ISCAS 中国科学院软件研究所学术年会'2022
暨计算机科学国家重点实验室开放周

学术论文

RegexScalpel: Regular Expression Denial of Service (ReDoS) Defense by Localize-and-Fix
RegexScalpel: 一种基于定位-修复的正则表达式拒绝服务(ReDoS)防御方法

Yeting Li, Yecheng Sun, Zhiwu Xu, Jialun Cao, Yuekang Li, Rongchen Li, Haiming Chen, Shing-Chi Cheung, Yang Liu, Yang Xiao
The 31st USENIX Security Symposium (USENIX Security 2022)
Yeting Li, 15801685206, liyt@ios.ac.cn

Motivition

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

There are three types of strategies to defense ReDoS attacks: (i) regex engine substitution, (ii) input length restriction, (iii) regex repair.

- Regex engine substitution omits extended features and brings semantic differences or incompatibilities.
- Input length restriction faces a dilemma known as "Goldilocks problem".
- Regex repair can greatly mitigate their vulnerabilities and is the most common defense strategy. But the existing regex repair works cannot promise the repaired regex preserves semantics or is invulnerable to ReDoS attacks.

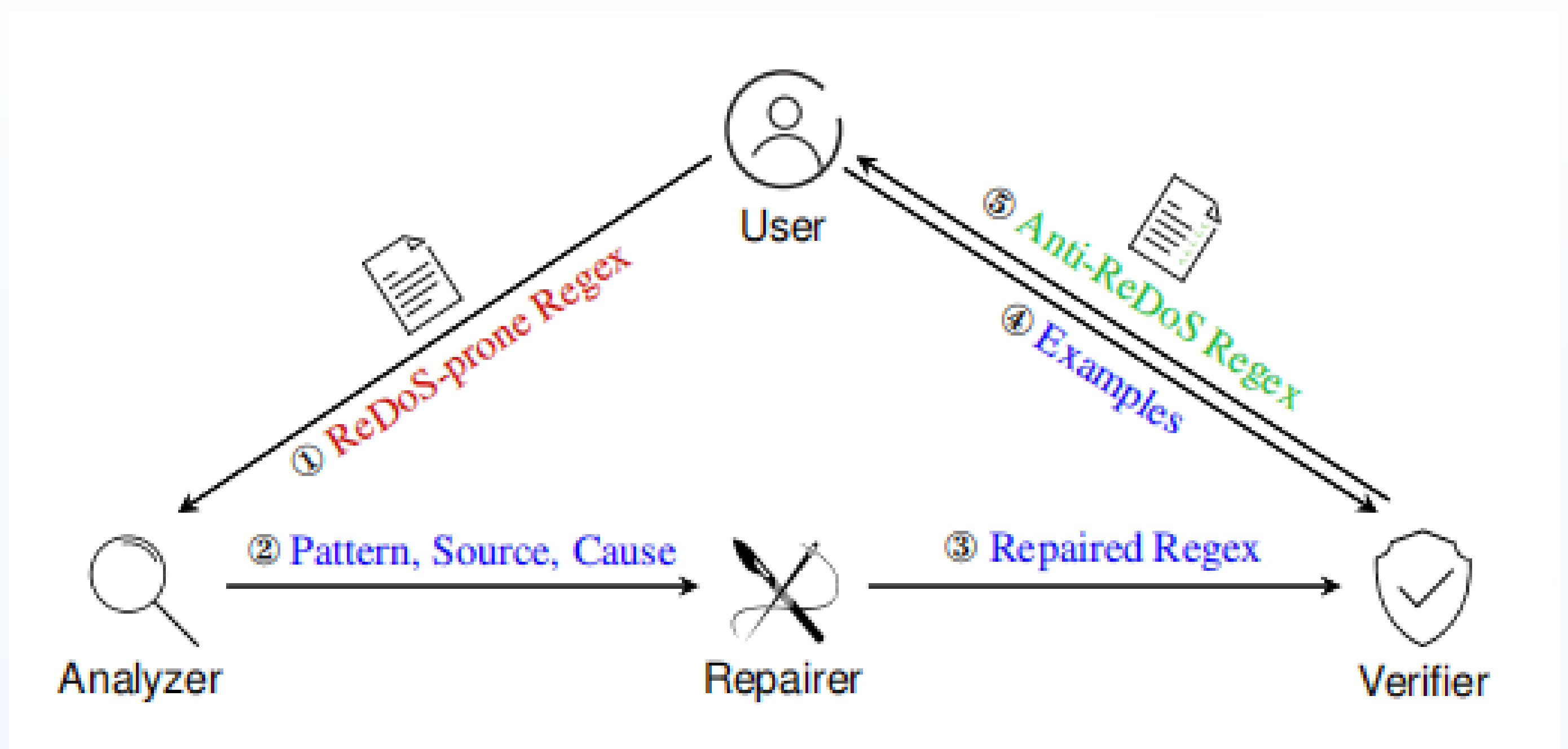
This motivates the need for a regex repair approach that can promise semantics preservation and invulnerability.

