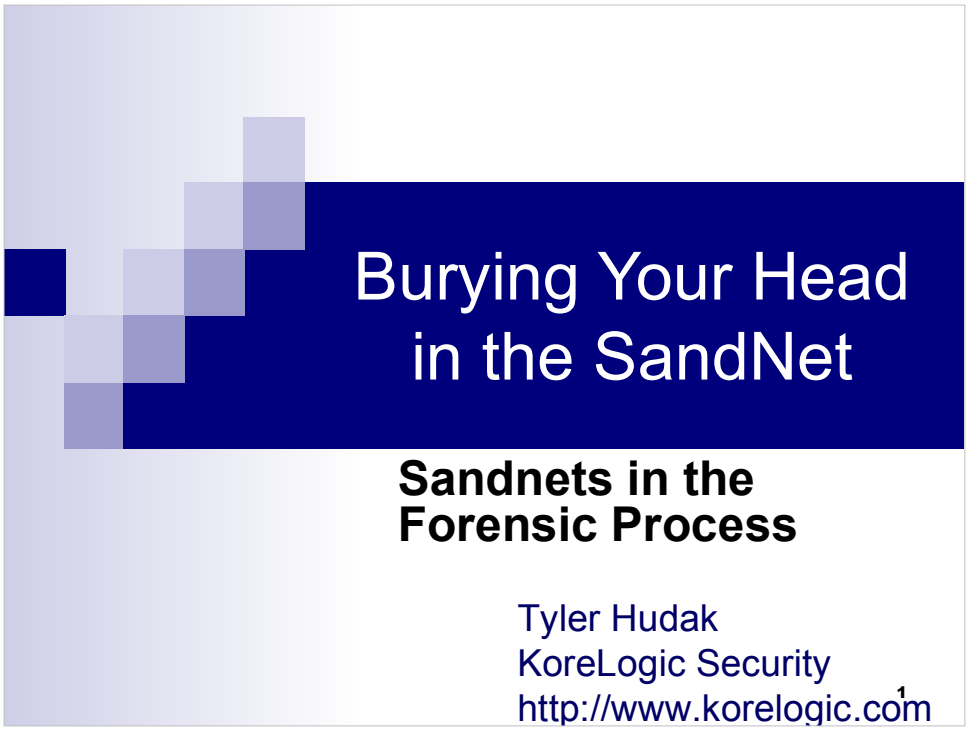
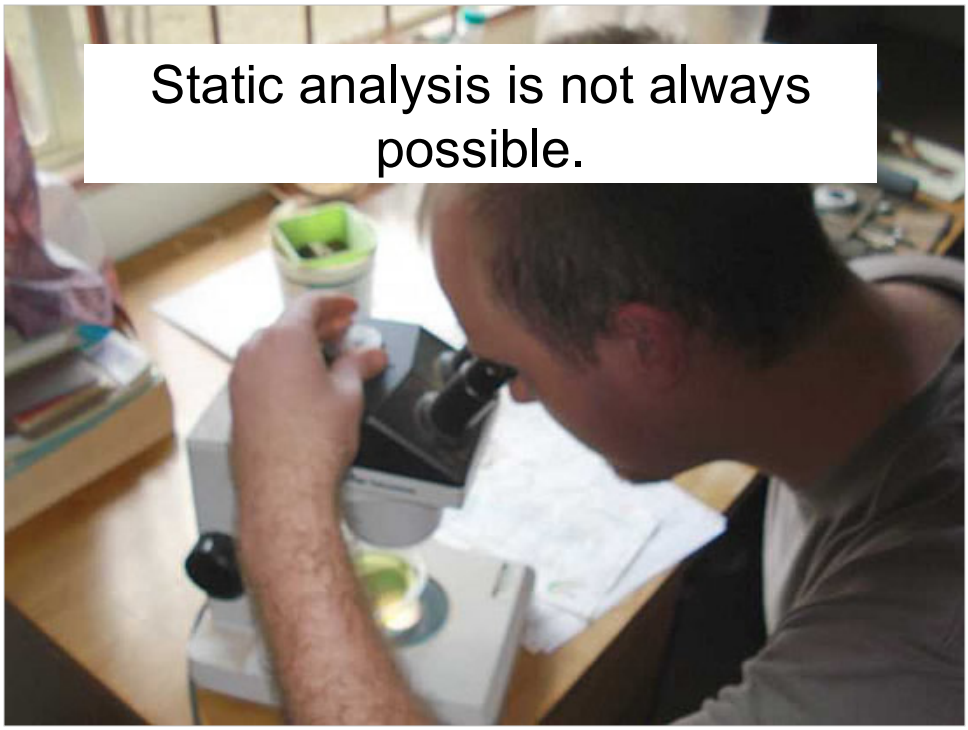


