta_D)$, there must exist one θ_D so that $EPO(MO^*, \theta_D) \geq EPO(MO^*)$. If we fix the defender's type to be the said θ_D (we can do this by ignoring the defender's bid and always ask the defender to pay 0), then the mechanism faced by the offender is exactly a posted-price mechanism, and the expected payment of the offender is at least $EPO(MO^*)$. □

In what follows, we discuss how to solve for the optimal posted-price mechanism. For presentation purposes, we focus on solving for the optimal posted-price mechanism where the offender makes decisions.

We start with the single-parameter setting. Using the same approach presented in Sect. 4, when there is only one agent, we can solve for the optimal revenue-maximizing mechanism. Here, we have only one agent denoted by agent 1. The mechanism picks an ending time that maximizes $\phi_1(\theta_1) x_1(\theta_1)$, where $\phi_1(\theta_1)$ is the virtual valuation of agent 1 and $x_1(\theta_1)$ is defined according to Proposition 2. It turns out that the optimal mechanism simply allocates agent 1 the whole time interval if and only if her virtual valuation is positive. The optimal posted-price mechanism therefore has the form $PP(0, +\infty, \ldots, +\infty, b)$ where b is the price of the whole time interval. The optimal b value can be calculated via a one-dimensional numerical optimization.

When k is small, we have another algorithm for solving for the optimal posted-price mechanism, and under this algorithm, we can drop the single-parameter assumption.

Let $PP(a_0, a_1, \ldots, a_k)$ be the optimal posted-price mechanism. It is without loss of generality to assume that for any $i < j$, if both a_i and a_j are finite, then $a_i < a_j$. Otherwise, the outcome i is never chosen, and we can set a_i to be infinite. It is also without loss of generality to assume that for any a_i, either it is infinite (meaning that this outcome is not allowed), or it must satisfy the following condition:

$$\exists \theta_O, v(i, \theta_O) - a_i = \max_{j<i} v(j, \theta_O) - a_j$$

Here, $v(t, \theta_O)$ is the offender's valuation for outcome t when her type is θ_O. The above condition basically says that there exists a type for the offender, if we increase a_i just a bit, then it would force the offender to choose an earlier outcome than i. If the condition is not true, then we can safely increase a_j. By doing so, we can charge more for those types that choose j. We can also charge more if under some types, the offender chooses a later outcome, which also means that more payment will be collected.

Based on the above condition, if we know the a_j for $j < i$ and θ_O, then we can calculate a_i. We already know that $a_0 = 0$. To calculate a_1, we can go over all θ_O, possibly by discretizing the offender's type space. Then, to calculate a_2, we can go over all θ_O again. We do this for every i. Let N be the number of types in Θ_O after the discretization. The total number of iterations is then N^k.

Next, we consider a scenario where the defender has a much lower valuation than the offender. This is typically true in practise under a zero-day exploit market. For example, according to [9], an exploit that attacks the Chrome browser sells between 80k and 200k for offensive clients (USD). According to Google's official bug bounty reward program for the Chrome browser [17], a serious exploit is priced between 0.5k and 15k. In this scenario, $EPD(MD^*)$ is generally much smaller than $EPO(MO^*)$.

According to Eq. 1, we have that $EPO(MO^*)$ is close to the optimal revenue $ER(M^*)$ when $EPD(MD^*)$ is small. Our next proposition says that the optimal AMA mechanism's revenue is at least $EPO(MO^*)$, which essentially means that the optimal AMA revenue is close to the global optimal revenue.

Proposition 5 *The optimal AMA revenue (in expectation) is at least $EPO(MO^*)$.*

