中国科学院软件研究所学术年会'2021
暨计算机科学国家重点实验室开放周

学术论文

ReDoSHunter: A Combined Static and Dynamic Approach for Regular Expression DoS Detection

ReDoSHunter:一种动静态结合的ReDoS检测算法

Yeting Li, Zixuan Chen, Jialun Cao, Zhiwu Xu, Qiancheng Peng, Haiming Chen, Liyuan Chen, Shing-ChiCheung

The 30th USENIX Security Symposium (USENIX Security 2021)

Yeting Li, 15801685206, liyt@ios.ac.cn

Motivation

Regular expression Denial of Service (ReDoS) poses a pervasive and serious security threat.

- Existing detection approaches mainly fall into two categories: **static and dynamic** analysis. However, they all suffer from either **poor precision or poor recall** in the detection of vulnerable regexes.
- ReDoS-vulnerable regex contain **more than one vulnerability** in reality.

This motivates the need for a ReDoS detection approach that can detect multiple vulnerabilities in a regex with high precision and recall .

Challenges

Reach both high precision and high recall is still an open problem.

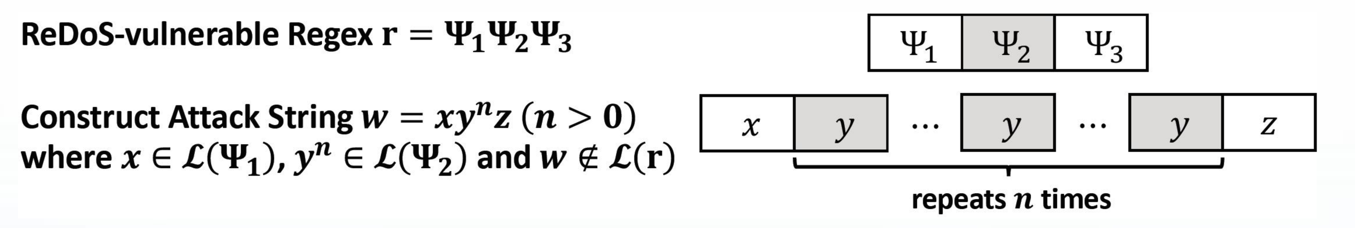
Existing static work with the highest recall (36.70%) turns out to result in only 57.96% precision. While the dynamic work with 100% precision, results in only 1.82% recall. The huge trade-off on precision and recall limits the usefulness of these approaches.

53.7% of ReDoS-vulnerable regexes containing more than one vulnerability.

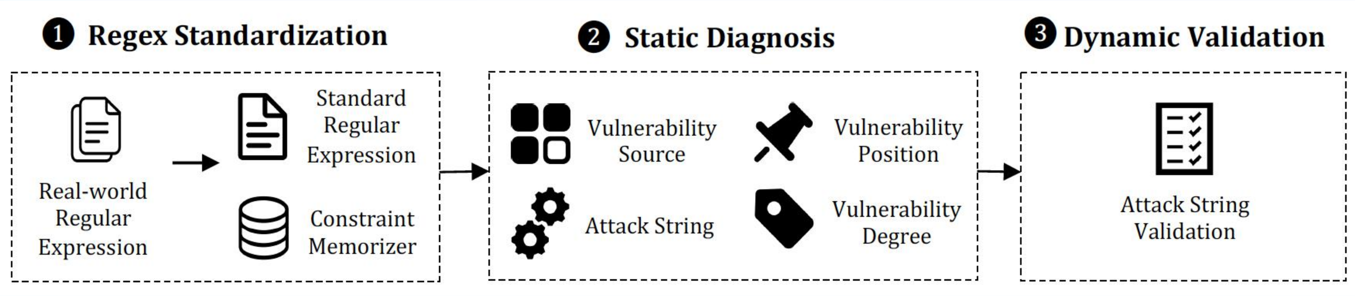
Existing works can hardly locate the root cause of a ReDoS-vulnerability. Even the root cause of the vulnerability can be located, they can only detect one vulnerability.

Approach

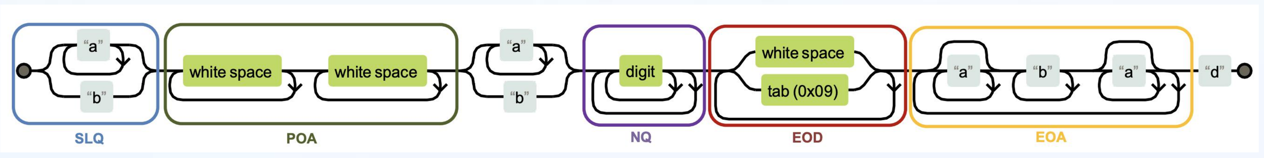
A regex r is ReDoS-vulnerable iff there exists a string w such that the regex on a backtracking regex engine has a super-linear behavior. Such strings are often called attack strings.



