***Título:*** Fixing Vulnerabilities automatically with Liters. ***Objetivo:*** Investigar o uso do Liters (ESLint) como ferramenta de teste estático para eliminar bugs, vulnerabilidades e melhorar a qualidade do software. ***Método:*** A aplicação do ESLint para solucionar problemas de vulnerabilidade e conhecer mais sobre esse tipo de ferramenta para melhorar o processo de teste estático, assim os profissionais conseguirem testar e aumentar a qualidade de software, a ideia é trazer os benefícios e desvantagens desse tipo de ferramenta. ***Resultados:*** O estudo trouxe uma série de vantagens sobre essa ferramenta no modo de aplicabilidade nas empresas como pontos fortes e limitações sobre o JSLint, plugins de segurança; Diretrizes de regras; Configuração e Recomendações. Observamos também que os linters devem ser facilmente configuráveis para se adaptar às novas facetas das vulnerabilidades de segurança que podem surgir nas bibliotecas, promovendo o rápido desenvolvimento de regras que abordem tais vulnerabilidades. Demonstramos que o concerto automático de vulnerabilidades é eficaz e acreditamos que os linters têm o potencial de permitir a detecção e correção de vulnerabilidades em bibliotecas antes mesmo que os desenvolvedores da biblioteca produzam um patch para o código vulnerável: um desenvolvedor de biblioteca pode adicionar uma regra para ESLint enquanto trabalhando em um x de longo prazo para a vulnerabilidade. Também observamos que os desenvolvedores de bibliotecas que não usam linters podem apresentar vulnerabilidades conhecidas em seu código. Linters que já têm regras orientadas pela comunidade que tratam de vulnerabilidades tão conhecidas podem, portanto, facilitar o relato delas aos desenvolvedores de bibliotecas. ***Conclusão:*** Neste trabalho, investigamos até que ponto os linters podem ajudar os programadores convencionais a fornecer detecção de vulnerabilidade, com um foco específico em como um linter plugável pode ajudar os programadores JavaScript a escrever código seguro. Descobrimos que o esforço necessário para criar regras ESLint para vulnerabilidades de segurança é bastante baixo, enquanto o impacto da criação de tais regras é potencialmente alto. Isso requer, no entanto, que as regras sejam bem documentadas e projetadas para minimizar falsos positivos.