Security vs. Reliability
Is one more difficult to achieve than the other?

Engin Kirda
ek@ccs.neu.edu

Northeastern University

Great to be back at TU Wien!

- … the institution where I studied, did my Ph.D., and worked as faculty until 2007 ;)

Overview of this talk

- I will talk a bit about my research background
- I will give a brief overview on some past academic research
  - This research led to the founding of Lastline, Inc. (LL)
- I will discuss a number of observations I have made based on our (my) experiences
  - This talk was a great opportunity to reflect on the security versus reliability debate

My Background

- I moved to NEU in January 2011
- I was faculty in Europe before
  - Technical University of Vienna
  - Institute Eurecom
- I am active in these areas:
  - Malware analysis and detection (since 2004)
  - Web security (since 2004)
  - Securing systems of all sorts

- Interested in all practical security problems

My Background in Reliability

- I have had some involvement in the reliability community over the years
  - Mainly DSN where I've served on the PC and have published papers
  - … and SRDS – back in 2008!
- However, I am certainly no authority in reliability – my sole focus has been security

Lastline: How it all began – 2004 – malicious code

- There is (was) a wide variety of malicious code
  - viruses, worms, spyware, rootkits, Trojan horses, …
- Common characteristic
  - perform some unwanted activity on your system
- No doubt, everybody had heard of viruses, worm epidemics, or spyware (more commonly called malware today)
  - reports in mainstream media
  - personal experience (at least, with virus scanners)

Malicious Code Analysis

- Understanding functionality of malware programs
  - modifications to compromised system
  - understand questions such as:

    how is program launched, what malicious actions are performed, *hidden functionality (with trigger)*, disabling of defense mechanisms, interaction with other processes …

- Necessary both for *detection* and removal

- Must keep up with increasing numbers of samples
  - fast
  - automated (at least, provide as much support as possible)
  - precise

- Interesting with regards to automated malware collection (honeypots)

Anubis

- Analyzing Unknown Binaries (Anubis)
  - http://anubis.iseclab.org (now obsolete)
  - Online service where Internet users could submit binaries
  - Reports were generated that described the actions of the binary
- Some of our users were…
  - Shadow Server, Team Cymru, CERT Australia, law-enforcement agencies, many anti-malware companies…

Analysis Information

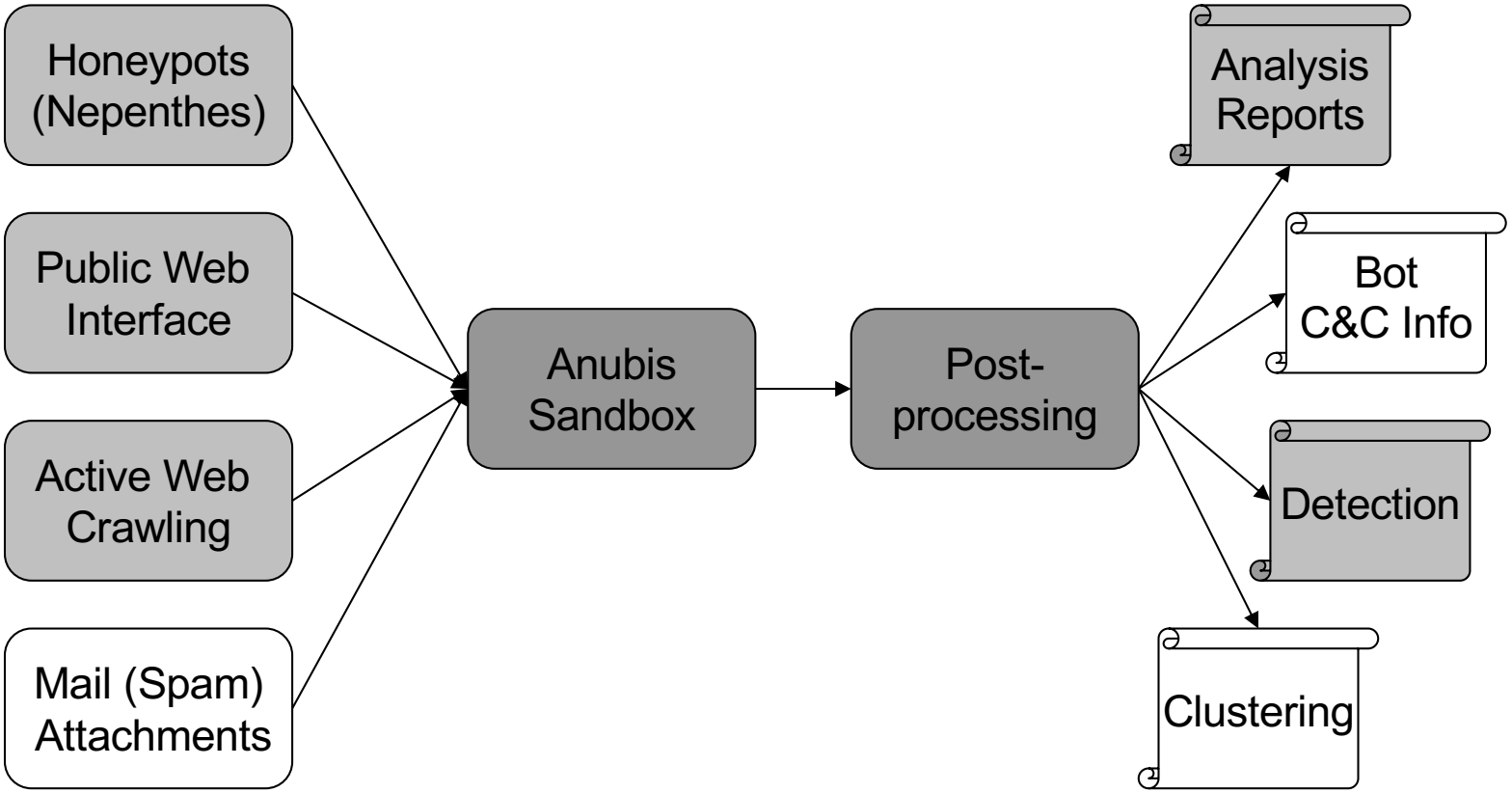
- Process interacts with operating system via system calls
  - needs OS for every interaction with environment
    - file system, network, registry, …
  - monitor system calls
  - unfortunately, on Windows, system calls were largely undocumented and could change without notice
  - developers were supposed to use Windows API, which denotes a collection of stable, user-mode, shared libraries
  - of course, Windows API could be bypassed

