Bridging the Void between Software Engineers and Security Experts

Tayyaba Nafees
University of Abertay Dundee
Kydd Building, Bell St
Dundee DD1 1HG
+447473004011
1405357@abertay.ac.uk

Natalie Coull, Ian Ferguson,
Adam Sampson
University of Abertay Dundee
Kydd Building, Bell St
Dundee DD1 1HG

ABSTRACT
The U.S. Department of Homeland Security (DHS) states that 90 percent of security incidents result from exploits against defects in software [14]. Software developers struggle against these recurring and consistent software errors (vulnerabilities) e.g. buffer overflows and integer overflows, which are exploited by hackers on a daily basis. Despite this, there exists a wealth of cybersecurity domain knowledge with regards to software vulnerabilities and errors held in vulnerability databases (VDBs) such as NVD [15], CWE [6] and CVE [7] and CAPEC [13]. It is clear that these resources are not being utilised effectively to provide knowledge to the software developer due to information overload, lack of techniques to systematically annotate and insufficient analysis of the data relating to these vulnerabilities[1, 9, 17].

The frequency and reoccurrence of recently discovered vulnerabilities undoubtedly proves that malicious hackers can know a lot more about systems than the developers who created them. They can be very effective due to their extensive knowledge sharing [11]. However, the security expert and software developer fail to show efficiency in their knowledge sharing. The problem of frequently recurring software vulnerabilities is very well known, but no standard solution has been universally adopted [3]. There exists a lack of knowledge transfer between the software developer and security expert communities. The root cause is a distinct communication gap between the software developer and security expert [5].

This project aims to transfer such cybersecurity knowledge from the ethical hacking community to the software engineering community by repurposing and disseminating the knowledge already held in VDBs. We propose a methodology for transferring the knowledge. It consists of two parts: Firstly, it extracts and normalizes the VDBs’ information and uses it to define safeguards and injury paths which associate security incidents with their low-level and high-level root causes during the various Software Development Life Cycle (SDLC) phases. Secondly, it aims to use a combination of patterns such as Security Patterns (SP), to fulfil security objectives [4], Software Fault Patterns (SFP, to find common foothold errors) [12, 16] and Attacks Patterns (AP, to set mitigation strategy) [13] to capture the VDBs’ extracted knowledge to make it understandable in a form useful to the software developer. The methodology describes the transformation of information in the VDBs into patterns suitable for incorporation into tools, checklists and training sessions for developers.

The transfer of knowledge from the VDBs in an understandable form like patterns for software developers could aid the handling of security concerns during the development process ([2, 8, 10]. Therefore, it bridges the communication gap between security and SE communities.

This will facilitate the sharing of vital information about common errors and help software developers identify the common security problems associated with the different areas of their work.

The work described forms the basis of the first year of a PhD program in cybersecurity and software engineering. Subsequent years will develop and evaluate the means of creating and disseminating the cybersecurity patterns.

CCS Concepts
• Security and privacy → Software and application security → Software security engineering → Software and its engineering → Software creation and management. • Security and privacy → Systems security → Vulnerability management

Keywords
Vulnerability Database (VDB); Software Fault Pattern (SFP); Security Pattern (SP); Attack Pattern (AP).

References