Proof We use $PP(a_0, a_1, \ldots, a_k)$ to denote the posted-price mechanism with the parameters a_0 to a_k. We use $M(u_O, u_D, a_0, a_1, \ldots, a_k)$ to denote the AMA mechanism with the parameters u_O (for offender), u_D (for defender), and the a_t. If the deciding agent under $PP(a_0, a_1, \ldots, a_k)$ is the offender, then $PP(a_0, a_1, \ldots, a_k)$ approaches $M(u_O, 1, 0, -a_1 u_O, \ldots, -a_k u_O)$, when u_O approaches infinity. If the deciding agent under $PP(a_0, a_1, \ldots, a_k)$ is the defender, then $PP(a_0, a_1, \ldots, a_k)$ approaches $M(1, u_D, -a_0 u_D, \ldots, -a_{k-1} u_D, 0)$, when u_D approaches infinity. For any posted-price mechanism, there exists an AMA mechanism whose expected performance is arbitrary close to it. That is, the optimal AMA mechanism's expected revenue is at least the optimal expected revenue of posted-price mechanisms. Combine this with Proposition 4, we have that the optimal AMA revenue is close to the expected revenue of the optimal mechanism M^*. □

10 Experiments

According to [9], an exploit that attacks the Chrome browser sells for at most 200k for offensive clients (USD). According to Google's official bug bounty reward program for the Chrome browser [17], a serious exploit is priced for at most 15k. We adopt an experimental setting based on the numbers above.

There are two agents. The offender's valuation function is

$$v(\theta_O, t) = \int_0^t \theta_O(1 - x)dx$$

θ_O is drawn uniformly at random from $U(0, 400)$. That is, the offender's valuation for the whole time interval [0, 1] is at most 200. The offender gets less and less interested in the exploit as time goes on (the instantaneous valuation gets to 0 as $1 - x$ approaches 0 when x approaches 1).

The defender's valuation function is

$$v(\theta_D, t) = \int_t^1 \theta_D x dx$$

θ_D is drawn uniformly at random from $U(0, 15)$. That is, the defender's valuation for the whole time interval [0, 1] is at most 15. The defender's instantaneous valuation in the exploit does not change over time.

10.1 Upper bound on the revenue: 53.75

The above valuation functions satisfy all the conditions needed for the single-parameter model. Based on the upper bound proposed in Sect. 9, we have that MO^* simply sells the whole interval to the offender for a fixed price p_O, and MD^* simply sells the whole interval to the defender for a fixed price p_D.

$$p_O = \arg\max_{p \le 200} pP(v(\theta_O, 1) \ge p) = \arg\max_{p \le 200} p\frac{200 - p}{200} = 100$$
$$EPO(MO^*) = 50$$
$$p_D = \arg\max_{p \le 15} pP(v(\theta_D, 0) \ge p) = \arg\max_{p \le 15} p\frac{15 - p}{15} = 7.5$$
$$EPD(MD^*) = 3.75$$

Therefore, the revenue upper bound according to Proposition 3 is 53.75.

10.2 Optimal revenue: 50.55

Since the valuation functions satisfy all the conditions needed for the single-parameter model, we are able to derive the revenue-maximizing mechanism.

The ending time is based on the following rule:

$$t^* = \arg\max_{t \in [0,1]} \{(2 \times \theta_O - 400)(t - t * t/2) + (2 \times \theta_D - 15)(1 - t)\}$$

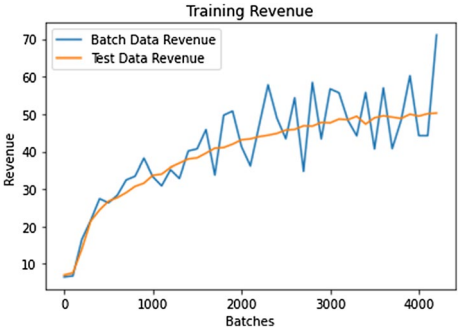


Fig. 5 Neural network training process

Table 1 Revenue via evolutionary computation: different representations of $a(t)$

Segmented straight line (50 segments):	47.67
Quartic polynomial:	38.80
Sextic polynomial:	37.20
Fourier series (N=5, p=2) :	44.54
Fourier series (N=30, p=2) :	46.88

The optimal revenue equals 50.55.

10.3 Revenue via iterative linear programming: 50.14

We pick $k = 10$ and $\epsilon = 0.01$. We use the VCG mechanism as the initial solution. The VCG mechanism's expected revenue is 6.9, which is very far away from the upper bound. Our technique starts from the VCG mechanism, and at the end produces a mechanism whose expected revenue equals 50.14.

