SigFree: A Signature-free Buffer Overflow Attack Blocker

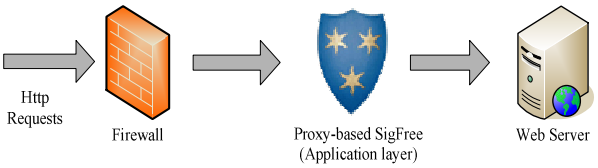
Xinran Wang, Chi-Chun Pan, Peng Liu, Sencun Zhu

Motivation

Buffer overflow attacks typically contain executables whereas legitimate client requests never contain executables in most Internet services such as Web Services, SQL Services, BIND, SNMP, and other remote access services. Based on this observation, SigFree blocks attacks by detecting the presence of code.

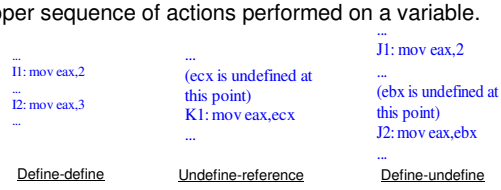
A Proxy based Blocker

SigFree is an application layer blocker that typically stays between a service and the corresponding firewall.



ISA – Scheme 2

Data flow anomaly: During a program execution, an instruction may impact a variable on three different ways: define – when set a value, reference – when referred to; undefine – when not set or set by another undefined variable. A data flow anomaly is caused by an improper sequence of actions performed on a variable.

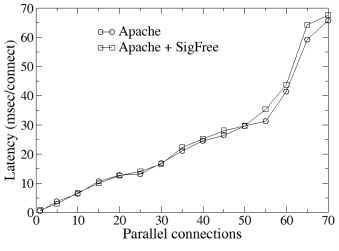


Observation: a random instruction sequence normally is full of data flow anomalies, whereas a real program has few or no data flow anomalies however, due to possible obfuscation, the number of data flow anomalies cannot be directly used to distinguish a program from a random instruction sequence.

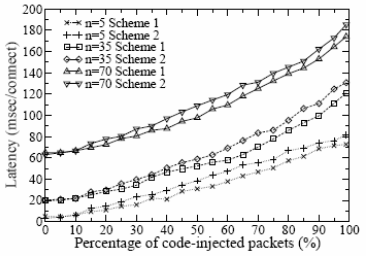
Code Abstraction: based on data flow anomalies, some instructions are "useless", whereas in a real program at least one execution path have a certain number of "useful" instructions, we leverage the detected data flow anomalies to prune useless instructions.

Criteria: After code abstraction , if the number of useful instructions exceeds a threshold, we consider the instruction sequence as a program

Performance



When there are no buffer overflow attacks, the average response time in the system with SigFree is only slightly higher than the system without SigFree.



The average latency increases slightly worse than linear when the percentage of attacks increases. Generally,Scheme 1 is about 20% faster than Scheme 2.

Code or Data ?

When a service requesting message arrives at SigFree, SigFree first uses instruction sequence distiller called *ISD* to distill instruction sequences from the requests. however, instruction sequences may be distilled from any binary strings..

In phase 2, SigFree uses instruction sequence analyzer called *ISA* to analyze these instruction sequences to determine whether an instruction sequence is a program.

ISA – Scheme 1

Observation: a program has certain characteristics implying the operating system on which it is running e.g., calls to operating system or kernel library, whereas a random instruction sequence normally has no such characteristics.

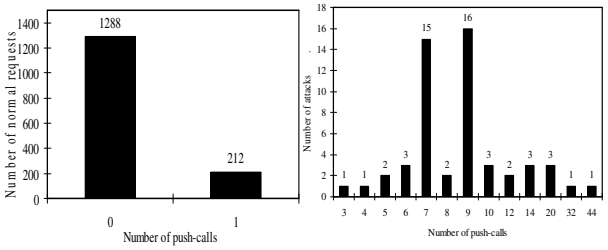
Criteria: if the number of push-call patterns in an instruction sequence exceeds a threshold, we consider it as an attack request.

Drawback: vulnerable to obfuscation.

Evaluation

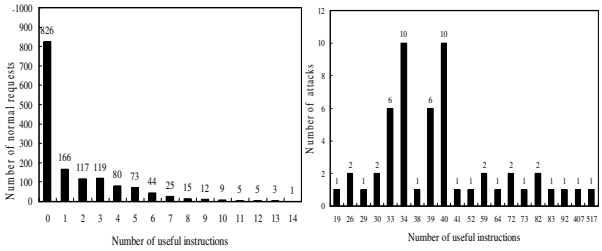
•50 attack requests generated by the Metasploit Toolkit, Worm Slammer, Code Red and one Code Red variation.
•1500 binary normal HTTP replies including encrypted data,audio,jpeg, png, gif and nd flash.
•200 attack requests generated by two famous polymorphic engine ADMmutate and CLET.

Scheme 1



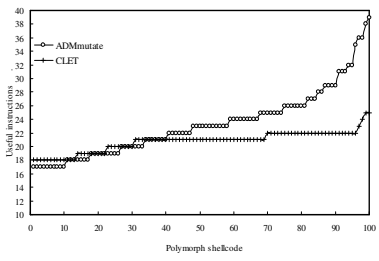
• normal requests contain less than 2 push-calls.
• attack requests contain more than 2 push-calls.
We can set the threshold as 2.

Scheme 2



• normal requests contain less than 15 useful instructions
• attack requests contain more than 18 useful instructions.
We can set the threshold a value between 15 and 18.

Detection of Polymorphic attacks



Observations:
. ADMmutate: #useful instructions: 17~39
. CLET: #useful instructions: 18~ 25

Research Supported by Army Research Office and NSF