→ we monitor Windows API calls and NT kernel calls

Analysis Report

- File activity
  - read, write, create, open, …
- Registry activity
- Service activity
  - start or stop of Windows services (via Service Manager)
- Process activity
  - start, terminate process, inter-process communication
- Network activity
  - API calls and packet logs

Initial Anubis Architecture



Malware Detection

- Run simple rules on output
  - can flag scanners (number of contacted IP addresses)
  - keyboard loggers (installed keyboard hooks)
  - mass mailers (spam mails sent)
  - bots (suspicious IRC traffic)
  - copy to system directory

- We could do a more powerful analysis
  - after all, we had a system emulator and complete control
  - detect unusual information access and processing patterns
  - capture information flows (tainting)

Anubis Became Very Popular

- ANUBIS started attracting thousands of users and fans
- We also worked on other technologies besides ANUBIS that was the main workhorse
  - WEPAWET (for Javascript analysis)
  - EXPOSURE (developed in France, for detecting malicious domain names)
- Around 2008, we started receiving many licensing requests from users
  - And some companies wanted to give us money to help them (i.e., consulting) to build similar systems

October 2009

- We decided to pull the trigger and create Lastline
  - Founders were Giovanni Vigna (UCSB), Christopher Kruegel (UCSB), and myself
- Problem: There was no money, and no product
  - Everything had to be created from scratch. You can't just take existing code and use it
- Solution: We licensed Anubis and Wepawet from UCSB (for a small fee)
  - We could use the malware analysis capabilities and infrastructure

Until 2020…

- Lastline raised $52 million VC investment (through to Series C)
- Grew to about 140+ employees
- Had offices in Europe, Asia, and the US
- Was headquartered in the Bay Area
- Made OEM deals with many companies
  - was providing threats intelligence and analysis services to them
- Had hundreds of customers and protected millions of end-users
- Was acquired by VMWare in 2020

Products Components at Lastline

- Sandboxing: Expert systems produce reliable metadata
- Malware traffic analysis: Machine learning produces intelligence
- Malware program analysis: Machine learning creates code clusters (JavaScript, binary) to classify behavior
- Email content analysis: Machine learning detects phishing attempts and Business Email Compromise (BEC) attacks
- Network traffic analytics: Machine learning establishes baselines for analyzed networks
- Anomaly detection: Machine learning identifies suspicious actions

So, which is easier to achieve?

