Effective Malicious Code Identification

I. Background

As attacks against source code have grown exponentially, malicious code injection has become a major threat to Internet security. Some code may engage in some suspicious and dangerous behaviors in the process of serving users, such as reserving special passwords or keys, connecting to the network without authorization, monitoring screens and keyboards, and hiding processes. These behaviors may damage the availability of computer systems and bring a series of losses and risks to users. The enterprise production, communication and information security are severely challenged

A. *Malicious Code Poisoning Cases*

- 2020.03, malware designed to enumerate and backdoor NetBeans projects. Uses the build process and its resulting artifacts to spread itself: https://bit.ly/3U2msZn (GitHub Security Lab)
- With 20 million + weekly downloads, the NPM library colors has been updated with version 1.4.1. The version was launched with malicious code that crashed upon installation, displaying an American flag: https://bit.ly/3SNGUf1 (GitHub colors.js)
- 2017.08 Xshell Build version 1322 was found and confirmed that there is a backdoor code in nssock2.dll component. Malicitious code will collect host information and send it to DGA domain name, and there are other more malicious function codes

II. Technical Challenges

Efficient and accurate identification of malicious source code based on static behavior analysis: The existing virus scanning engine based on binary cannot detect malicious behavior in source code well. Although more and more related studies are concerned, the detection techniques and effects are different due to language differences. The static behavior analysis of single-file source code and full source code may be missing or inefficient, which cannot meet the security scanning requirements of a large number of software

III. Technical Requirements

- **Technical versatility/flexibility**: Gain insight into the malicious behavior of source code of each language, provide multi-language common detection technologies, and support detection of multiple malicious behavior patterns, including but not limited to backdoor, information leakage, bounce shell, and encryption of malicious code.
- **Detection performance**: Supports automatic filtering and analysis of files related to malicious behavior. Supports

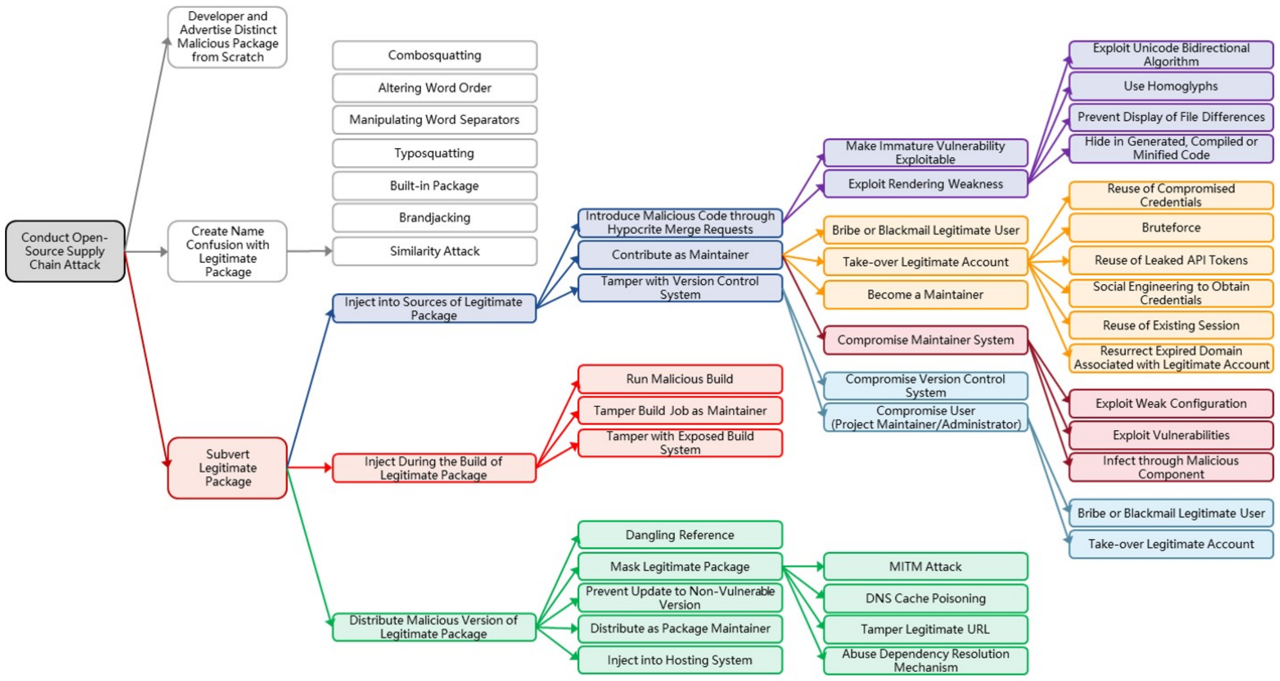
Conduct Open-Source Supply Chain Attack
Developer and Advertise Distinct Malicious Package from Scratch
Create Name Confusion with Legitimate Package
Combosquatting
Altering Word Order
Manipulating Word Separators
Typosquatting
Built-in Package
Brandjacking
Similarity Attack
Subvert Legitimate Package
Inject into Sources of Legitimate Package
Introduce Malicious Code through Hypocrite Merge Requests
Contribute as Maintainer
Tamper with Version Control System
Make Immature Vulnerability Exploitable
Exploit Rendering Weakness
Exploit Unicode Bidirectional Algorithm
Use Homoglyphs
Prevent Display of File Differences
Hide in Generated, Compiled or Minified Code
Bribe or Blackmail Legitimate User
Take-over Legitimate Account
Become a Maintainer
Reuse of Compromised Credentials
Bruteforce
Reuse of Leaked API Tokens
Social Engineering to Obtain Credentials
Reuse of Existing Session
Resurrect Expired Domain Associated with Legitimate Account
Compromise Maintainer System
Exploit Weak Configuration
Exploit Vulnerabilities
Infect through Malicious Component
Compromise Version Control System
Compromise User (Project Maintainer/Administrator)
Bribe or Blackmail Legitimate User
Take-over Legitimate Account
Inject During the Build of Legitimate Package
Run Malicious Build
Tamper Build Job as Maintainer
Tamper with Exposed Build System
Distribute Malicious Version of Legitimate Package
Dangling Reference
Mask Legitimate Package
Prevent Update to Non-Vulnerable Version
Distribute as Package Maintainer
Inject into Hosting System
MITM Attack
DNS Cache Poisoning
Tamper Legitimate URL
Abuse Dependency Resolution Mechanism

Fig. 1. Taxonomy of Attacks on OSS Supply Changes [1]

cross-file malicious behavior detection and improves code analysis efficiency

IV. Related Work

- **Rule scanning**: Malicious rule patterns are manually defined, which has good detection performance. However, the detection accuracy is not ideal.
- **Malicious behavior analysis**: A recent article [2] introduces a method for malicious behavior detection in software packages and successfully identifies over 300 malware packages. In addition, the malicious behavior of single files in different interpreted languages is studied and analyzed, which can improve the detection accuracy of rule scanning.

References

[1] P. Ladisa, H. Plate, M. Martinez and O. Barais, "SoK: Taxonomy of Attacks on Open-Source Software Supply Chains," in 2023 IEEE Symposium on Security and Privacy (SP) pp. 167-184. doi: 10.1109/SP46215.2023.00010.

[2] Towards Measuring Supply Chain Attacks on Package Managers for Interpreted Languages, NDSS2021.