Challenges

- Difficulty of localizing the vulnerabilities and obtaining the information necessary for the repair.
- Difficulty of promising the repaired regex preserves semantics.
- Difficulty of promising the repaired regex is invulnerable to ReDoS attacks, that is, no vulnerabilities will be newly introduced and all vulnerabilities in the regex will be repaired.

Approach

To overcome these challenges, we propose RegexScalpel, a regex ReDoS vulnerability analysis and repair framework based on localize-and-fix.



For four types of vulnerable patterns (i.e. NQ, QOD, QOA, SLQ), RegexScalpel first leverages the fine-grained vulnerability patterns (see one example in the right part), which enables analyzing the information necessary for the repair, to localize the vulnerabilities.

No.	Sub-pattern	Vulnerability Description	Example Regex	Returned Triple
#1	$\mathcal{QOD}_1$	$r=\alpha\{m,n\}$, $\alpha=(r_1\mid\dots\mid r_k)$, and $\exists\ \mathcal{L}(\alpha_1)\cap\mathcal{L}(\alpha_2)\neq\emptyset$, where $\alpha_1=r_{p_1}r_{p_2}\dots r_{p_t}$, $\alpha_2=r_{q_1}r_{q_2}\dots r_{q_s}$, $1\le t,s\le k$, $1\le i\le t$, $1\le j\le s$, $1\le p_i,q_j\le k$, and $p_1\neq q_1$	δ_4 = `^(\d\.|\.\d|\d)+$` ([14])	($\mathcal{QOD}_1$, `(\d\.|\.\d|\d)+`, [`\d\.\d`, `\d\.\d`])
#2	$\mathcal{QOD}_2$	$r=\alpha\{m,n\}$, $\alpha=(r_1\mid\dots\mid r_k)$, and $\exists\ \mathcal{L}(r_p)\cap\mathcal{L}(\alpha_1)\neq\emptyset$, $\mathcal{L}(r_l)\cap\mathcal{L}(r_p)=\emptyset$, where $\alpha_1=r_{q_1}r_{q_2}\dots r_{q_s}$, $1\le i\le k$, $r_i=r_lr_p$, r_l is not nullable, $1\le s\le k$, $1\le j\le s$, $1\le q_j\le k$	δ_5=`(?:[\w-]|\$[-\w]+|#\{\$[-\w]+\})+(?=\s*:)` (CVE-2021-23341)	($\mathcal{QOD}_2$, `(?:[\w-]|\$[-\w]+|#\{\$[-\w]+\})+`, [`[\w-]`, `\$[-\w]+`])

RegexScalpel then aims at fixing the pathological sub-regexes according to the predefined repair patterns and the localized vulnerability information. Our repair patterns ensure that the repair regexes preserve semantics, and our iterative repair method also keeps out vulnerabilities of the repaired regexes.

No.	Repair Pattern	No.	Repair Pattern	No.	Repair Pattern
τ_4	$\dfrac{\mathcal{L}(r)=\mathcal{L}((r_1\mid\dots\mid \cancel{r_k}\mid\dots\mid r_k)\{m,n\})}{\alpha\Longrightarrow(r_1\mid\dots\mid \cancel{r_k}\mid\dots\mid r_k)}\ (\mathcal{QOD}_1)$	τ_5	$\dfrac{t>1}{r_{p_1}\Longrightarrow r_{p_1}(?!\Phi(r_{p_2}))}\ (\mathcal{QOD}_1)$	τ_6	$\dfrac{s>1}{r_{q_1}\Longrightarrow r_{q_1}(?!\Phi(r_{q_2}))}\ (\mathcal{QOD}_{\{1,2\}})$
τ_7	$\dfrac{}{r_{p_1}\Longrightarrow(?!\Phi(\alpha_2))r_{p_1}}\ (\mathcal{QOD}_1)$	τ_8	$\dfrac{}{r_{q_1}\Longrightarrow(?!\Phi(\alpha_1))r_{q_1}}\ (\mathcal{QOD}_1)$	τ_9	$\dfrac{scs(\alpha_1)=true,\quad \Sigma_{\alpha_1}\setminus\text{first}(r_{q_1})\neq\emptyset}{\alpha_1\Longrightarrow\Theta(\Sigma_{\alpha_1}\setminus\text{first}(r_{q_1}))}\ (\mathcal{QOD}_1)$
τ_{10}	$\dfrac{scs(\alpha_2)=true,\quad \Sigma_{\alpha_2}\setminus\text{first}(r_{p_1})\neq\emptyset}{\alpha_2\Longrightarrow\Theta(\Sigma_{\alpha_2}\setminus\text{first}(r_{p_1}))}\ (\mathcal{QOD}_1)$	τ_{11}	$\dfrac{}{r_p\Longrightarrow r_p(?<!\Phi(\alpha_1))}\ (\mathcal{QOD}_2)$	τ_{12}	$\dfrac{}{r_{q_1}\Longrightarrow(?!\Phi(r_p))r_{q_1}}\ (\mathcal{QOD}_2)$
τ_{13}	$\dfrac{scs(\alpha_1)=true,\quad \Sigma_{\alpha_1}\setminus\text{first}(r_p)\neq\emptyset}{\alpha_1\Longrightarrow\Theta(\Sigma_{r_{q_1}}\setminus\text{first}(r_p))}\ (\mathcal{QOD}_2)$	τ_{14}	$\dfrac{r_p=r_u\{m_u,n_u\},\quad scs(r_u)=true,\quad \Sigma_{r_u}\setminus\text{first}(r_{q_1})\neq\emptyset}{r_p\Longrightarrow\Theta(\Sigma_{r_u}\setminus\text{first}(r_{q_1}))}\ (\mathcal{QOD}_2)$		

