論　文　の　要　旨

題　目　　Modeling and Detecting Security Vulnerabilities with Static Analysis
（静的解析によるセキュリティ脆弱性のモデリングと検出）

氏　名　　WANG PINGYAN

Security vulnerabilities in software can be exploited by attackers to launch malicious attacks, which in turn can cause security failures. Static analysis is a widely used technique for vulnerability discovery in both academia and industry. While many existing static analysis approaches have been proved to be effective, there remain some challenging open problems. This dissertation focuses on two important problems. The first problem is how to perform analysis on incomplete programs and obtain real-time analysis results during program development, which enables vulnerability detection at an early stage. Another interesting problem is that automated static analysis alone tends to miss some complex and subtle vulnerabilities, thus requiring the incorporation of certain manual security expertise to augment its capabilities.

To enable real-time analysis on incomplete programs, we first present a general framework used in a paradigm known as Human-Machine Pair Programming. The framework employs attack trees to model a given class of vulnerabilities and then crafts patterns for each individual vulnerability. The programmer will be alarmed during coding when patterns match potentially vulnerable code. To identify specifically taint-style vulnerabilities in Human-Machine Pair Programming, we further present two pointer-analysis-based approaches, namely exhaustive pointer analysis and demand-driven pointer analysis. Both the approaches support an incremental pointer analysis and provide points-to information for vulnerability discovery during program development. Our experiment results show that the proposed approaches can detect all the potential vulnerabilities in Securibench Micro with low false positives.

To benefit from both manual audits and automated static analysis, we propose vulnerability nets, a graphical code representation for modeling source code. With a combination of Petri nets, data dependence graphs, and control flow graphs, vulnerability nets explicitly describe the key information of a given program and provide a graphical view for analysts to perform auditing. Our evaluation shows that the proposed approach outperforms the tool SonarQube in generating fewer false negatives when tested in Securibench Micro.