- Reliability
  - Building a software product that is stable, efficient, of great performance, and free of bugs
  - Failure to build a reliable product means your customers will be upset, less protected, and it'll cost them money
- Security
  - Your security product needs to identify all possible threats, deal with active evasions, be vulnerability-free, and also be easy to use
  - Failure to build such a product means your customers will be upset, less protected, and it'll cost them money

Observation 1

- Argument: QA / reliability people have the advantage that they can use specs to determine ”what is a bug and what is a feature”
  - Security teams do not have this advantage of course and threat models are often incomplete
  - Security teams are often reactive because they need to first see what bad guys are up to (and then react to it)
- The problem here is that you do not often have specs
  - Or the specs you have might be ambiguous and incomplete!
  - Or you depend on third-party code
  - For us at LL, this was the case for most of our networking code

Capturing Network Data

- We relied on Suricata to capture network traffic from the wire
  - It's open source, well-known, and should be reliable
- The reality was that reliability of the network capture became a huge issue for us
  - Missing packets
  - Intermittent failures
  - Crashes

- At times, getting the network capturing reliably at high speeds became more challenging than the security issues at hand

Observation 2

- Argument: Security people have the advantage that they only have concrete threats they need to deal with, not the entire bug-space
  - The reality, though, is that some of the threats are insanely complicated
  - The adversary is very sly and cunning
  - And technically, there is no easy and complete solution to address the issue
- For us at LL, sandbox evasion was a *constant* issue

Evading Dynamic Analysis

- Malware can detect runtime or analysis environment
  - differences between virtualized and bare metal environment
  - checks based on system (CPU) features
  - checks based on operating system artifacts (files, …)
- Malware can exploit limited context

Environmental Awareness

- Malware can avoid being analyzed
  - tricks in making code run that analysis system does not see
  - wait until someone does something
  - time out analysis before any interesting behaviors are revealed
  - simple sleeps, but more sophisticated implementations possible
  - move code into kernel space (rootkits)

