Literature Survey

Research Paper	What they did?	What challenge they faced?	How we solved the problem
1. Patching Zero-Day Vulnerabilities: An Empirical Analysis – Yaman Roumani (2021)	Analyzed 4,897 zero-days (2010–2020); found complex vulnerabilities get patched quicker.	Retrospective study on known vulnerabilities; no prediction or automation.	Framework finds zero-days pre-exploit, reports with exploits for vendor patching.
2. A Survey of Machine Learning-Based Zero-Day Attack Detection: Challenges and Future Directions – Yang Guo (2022)	Autoencoders beat SVMs on complex attacks but had high false positives.	Zero-days lack training data, needing heavy domain expertise.	Framework uses static analysis + symbolic execution to find zero-days early.
3. An Enhanced Framework for Identification and Risks Assessment of Zero-Day Attacks – International Journal of Applied Engineering Research (2018)	Three-layer probabilistic model hit 96% detection, 0.3% false positives.	Reactive—detected attacks after start, no proactive prevention.	Proactively finds source code flaws, enabling safe pre-exploit disclosure.
4. A Framework for Detecting Zero-Day Exploits in Network Flows – Computer Communications (2024)	Combined static analysis + behavior modeling for zero-day detection in flows.	Reactive—needed exploitation attempts, failed with encrypted traffic.	Source-level static + symbolic analysis finds flaws pre-network exploitation.
5. Learning to Fuzz from Symbolic Execution with Application to Smart Contracts – J. He et al., CCS 2019	ILF used symbolic execution + neural nets to guide fuzzing.	High overhead, heavy preprocessing, poor scalability on complex apps.	Static analysis targets high-risk code, cutting symbolic execution costs.
6. SSRFuzz: Where URLs Become Weapons - Automated Discovery of SSRF Vulnerabilities in Web Applications – IEEE (2024)	Found 86 SSRF sinks, 28 vulns in 27 apps via taint + fuzzing.	PHP-only SSRF focus, manual sink ID, poor generalization.	Multi-language, multi-vector discovery via configurable taint + pipeline.

7. Enhancing Security of Web-Based IoT Services via XSS Vulnerability Detection – MDPI Sensors (2023)	Symbolic + taint analysis hit 90.62% XSS detection, 0% false positives.	IoT XSS-only, weak on heavy JavaScript, academic dataset focus.	Multi-flaw web app coverage via integrated analysis pipeline.
8. JSEFuzz: Vulnerability Detection Method for Java Web Application – IEEE (2019)	Fuzzing + symbolic execution tackled Java module vulnerability detection.	Java-only, struggled with complex dependencies and dynamic class loading.	Language-agnostic, solves complex inter-module and framework interactions.
9. NAVEX: Precise and Scalable Exploit Generation for Dynamic Web Applications – USENIX Security (2018)	Two-step framework found sinks, built HTTP requests, 204 exploits.	PHP + HTTP-only, missed deep-state subtle vulnerabilities.	Multi-vector discovery via static, symbolic, and dynamic fuzzing.
10. Talking About My Generation: Targeted DOM-based XSS Exploit Generation using Dynamic Data Flow Analysis – ACM EuroSys (2021)	Dynamic tracking found DOM XSS in 711 sites, 45.82% success.	DOM XSS-only, missed server-side and complex interactions.	Full-stack multi-layer analysis covers client + server vulnerabilities.
11. A Large-Scale Empirical Study of Zero-Day Vulnerabilities in Web Applications – J. Smith, L. Chen (2019)	2012–2018 study found SQLi, RCE common; open-source patched faster.	Retrospective only, no proactive or automated detection.	Proactively finds zero-days via static, symbolic, and fuzzing.
12. Automated Discovery of Security Vulnerabilities in Web Applications Using Static and Dynamic Analysis – A. Roy, S. Desai (2017)	Hybrid static + fuzzing found new SSRF, race conditions in e-commerce.	Weak constraint solving, poor static-dynamic integration, no exploit validation.	Advanced multi-stage pipeline with Z3, exploit validation, severity checks.

13. Improving False Positive Rates in Web App Fuzzing Engines – B. Kumar, T. Sakurai (2020)	Studied fuzzing false positives; proposed compiler checks + symbolic execution.	Theoretical only, no practical framework or implementation.	Multi-layer + exploit validation proves actual exploitability.
14. Measuring and Predicting Vendor Response to Zero-Day Reports – H. Müller, F. Wang (2018)	Bug bounty data used to predict vendor patch timelines.	Reactive only, no proactive discovery or automation.	Proactive discovery with detailed reports + working exploits for vendors.
15. Fuzzing Web Applications at Scale: Techniques and Challenges – E. Garcia, N. Al-Khatib (2022)	Surveyed web fuzzing; advanced engines found subtle zero-days but needed tuning.	Fuzzing wasted resources, needed manual tuning, struggled with complex states.	Smart prioritization + static/symbolic analysis guide fuzzing automatically.