We propose ReDoSHunter, a ReDoS-vulnerable regex detection framework which can pinpoint multiple root causes of vulnerabilities and generate attack triggering strings.



ReDoSHunter consists of three key components.



Algorithm 1: ReDoSHunter

Input: a regex α
Output: *true*, a diagnostic information list Γ if α is ReDoS-vulnerable or *false* otherwise

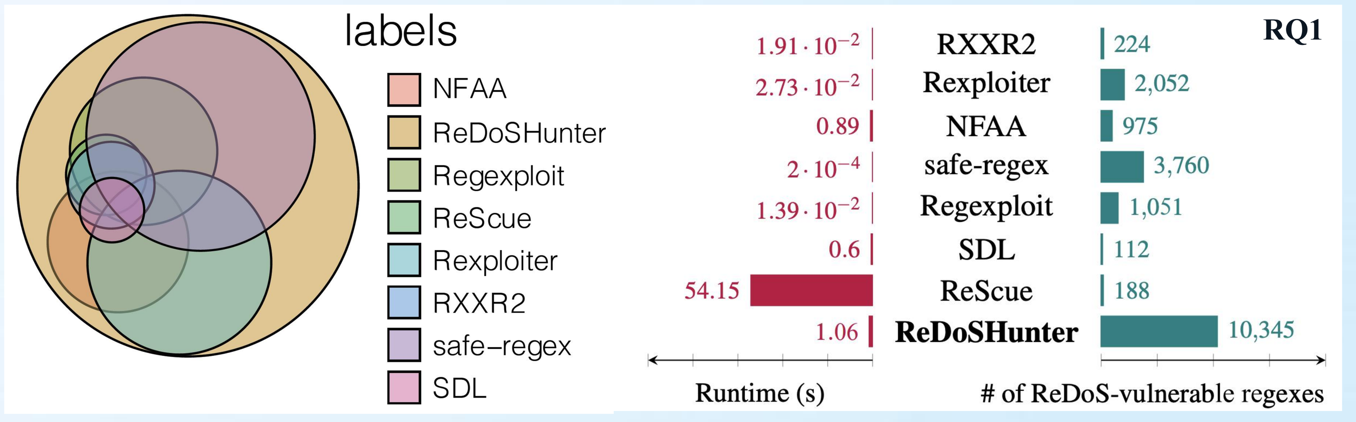
```
β, M ← TransRE(α);
Γ_NQ ← CheckNQ(β, M);
Γ_EOD ← CheckEOD(β, M);
Γ_EOA ← CheckEOA(β, M);
Γ_POA ← CheckPOA(β, M);
Γ_SLQ ← CheckSLQ(β, M);
Γ ← Γ_NQ ∪ Γ_EOD ∪ Γ_EOA ∪ Γ_POA ∪ Γ_SLQ;
if |Γ| = 0 then return false;
foreach info (vulDeg, vulSrc, vulPos, atkStr) ∈ Γ do
    if verifyAtk(α, atkStr, vulDeg) = false then
        delete info (vulDeg, vulSrc, vulPos, atkStr) from Γ;
if |Γ| > 0 then return true, Γ;
else return false;
```

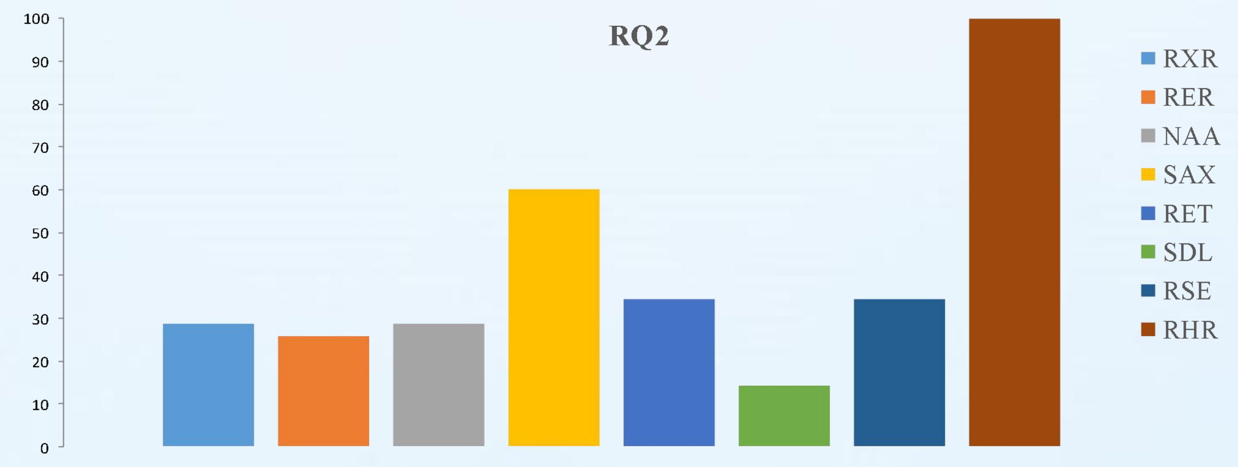
We introduce five ReDoS patterns (NQ, EOD, EOA, POA SLQ) that are identified from our massive investigation and analysis.The vulnerability candidates detected are then dynamically validated such that only the true vulnerabilities are reported.