RegexScalpel finally determines whether the repaired regexes are ReDoS-invulnerable and whether it can pass all given test cases.

Evaluation

RQ1. Can RegexScalpel outperform state-of-the-art regex defense techniques?
RQ2. Can RegexScalpel outperform handcrafted defense actions?
RQ3. Can RegexScalpel detect new vulnerabilities and synthesize useful repairs to maintainers?
RQ4. Can RegexScalpel synthesize repaired regexes preserving the semantics of the original ones?

We evaluate the effectiveness of RegexScalpel on ReDoS-vulnerable regexes from the SOLA-DA benchmark and ReDoS-related CVEs, compared with five state-of-the-art ReDoS defense techniques varying from regex engine substitution, input length limit and regex repair.

Summary to RQ1: RegexScalpel can effectively defend 98.88% of vulnerable regexes, compared with 21.20% achieved by the best work.
Summary to RQ2: Among the 413 repaired vulnerable regexes handcrafted by the maintainers, only 319 (77.23%) are ReDoS free. RegexScalpel successfully repairs 409 (99.03%) of the 413 regexes.
Summary to RQ3: RegexScalpel helped to repair 16 ReDoS vulnerabilities in the ten real-world projects and got confirmed by the maintainers, resulting in 8 confirmed CVEs.
Summary to RQ4: RegexScalpel can synthesize repaired regexes preserving the semantics of the original ones and keeping the semantics as close as possible to the original ones.

Table 10: Success Defense Rate Across Automated Tools.

Tool	SOLA-DA	CVE	Total
RE2	18 (52.94%)	35 (8.45%)	53 (11.83%)
LLI(100)	22 (64.71%)	45 (10.87%)	67 (14.96%)
LLI(500)	26 (76.47%)	18 (4.35%)	44 (9.82%)
LLI(5000)	26 (76.47%)	19 (4.59%)	45 (10.04%)
FlashRegex	4 (11.76%)	91 (21.98%)	95 (21.20%)
RegexScalpel	**33 (97.06%)**	**410 (99.03%)**	**443 (98.88%)**
#Regex	34	414	448

Table 14: Success Defense Rate of the Repairs by Maintainers and RegexScalpel.

	SOLA-DA	CVE	Total
Manual Repair	14 (66.67%)	305 (77.81%)	319 (77.23%)
RegexScalpel	**21 (100%)**	**388 (98.98%)**	**409 (99.03%)**
#Regex	21	392	413

Table 17: Demographics of New Vulnerabilities Repaired by RegexScalpel.

No.	Project	Disclosure Date	CVE ID	#Vuln. Regex
#1	Python	Jan 30th, 2021	CVE-2021-3733	1
#2	NLTK	Sep 5th, 2020	–	1
#3	pylint	Sep 3rd, 2020	–	2
#4	mpmath	Oct 8th, 2021	CVE-2021-29063	1
#5	browserslist	Apr 28th, 2021	CVE-2021-23364	6
#6	code-server	Sep 17th, 2021	CVE-2021-3810	1
#7	ansi-regex	Sep 12th, 2021	CVE-2021-3807	1
#8	nth-check	Sep 17th, 2021	CVE-2021-3803	1
#9	nodejs-tmpl	Sep 15, 2021	CVE-2021-3777	1
#10	jspdf	Feb 12th, 2021	CVE-2021-23353	1
			Total	16

Conclusion

We propose RegexScalpel, which can defend ReDoS attacks by automatically localizing and repairing ReDoS-vulnerable regexes. RegexScalpel is the first approach to localize and fix the vulnerable regexes considering comprehensive and fine-grained types of vulnerable causes.