Tyler Hudak
Senior Security Consultant
KoreLogic Security
thudak@korelogic.com

Often, rapid turn-around is needed in an investigation, especially when you are working on time-sensitive cases.

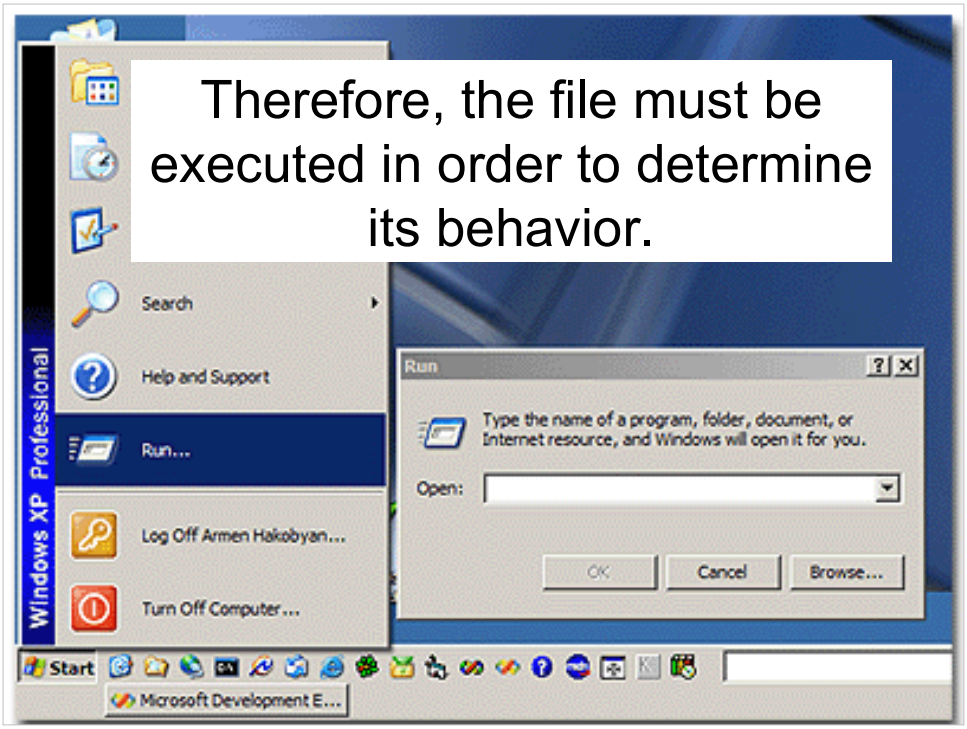
Even if you don't work on these types of cases, someone wants the results from your investigation yesterday. It is very rare someone tells you to take your time on a case.

Inevitably, during a forensic investigation or in incident response, you are bound to come across unknown executables which you will need to determine what they do. Typically this is done in 2 ways: static analysis and dynamic analysis



Static analysis involves looking at the file without executing it. This is accomplished by examining internal strings, the file header, putting it into a decompiler, etc. However, more and more malicious files are using packers to encode their internal data and make it extremely difficult to determine what they do.

Even if the file is not packed, the internal strings may not reveal its behavior and the assembly language of the file will have to be examined in a decompiler. However, very few people have the knowledge and the time to do this.