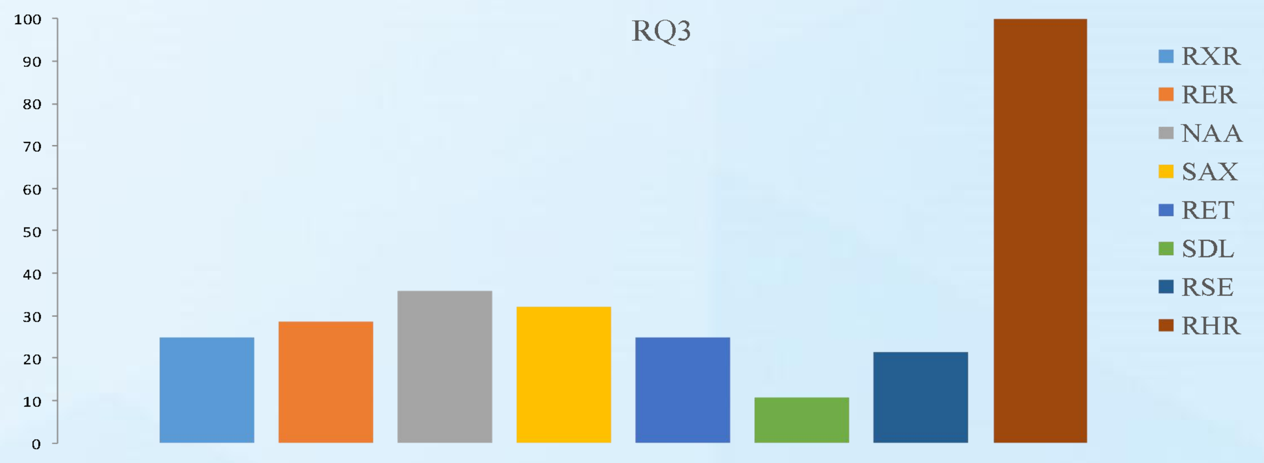
Evaluation

RQ1. How is the effectiveness and efficiency of ReDoSHunter on large-scale regex sets?
RQ2. How is the effectiveness of ReDoSHunter on identifying known vulnerabilities?
RQ3. How is the effectiveness of ReDoSHunter on exploring unknown vulnerabilities?

We evaluated ReDoSHunter on three types of datasets (i.e., *regex sets*, *known ReDoS-vulnerabilities*, and *intensivelytested projects*).We compared ReDoSHunter with seven approaches (i.e., *RXXR2*, *Rexploiter*, *NFAA*, *safe-regex*, *Regexploit*, *SDL* and *ReScue*).







Summary to RQ1: ReDoSHunter can achieve **100% precision** and **100% recall** against four tested regex engines. ReDoSHunter achieved a remarkable balance between effectiveness and efficiency empowered by the advantages of both static and dynamic methods.
Summary to RQ2: ReDoSHunter can identify all **35 ReDoS-related CVEs**, compared with the best work identifying only over 60.00% of them.
Summary to RQ3: ReDoSHunter is capable to be applied to exploring unknown ReDoS-vulnerabilities in the wild. Among 28 identified vulnerabilities, **26 of them were assigned CVEs or 2 of them were fixed by maintainers.**

Conclusion

We proposed **ReDoSHunter, a ReDoS-vulnerable regex detection framework** that can pinpoint multiple root causes of vulnerabilities, diagnose vulnerability locations, assess vulnerability degrees and generate attack-triggering strings.