Optimizing the Sequence of Vulnerability Scanning Injections

Koichi Funaya
NEC Labs Europe
Heidelberg, Germany
koichi.funaya@neclab.eu

Samir Bajaj
NEC Technologies India
Noida, India
samir.bajaj@india.nec.com

Kumar Sharad
NEC Labs Europe
Heidelberg, Germany
kumar.sharad@neclab.eu

Alok Srivastava
NEC Technologies India
Noida, India
alok.srivastava@india.nec.com

***Abstract*— Security Operation Centers face human resource bottleneck in scaling the operations. We attempt to address the issue by developing a framework to prioritize the operational actions, especially in vulnerability scanning and penetration testing tasks. More specifically, we prioritized scanning injections in the order of expected rewards, calculated with the combination of priors and injection-to-injection similarity measures. The framework is shown to reduce the total number of actions while maintaining the amount of vulnerabilities revealed.**

I. INTRODUCTION

SOCs (Security Operation Centers) deal with increasing load and complexity in preparing to defend enterprise IT systems from cyber security threats, thus constantly facing cost-resource trade-offs. We especially look to the workload on security investigation tasks, and reduce its cost by introducing prioritization and automation.

A. *The Objective*

Here in this paper, we consider a system vulnerability scanning scenario, in which we sequentially send a batch of scanning injections to find potential vulnerabilities. The research objective is to reduce the number of total queries to be submitted to the system, thus to reduce the workload, while fulfilling the task of revealing vulnerabilities.

B. *Related work*

Machine Learning techniques are applied to system vulnerability analysis in the past. Most of the successful methods first extract pre-defined static / dynamic features from network protocol (e.g. http) headers and payloads ([1]), possibly with graphical relationships among them ([2]). Then they feed these features to ML methods. We also took this approach of using pre-defined features.

To select the injections, several approaches are investigated. Of them, one is to use a deterministic approach where actions and assets are modeled in the PDDL language, and are used to generate attack plans ([3]). But deterministic approaches miss an important element of information, i.e. the probability of succeeding in the attack plans, thus to prioritize the actions.

One of the approaches to incorporate uncertainty is based on partially observable Markov decision processes (POMDP) ([4], [5]) in which the information gathering was modeled as an integral part of the planning problem. However this solution has difficulty in scaling to large real-life networks.

Another approach is to assign success probability to each of the actions, and prioritize actions in the order of high success probability ([6]). We took this approach and extended further, by using matrix factorization based similarity measure ([7]), and applying to the selection of the injection batches.

II. SELECTING INVESTIGATIVE ACTIONS

The proposed investigation mechanism consist of the following steps.

1) ***Train models offline**: from the training samples, learn the priors, clusters, classifier and similarity measures.*
2) ***Scan & detect**: send injections in a batch, convert both the injections and their responses into feature vectors, and infer the vulnerability using a trained classifier.*
3) ***Select injections**: select the next batch of injections which as per model are highly likely to attack, and repeat the cycle until we exhaust the selection.*

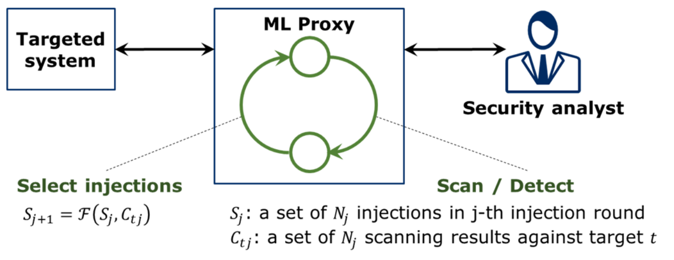
Targeted system
ML Proxy
Security analyst
Select injections
$S_{j+1} = \mathcal{F}(S_j, C_{tj})$
Scan / Detect
S_j: a set of N_j injections in j-th injection round
C_{tj}: a set of N_j scanning results against target t

Figure 1 Solution Concept

Below we explain the actual steps in the solution.

A. *Initial injections*

From a pool of predefined vulnerability scanning injections S, we select a set of injections $S_0 \subset S, |S_0| = N_0$.

When selecting the initial set of injections above, we combined two approaches; one is to raise expected attack rate, and another is to achieve diversity in the injection types. The former is achieved by using the statistical priors calculated from the past scanning results of other web pages.

B. *Request and response feature extraction*

We embed each of these injections s into an http query as payload, send it to the targeted web page t and receive a response with payload $r(t,s)$.

For the tasks of scanning web vulnerabilities and selecting injections, we take contents and associated metrics from the injections and responses, then convert into vector of N_f feature values $U(x) = [u_l(x)]^T, l = 1, \ldots, N_f$. In the case of R-XSS, x is the combination of the injection s and its response, i.e. $x = [s, r(t,s)]$. Feature values in this case are the presence of Input Validation and Sanitization patterns ([1]).