10.4 Revenue via neural networks: 50.31

We apply the PyTorch library with the following setup:

- We use a fully connected network with three hidden layers (200 nodes per layer) to represent $a(t)$.
- We use the Adam optimizer with a learning rate of 0.0001.
- The batch size is set to 16.
- The training set consists of 80000 randomly generated type profiles.
- The testing set consists of 20000 randomly generated type profiles.

The achieved expected revenue equals 50.31.

10.4.1 Main computational challenges for the neural networks

Setting aside $u_{defender}$, every AMA mechanism is characterized by a **curve** $a(t)$, where $t \in [0, 1]$. We note that the main computational bottleneck is due to the learning batch size. Our fully-connected network structure with 3 layers and 200 nodes per layer is more than

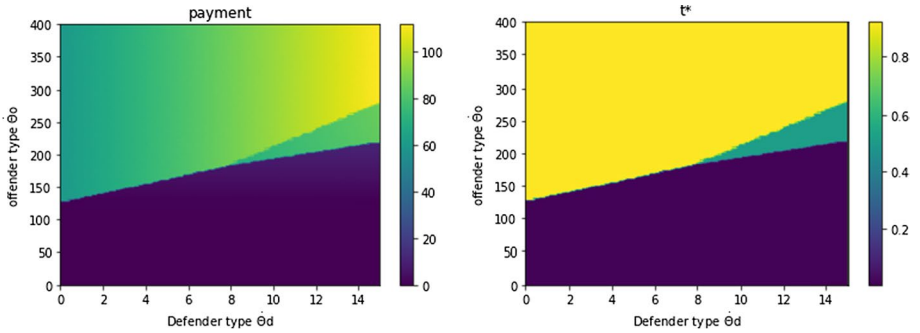


Fig. 6 Total payment and ending time t^* as functions of type profiles for the AMA mechanism derived via evolutionary computation (Fourier series N = 30)

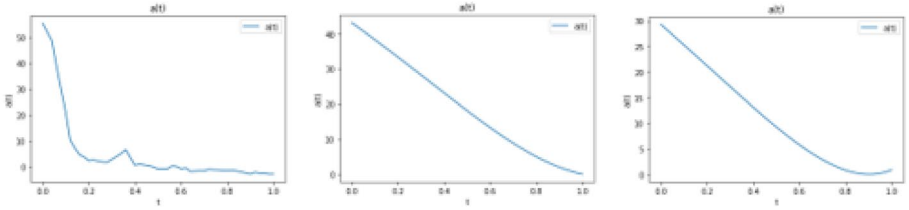
expressive enough to represent curves. In our experiments, we generate 20, 000 type profiles to evaluate an AMA mechanism's expected revenue. We cannot afford to do this in every learning iteration. Instead, we use a batch size of 16 during learning. That is, we use the average revenue of 16 randomly generated type profiles to estimate the parameter gradient. Certainly, the gradient direction obtained this way is not very accurate. Fortunately, as long as the learning rate is small, and as long as most of the time, the gradient direction is generally correct, then we still are able to successfully train the neural network. Figure 5 is an illustration of the neural network training process. As shown in Fig. 5, the average revenue of 16 randomly generated type profiles perturbs wildly[10], but the mechanism is still improving steadily during the training, which shows that a small batch is good enough for training.

10.5 Revenue via evolutionary computation: 47.67

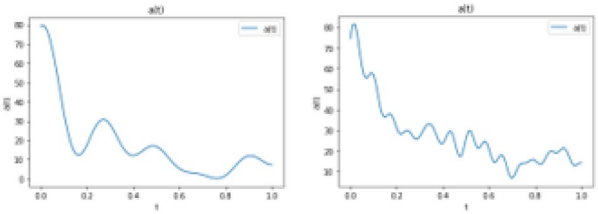
Our last numerical technique approximates $a(t)$ using segmented straight lines, polynomials, and Fourier series. The parameters (*i.e.*, polynomial coefficients) are adjusted according to an evolutionary algorithm (Table 1) (Algorithm 1).

We present all the $a(t)$ functions below:

[10] In Fig. 5, the presented data points for the batch data revenue are the average revenue for every 100 batches.



