Name: Vedant Borkute
SID: 862552981
NetID: vbork001
HW 1

Slammer Worm:

1. Figure 2 shows an interesting curve where the growth rate is very fast at the beginning then peaked out. Explain why the curves look like that.

Ans: The growth rate slows and peaks because random-scanning worms initially spread exponentially, but their rapid infection of new hosts slows as they keep trying to infect addresses that are already infected or immune.

2. Why does Slammer spreads much faster than Code Red?

Ans: Slammer spread nearly 100 times faster than Code Red. This is because Code Red was limited by latency, while Slammer was limited by bandwidth. Code Red spread using multiple threads, each calling connect() to open a TCP session. After sending a TCP SYN packet, each thread had to wait for a SYN/ACK response or time out, which prevented the thread from infecting other hosts. Its payload was also large (4 Kbytes), while Slammer's was much smaller. Slammer used a single UDP packet instead. This allowed it to broadcast scans without needing responses from potential victims. Its payload was very small compared to Code Red's.

3. Figure 3 shows the actual scanning rate of Slammer, why it didn't match the expectation?

Ans: After a while, Slammer fails to match the expected exponential growth. This difference happened because Slammer's aggressive scanning interfered with its own growth. In the later stages, it began to overload networks with its scans, leading to high bandwidth use and network outages, which limited its growth.

4. Why Slammer traffic caused problems for internet infrastructure devices like routers and switches?

Ans: Slammer traffic overloaded networks with a vast amount of traffic, many packets, and a large number of new targets (including multicast addresses). Many routers and switches crashed under this load. The disruptions mainly resulted from unintended internal Denial of Service (DoS) attacks caused by infected machines sending packets at their maximum rates.

………………………

Effective and Efficient Malware Detection at the End Host:

1. Why the authors propose modeling behavior using graph instead of sequence?

Ans: Malware authors can easily rearrange code, reorder independent system calls, or add irrelevant calls. This makes capturing the sequence ineffective. A behavior graph provides a more flexible way to show the true relationships (dependencies) between system calls while allowing independent calls to appear in any order.

2. What are nodes and edges in the behavior graph?

Ans:

a. The nodes represent system calls
b. An edge from node x to y indicates a data flow from system call x to y

3. Why system calls and why using a graph instead of sequences?

Ans:
a. System calls capture the program's interactions with its environment, making it likely that any relevant security violation is visible as one or more unintended interactions captured by system calls.

b. As mentioned earlier in Q1, malware detection models that rely on system call sequences are vulnerable to evasion techniques, such as reordering of independent calls. The graph structure solves this issue by capturing the essential data dependencies between system calls instead of depending on their strict temporal order.

4. The key to achieve "efficient detection" is to avoid using expensive dynamic data flow tracking (tainting) to determine the dependencies between system calls (i.e., match the edges during runtime), how does the proposed method work?

Ans: Their proposed method achieves efficient detection by monitoring only system calls and their arguments, without needing costly tainting during runtime.

a. Offline Analysis: During the initial analysis of the malware sample, the system uses the observed data flow to extract the relevant program slices (sequence of instructions) responsible for reading input and transforming it into the corresponding output.

b. Symbolic Expression Generation: Based on the program slice, a symbolic expression or algorithm is created that represents the semantics of that slice. This allows the system to pre-compute the expected output based on a given input.

c. Runtime Matching: When a monitored program executes a system call x, the scanner extracts the relevant output values. These outputs are then used as input to the symbolic functions associated with potential subsequent system calls (children of x in the behavior graph).

d. Dependency Verification: When a subsequent system call y is invoked, the scanner checks its arguments. If the value of system call y's argument matches the expected value calculated using the output of x and the function f, the data flow is considered detected, confirming the relationship between x and y.

……………………..

Click Trajectories:

1. What is the motivation/goal(s) of this work?

Ans: The goal is to develop a method for characterizing the resource dependencies (trajectories) behind individual spam campaigns and analyze the relationships among these dependencies.

2. What are the necessary infrastructure to host a spam website (i.e., the "click support")?

Ans:
a. Redirection sites: URLs that redirect to additional URLs.
b. Domains: Domain name resources registered through a domain registrar or reseller.
c. Name servers: Infrastructure provided by spammers or third parties (often "bulletproof" hosting services) to supply DNS records for the domain.
d. Web servers: Servers specified by the name servers that host or proxy the website content.
e. Stores and Affiliate Programs: These serve as storefronts and function as the advertising front end, providing store templates, shopping cart management, and contracting for payment and fulfillment services.

3. What are the three strategies to disrupt the spammer's business model?

Ans:
a. Registrar intervention: Disrupting domains by having individual registrars suspend domains used for advertising or hosting sites.
b. Hosting intervention: Interrupting the spam domain trajectories by contacting Autonomous Systems (ASes) that host the DNS and Web servers and pressuring them to shut down associated hosts.
c. Payment intervention (Banking): Interfering with the payment step by focusing on the role of the acquiring banks for each program.

4. What are the findings of this study (i.e., have they achieved their research goals)?

Ans: The study achieves its research goals and identifies concentration of resources in the payment tier:

a. Payment Bottleneck: The study provides strong evidence of payment bottlenecks in the spam value chain. For the pharmaceutical, replica, and software products analyzed, 95% of spam-advertised monetization is handled by just a few banks.

b. Infrastructure (Registrars/Hosting): Although some large providers (like NauNet, which registers nearly 40% of spam-advertised domains) exist, the supply of hosting resources is vast (thousands of providers, millions of compromised hosts), and the switching cost is low.

c. Intervention Efficacy: Because of the high concentration and high replacement costs for new banks, intervention at the payment tier is thought to be the most critical and potentially effective strategy for disruption. The study suggests that a “financial blacklist” enforced by Western issuing banks could significantly cut off funding for the operation.