**Spotless Sandboxes: Evading Malware Analysis Systems using Wear-and-Tear Artefacts** - Najmeh Miramirkhani, Mahathi Priya Appini, Nick Nikiforakis, Michalis Polychronakis - Stony Brook University

Malware sandboxes are one of the most popular methods used by anti-virus testing engineers to find out if the effects of various malware in order to develop and create sophisticated counter measures. These are virtual testing playgrounds that are built with weak protective measures in order to get affected by a virus.

However modern viruses have come up with a very new and innovative method against sandboxes in testing. The virus creators’ aim is to make sure the virus or malware is not caught at the testing phase and will go under the radar of the sandbox. Hence, without a countermeasure, it can release fatal payloads onto real systems for adverse effects. The proposed methodology to evade sandboxes are by ensuring differential behaviour in sandboxes. By detecting the environment, it releases payload conditionally.

This publication provides a method for sandboxes to catch such malwares with advanced evasion techniques by introducing wear and tear to the malware. Essentially, the sandbox is made as close to a real-life system as possible – specifically an old system. This is the wear and tear being introduced. It can be event logs, recycle bin size, cache entries, network entries, registry, cookie count and so on.

**Stuxnet -** Marie Baezner, Patrice Robin

Stuxnet is one of the most advanced and sophisticated worms ever written. It was designed by the Intelligence agency in the United States and targeted at nuclear refinement plans in Iran. They released a worm through USB to the general public which latches on to any system and gathers administrative privileges for the system using zero-day exploits. This virus uses a total of 4 zero-day exploits. Then it searches for specific Siemens Centrifuge controllers connected to the system. If it does not exist, the worm destroys itself.

Once it has control of the controller, it slowly varies the centrifuge speeds over a period of time so that they are all destroyed within a year. The also increase the gas pressure to form rocks within the centrifuges so that they degrade in quality – slowly but steadily.

The worm is hailed as one of the most complicated codes ever written in the history of mankind. It floated on the internet for over a year without being detected by any machine or anti-virus.

**Catch Me If You Can: Evaluating Android Anti-Malware Against Transformation Attacks -** Vaibhav Rastogi, Yan Chen, and Xuxian Jiang

An analysis on malware detection and removal techniques designed for android devices shows worrisome results. Mobile Industry has grown exponentially at a very large rate in the last few years. This period of time is also marked with an increase in malware count designed for android. However, the detection rates stay the same for the most part.

The analysis classifies various malware evasion techniques into three categories – Trivial, DSA or static analysis, and Undetectable. The malware is taken through one of these transformations and passed into a system with an anti-malware setup on it.

The study came up with many findings. The major takeaway is that all anti-malware systems on android are vulnerable to many transformations and are not up to standard.

The second finding is that android anti-malware works primarily based on code level artefacts – package names, asset names etc. However, at least 43% of malware don’t use these techniques at all and hence have an easy path into the system.

90% of the malware designed did not need protection against static analysis of bytecode. The android security system does not bother implementing this vital layer on their systems and hence it is a waste of resource to enable evasion techniques for this method.

**Malware Detection and Evasion with Machine Learning Techniques: A Survey** - Jhonattan J Barriga, Sang Guun Yoo - Escuela Politécnica Nacional

With the advent of cheap computing power available freely for the general public, malware detection has become near impossible. Malware uses various techniques such as Encryption, Oligomorphic, Polymorphism, Metamorphism, Obfuscation, Fragmentation, Session splicing, Protocol Violations, Code reuse Attacks and more.

The most common malware detection techniques used are Signature method, Behaviour method, and Heuristic method. However, these are simply not enough to catch the complex multi layered evasion techniques that most malwares use nowadays.

The method being used here is a neural network to catch trojans. The suspected payloads are injected as weights into the neural network. The network catches the trojan when the network receives a very specific type of data.

**Malware Obfuscation Techniques: A Brief Survey -** Ilsun You, Kangbin Yim

Obfuscation is by far the most dominant technique along with Signature methods used by malware for evasion and staying out of reach of anti-malware systems. There are various Obfuscation techniques and this paper analyses some of those methods.

The simplest obfuscation technique is the dead code technique. It inserts harmless pieces of code at random to various parts of the payload but does not alter the behaviour in any way. The code is placed at clever locations that skip anti-malware scans. Another complimentary method is Register Reassignment. Registers support legacy code. By switching from current standard to older generation of registers often, the Malware becomes very hard to find.

Subroutine reordering is a very simple but effective Obfuscation technique. It generates n! variants where n is the number of subroutines. It randomly rearranges the subroutine while having minimal changes on the payload or the behaviour of the malware.

Instruction Substitution and Code Transposition are two other methods of obfuscation. Instruction Substitution splits a single instruction into multiple instructions while having the same effect. It effectively changes code with equivalent instructions. Code transposition reorders entire sequences of codes rather than just subroutines.