(a) Segmented straight lines (50 segments) (b) Quartic (c) Sextic



(d) Fourier series N=5 (e) Fourier series N=30

For Fourier series (N = 30), the payment and ending time are as follows (Fig. 6):

10.6 Comparison of different AMA solution techniques

The neural network based approach produces slightly better result than the linear programming based approach. On the other hand, the neural network based approach realistically only works for two agents and cannot deal with black-box valuation functions, as it requires the valuation functions' analytical forms for auto differentiation. The evolutionary computation technique scales the best if we adopt a representation with small number of parameters (*i.e.*, if we approximate $a(t)$ using a quadratic polynomial such as $a(t) = c_2t^2 + c_1t + c_0$, then we only need to adjust three parameters). The down-side of the evolutionary computation technique is that all the representations we have tried (polynomials or Fourier series) are not as expressive as neural networks, so the achieved revenue using the evolutionary computation technique is slightly worse.

11 Conclusion

In this paper, we study markets for zero-day exploits from a revenue-maximizing mechanism design perspective. We proposed a theoretical mechanism design model for zero-day exploits markets. By requiring a new mechanism property called straight-forwardness, we also showed that for the purpose of designing revenue-maximizing mechanisms, it is without loss of generality to focus on mechanisms that "divide" the time frame into two regions, which makes our model similar to the cake-cutting problem.

We first considered a simplified single-parameter model, where every agent's type is characterized by a single parameter. With necessary modification and extension at the last

step, we were able to apply Myerson's classic technique for designing optimal single-item auction to our model and derived the optimal mechanism for single-parameter models.

For the general model, we adopted the computationally feasible automated mechanism design approach. We focused on the AMA mechanisms. To identify an AMA mechanism with high revenue, we proposed three numerical techniques for optimizing within the AMA family. All techniques are able to produce near optimal mechanisms.

11.1 Future work

We propose several future directions below:

- Welfare-maximizing markets: In this paper, the aim of our mechanism design is to maximize revenue. One future research direction is to consider welfare-maximizing markets, *i.e.*, markets that aim to maximize the participants' combined utility.
- Markets without defenders: The core idea of our mechanism is that we reveal the exploit to all offenders, and they must pay in order to keep the exploit alive. We have a defender to act as a threat. *I.e.*, if the offenders do not pay enough, then the exploit is given to the defender, who is capable of rendering the exploit useless. The above idea does not work for scenarios without defenders or scenarios where the defenders are not capable of fixing the bug (*i.e.*, hardware exploits are not easily fixable for existing users). We need new mechanisms for such settings. One initial idea is to sell exploits via *staged releases*. We could gradually reveal (more and more) information regarding the exploit to the offenders. The offenders pay to speed up the release. For example, we gradually reveal information and at some point, the revealed information is enough to convince the offenders that the exploit is valuable, then the offenders will pay in order to speed up the release. Earlier access to zero-day exploits is often deemed valuable—for example, an exploit is more effective when most servers have not yet been patched.
- Markets with a "not sell" option: Our mechanisms reveal the exploit to all offenders before they bid, which means that there is not a "not sell" option. If all agents have 0 valuations for the exploit, then the final revenue is 0. Therefore, under our mechanisms, the seller faces the risk of revealing the exploit without pay. It is desirable to be able to implement a reserve price. One idea to implement a reserve price is to only reveal the exploit to a small group of offenders, and then based on their bids, estimate whether a larger-scale revealing is able to achieve the desired revenue threshold. We could also try to ensure that the offenders agree to pay before we reveal the exploit to them. For example, we could use some offenders' actions to signal to the other offenders (*i.e.*, if the offenders who are given the exploit information are willing to pay high prices, then that sends strong signals to the other offenders and may convince the others to also pay, even though they have not yet received the exploit information).

Ackowledgments The work was supported by the Cyber Security Cooperative Research Centre whose activities are partially funded by the Australian Government's Cooperative Research Centres Programme.

