CS503 Paper Review

David Perry
perry74@purdue.edu

An Architectural Approach to Preventing Code Injection Attacks
Ryan Riley, Xuxian Jiang, and Dongyan Xu

1. **Give three technical reasons why this is a good paper.**
   - Other code injection attack defense methods may prevent an attacker from modifying execution flow or prevent code execution in memory pages. However, the defense in this paper attempts to fix the problem by removing the core reason these type of attacks can exist. In von Neumann style architectures there is no separation from memory that is being used as code and memory that is being used as data. As no separation exists it is possible for an attacker to insert data that is later used as code during execution. However, the authors of this paper suggest the use of the Harvard architecture. In this architecture some form of a barrier exists between code and data. This separation ensures that parts of memory considered to be data are imposisble to be treated like code. Therefore, code injection cannot occur as the operating system cannot fetch and execute an instruction that is actually data.

   - Another great strength of this paper is that the defense mechanism it suggests is very practical. The paper recognizes that in order for a security mechanism to be useful it must run on commodity hardware. Therefore, the paper describes an approach that could be implemented with a simple software based patch. To achieve such an architectural change in a system's function without modifying the hardware the authors choose to heavily really on virtualization. This way they were able to achieve the data and code separation key to the Harvard architecture while still running on the backbone of an x86 processor. The practicality of the system was even further supported by the means of implementation. This performance was achieved through the clever exploitation of the existing two TLB's that exist in modern architectures.

   - The last strength of the method proposed in the paper deals with the amount of flexibility the system allows. While most methods that detect a code injection attack simply allow the system to crash the described method allows multiple behaviors. The behaviors described in the paper describe the following three different modes: break, observe, and forensic. During break mode the system will simply crash. The observe mode attempts to allow the code injection attack to continue loggin its behavior. This allows the computer to learn about the attack and would likely be useful in the implementation of honey pots. Finally, the forensic attempts provide deep forensic analysis on the detected attack by starting analysis right before the first injected attack code is executed.

2. **Describe at least one technical weakness or limitation of the paper and propose possible improvement.**
   - The largest weakness of the method described in the paper tends to be the problem that occurs in most systems that heavily leverage the use of virtualization. Since one architecture is being emulated on top of another architecture performance overhead is expected. The authors claim that the performance penalty that occurs as a result of their implementation is around 10 to 20%. While some systems may not be largely effected by such a penalty it is likely that in some systems this overhead would be too much to bare. While this method seems to semantically fix the problem of code injection attacks it may not serve as a universal fix as it may not be viable in some systems.

The review should be limited to one page