Timing-based Evasion

Detect Analysis Environment

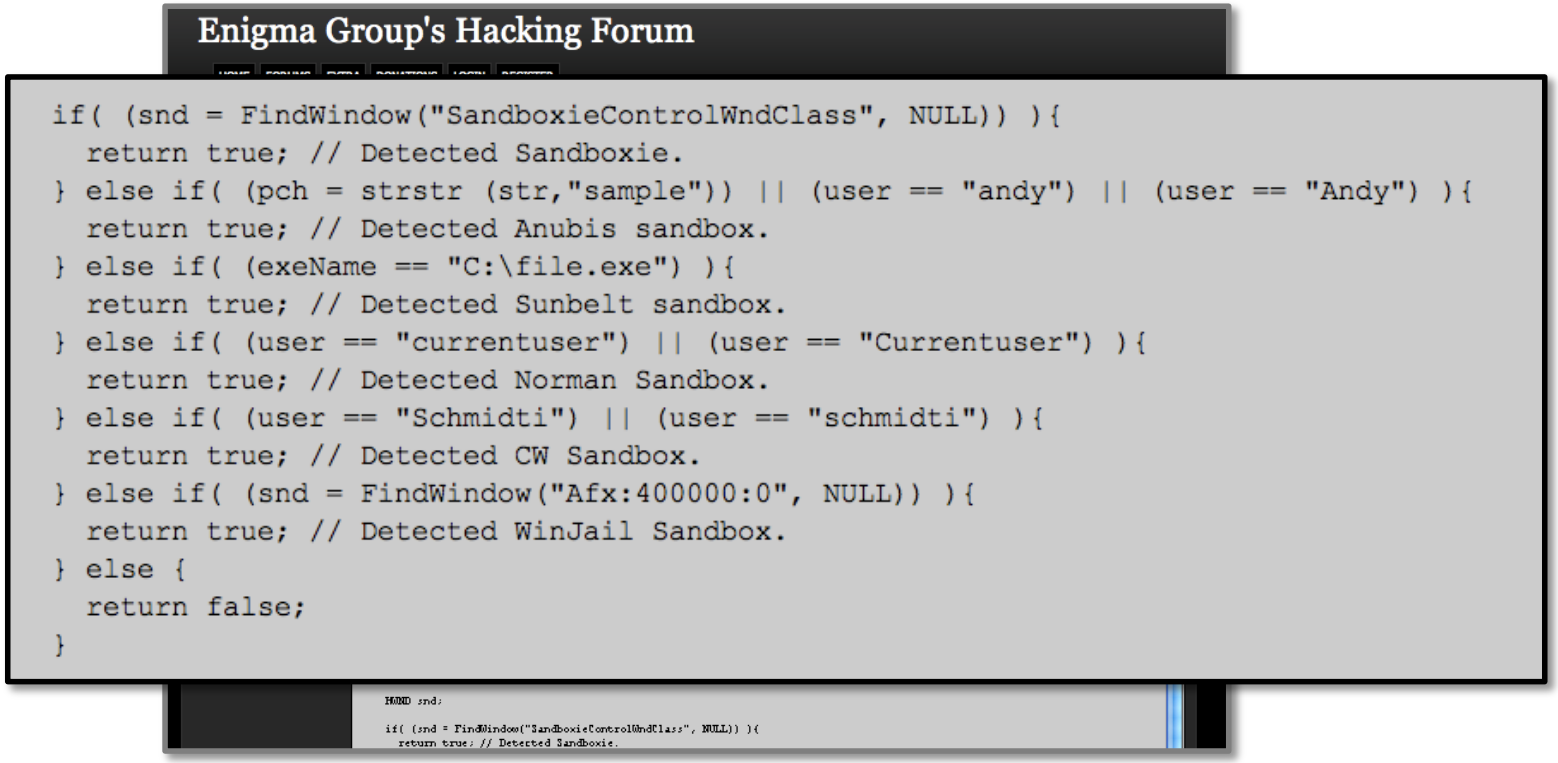
- Check Windows Product ID

  `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProductID`
- Check for specific user name, process names, hard disk names

  `HKLM\SYSTEM\CURRENTCONTROLSET\SERVICES\DISK\ENUM`
- Check for unexpected loaded DLLs or Mutex names
- Check for color of background pixel
- Check of presence of 3-button mouse, keyboard layout, …

Detect Analysis Environment

Detect Analysis Environment



Observation 3

- Argument: Reliability people have the advantage that they can use metrics
  - Metrics: Bayesian statistics, reliability modeling, Mean Time Between Failure, etc.
  - This is true and a major improvement over us security people!
- In security, the community has made attempts, but nothing has really stuck
  - We count vulnerabilities to try to predict, but prediction rarely works
- For us at LL, we really did not have a way to measure success (how much are we better?)

“Why are you better?”

- A common question at customer meetings when you are selling a security product
  - How do you show that your product is *better* and provides more security than another product?
  - What metrics do you use?
  - Why are these metrics the right metrics?
- The most common customer methodology
  - A product “bake off” where products are pitched against each other
  - Number of alerts are compared
  - Problem: Not all alerts are created equal

Third-Party Evaluations

- Of course, there are third-party evaluations too…
  - The Gartner Magic Quadrant
  - An analyst evaluates you, and places you somewhere
  - You do need a good connection to Gartner…

- NSS Labs
  - A company that made a good attempt to evaluate different products and rate them
  - The problem: First, you need to pay to play, Second, how realistic are the tests? Third, there is time for optimizations…

Observation 4

- Argument: Yes, security people are bad at metrics, but they are good at reacting to and mitigating threats
  - Entire classes of vulnerabilities have been removed (e.g., stack overflows)
  - If a new trick emerges, or a new attack, security people can often quickly identify and analyze it
  - The reason why there is an arms race is because security people catch up quickly with the bad guys
- For us at LL, we had “threats intelligence” teams constantly looking for new threats, and informing product development

Threats Intelligence

- Teams constantly look through your detections
  - Try to identify novel threats
  - Analyze detections, and write stories
- A good threats intelligence teams can create great publicity and awareness
  - Engineering teams can quickly try to catch up and mitigate the new threat

So, which is easier to achieve then?

- Reliability and security, obviously, are both critical for customer protection and satisfaction
  - Although both communities are very lively, there is less communication between them than should be
  - Both communities can learn from each other
  - Security people often are not aware of the decades worth of reliability research
  - Reliability people are often not aware of the existing security research, and sometimes ”reinvent” the wheel

- The answer is: It depends…

Conclusions

- I gave a brief overview on the company I co-founded, and my research background in security
- I talked about reliability versus security, and elaborated on if one is easier to achieve than the other
- Sure, my views are biased and are based on my background and experiences
- In any case, I hope there is more integration of the reliability and security research areas in the future

Questions?