References

1. Algarni, A., & Malaiya, Y. (2014). Software vulnerability markets: Discoverers and buyers. *International Journal of Computer, Information Science and Engineering, 8*(3), 482–484.
2. Bilge, L., & Dumitras, T. (2012). Before we knew it: An empirical study of zero-day attacks in the real world. In *Proc. of 2012 ACM Conf. on Computer and Communications Security, ACM, New York, NY, USA, CCS '12* (pp. 833–844). https://doi.org/10.1145/2382196.2382284.
3. Brams, S. J., Jones, M. A., & Klamler, C. (2007). Better ways to cut a cake - revisited. In Brams, S., Pruhs, K., Woeginger, G. (Eds) *Fair Division, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, Dagstuhl, Germany*, no. 07261 in Dagstuhl Seminar Proceedings.
4. Chen, Y., Lai, J. K., Parkes, D. C., & Procaccia, A. D. (2013). Truth, justice, and cake cutting. *Games and Economic Behavior, 77*(1), 284–297. https://doi.org/10.1016/j.geb.2012.10.009.
5. Duetting, P., Feng, Z., Narasimhan, H., Parkes, D., & Ravindranath, S. S. (2019). Optimal auctions through deep learning. In Chaudhuri, K., Salakhutdinov, R. (Eds) *Proceedings of the 36th International Conference on Machine Learning, PMLR, Long Beach, California, USA, Proceedings of Machine Learning Research* (Vol. 97, pp. 1706–1715).
6. Egelman, S., Herley, C., & van Oorschot, P. C. (2013). Markets for zero-day exploits: Ethics and implications. In *Proceedings of the 2013 New Security Paradigms Workshop, Association for Computing Machinery, New York, NY, USA, NSPW '13* (p. 41–46), https://doi.org/10.1145/2535813.2535818.
7. Emek, Y., Feldman, M., Gamzu, I., PaesLeme, R., & Tennenholtz, M. (2014). Signaling schemes for revenue maximization. *ACM Transactions on Economics and Computation*. https://doi.org/10.1145/2594564.
8. Fisher, D. (2015). Vupen founder launches new zero-day acquisition firm zerodium. July 24, 2015 online: https://threatpost.com/vupen-launches-new-zero-day-acquisition-firm-zerodium/113933/.
9. Greenberg, A. (2012). Shopping for zero-days: A price list for hackers' secret software exploits. March 23, 2012 online: http://www.forbes.com/sites/andygreenberg/2012/03/23/shopping-for-zero-days-an-price-list-for-hackers-secret-software-exploits/.
10. Guo, M., & Conitzer, V. (2010). Computationally feasible automated mechanism design: General approach and case studies. In Fox, M., Poole, D. (Eds) *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, AAAI Press. http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1868
11. Guo, M., & Deligkas, A. (2013). Revenue maximization via hiding item attributes. In: Rossi, F. (Ed) *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013, IJCAI/AAAI* (pp. 157–163). http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6909.
12. Guo, M., Deligkas, A., & Savani, R. (2014). Increasing VCG revenue by decreasing the quality of items. In Brodley, C. E., Stone, P. (Eds) *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada* (pp. 705–711). AAAI Press., http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8186
13. Lavi, R., Ahuva Mu'alem, & Nisan, N. (2003). Towards a characterization of truthful combinatorial auctions. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings* (pp. 574–583).
14. Manisha, P., Jawahar, C. V., & Gujar, S. (2018). Learning optimal redistribution mechanisms through neural networks. In: André, E., Koenig, S., Dastani, M., Sukthankar, G. (Eds) *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018, International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM* (pp. 345–353).
15. Myerson, R. B. (1981). Optimal auction design. *Mathematics of Operations Research, 6*(1), 58–73.
16. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.
17. Projects, T. C. (2015). Severity guidelines for security issues. Retrieved September 15, 2015 online: https://www.chromium.org/developers/severity-guidelines.
18. Sandholm, T., & Likhodedov, A. (2015). Automated design of revenue-maximizing combinatorial auctions. *Operations Research, 63*(5), 1000–1025.
19. Shen, W., Tang, P., & Zuo, S. (2019). Automated mechanism design via neural networks. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, AAMAS '19* (pp. 215–223).

20. Wang, G., Guo, R., Sakurai, Y., Babar, A., & Guo, M. (2020). Mechanism design for public projects via neural networks. arXiv:2002.11382