C. *Vulnerability scanning*

For the vulnerability detection, we feed the above mentioned feature vectors to a machine learning based classification algorithm. More specifically, from the N_j scanning injections we get equal number of binary values as a result of vulnerability scan: $C_{tj} = \{c(t,s)\}, s \in S_j$. We used SVM approach for its prediction accuracy and robustness.

D. *Selecting the injections*

Based on the result as described above, we now select the next batch of injections which have expected success rate above certain threshold. To calculate the expected success rate, we took an approach based on matrix factorization ([7]).

In this approach, the expected success rate of the injection s against the target t is described as the product of vector representations of the injections (q_s) and targets (p_t):

$$\hat{c}(t,s) = q_s^T p_t, \quad s \in S, t \in T \qquad (1)$$

By optimizing the factors, we derive:

$$\hat{c}(t,s) = \sum_{s' \in S_j} \rho_{ss'}^t c(t,s') \qquad (2)$$

In the above equation, the similarity measures $\rho_{ss'}^t$ indicate the similarity in scanning results between injections s and s', when scanning the target t. In reality $\rho_{ss'}^t$ is substituted with $\rho_{ss'}$ which is trained offline with training sample web pages.

Then we select the next batch of injection S_{j+1} as below.

$$S_{j+1} = \{s | \hat{c}(t,s) \geq threshold, \quad s \in S \,/\, \{S_0, \ldots, S_j\}\} \qquad (3)$$

We repeat steps B-C-D till no new set of injections are suggested by the model.

III. Experiments

We tested two key components of the model i.e. *Scan & Detect*, and *Select Injections*, individually on open-source test web applications which are deliberately designed for security testing. To prepare dataset, we scanned vulnerabilities against cross-site scripting (R-XSS) attacks, taking injection patterns from OWASP ZAP (https://www.zaproxy.org/), producing 44,000+ logs of http request-response pairs.

The key aim of research is to significantly minimize number of injections and ensure acceptable vulnerability detection accuracy. Hence, in the experiment we controlled the hyper-parameters comprising the number of initial injections and threshold similarity measures $\rho_{ss'}^t$ in Eq. (2).

As per Table 1, if we set the hyper-parameters at 31 and 1.5, the resultant model best balances between dual objectives of High *Recall* and High *Scanning Budget Reduction*.

Hyper-Parameters		*Outcome*	
# of Initial Injections	Threshold Similarity Measure	Average Recall	Scanning Budget Reduction
31	1	98.69%	54,82%
31	1.5	97.87%	61,33%
31	2	97.43%	64,52%
44	1	98.35%	52,32%
44	1.5	97.85%	56,54%
44	2	97.55%	57,74%

Table 1 Injection Selection Model Experimentation Result

Consequently, the integrated ML Proxy solution minimizes on an average number of injected queries by 61.3% and identifies 89.3% of all vulnerabilities with 71.0% precision.

IV. Conclusion and Next Steps

The framework for selecting the sequence of injection batches provides a significant reduction to the vulnerability scanning budget while maintaing the level of vulnerability revelation.

For the next steps, current model could be extended to provide an operator with a list of injections that potentially indicate False Negative outcomes. It should also be worth extending the method to cover APT investigations, in which case the investigation sequence is possibly triggered by anomaly detection, and select subsequent services on adjacent nodes in the network to be investigated.

References

[1] L. K. Shar, L. C. Briand, and H. B. K. Tan, "Web Application Vulnerability Prediction Using Hybrid Program Analysis and Machine Learning", IEEE Transactions on Dependable and Computing, Nov/Dec 2015

[2] D. Balzarotti, M. Cova, V. Felmetsger, N. Jovanovic, E. Kirda, C. Kruegel, and G. Vigna, "Saner: Composing Static and Dynamic Analysis to Validate Sanitization inWeb Applications", 2008 IEEE Symposium on Security and Privacy

[3] M. Boddy, J. Gohde, T. Haigh and S. Harp, "Course of Action Generation for Cyber Security Using Classical Planning", ICAPS 2005

[4] C. Sarraute, O. Buffet, and J. Hoffmann, " POMDPs Make Better Hackers: Accounting for Uncertainty in Penetration Testing", AAAI 2012

[5] J. Hoffmann, "Simulated Penetration Testing: From Dijkstra to Turing Test++", Invited paper in Proceeding ICAPS'15

[6] C. Sarraute, G. Richarte, and J. L. Obes, "An Algorithm to Find Optimal Attack Paths in Nondeterministic Scenarios", in AISec 2011

[7] Y. Koren and R. Bell, "Advances in Collaborative Filtering", ICDM 2