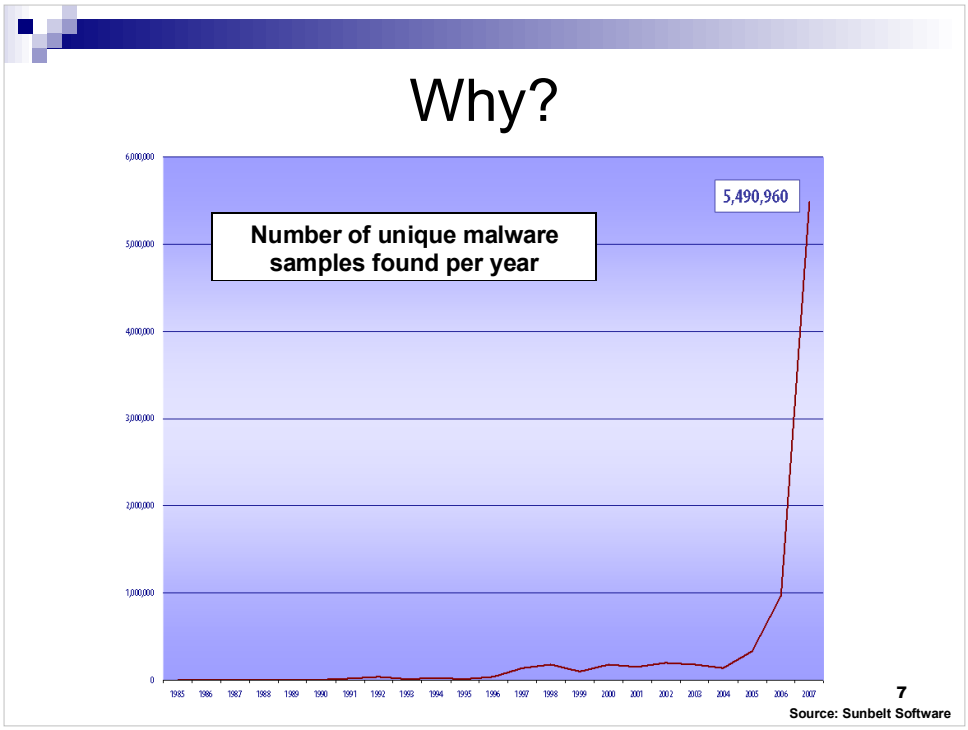
Therefore, the best way to determine what an unknown file does is by dynamic analysis, or executing it to determine a file's behavior.

Executing an unknown executable can be a tricky thing since it could be malicious. You don't want to execute it on your examination workstation, a production machine or anything connected to the corporate network. If the machine was infected and the infection spread internally, you'd be in pretty bad shape.

Therefore, a solution is needed where:

7. You control the environment of the system so you know exactly what is normal on the system.
8. You can monitor the executable's behavior so you know exactly what it does.
9. The process is automated to allow analysis to be done quickly.

The solution: sandnets. By utilizing a sandnet, you will have a safe way to analyze potentially malicious software quickly.



So, how do sandnets fit into the forensic process? The picture above explains it all. This graph, courtesy of Sunbelt Software (http://sunbeltblog.blogspot.com/2008/01/growth-of-malware.html), shows the number of unique malware samples by year up to 2007. It should be obvious that with over 5 million unique samples in 2007 malicious programs are not going away any time in the near future. Additionally, not only is there more malware, but malware is getting smarter, more devious and targeted towards individuals and organizations.

With these trends, anti-virus companies will not be able to be relied upon as heavily as they have been – and incident response and forensic analysts will need to be prepared to quickly analyze an unknown binaries. Analyzing the static qualities of the file will yield little to no results and your only solution will be to execute and observe the program.

Having a sandnet in place, within your forensic process and lab, will allow you to determine the behavior of an unknown executable quickly and in a way which does not endanger your systems or networks.

Sandnet

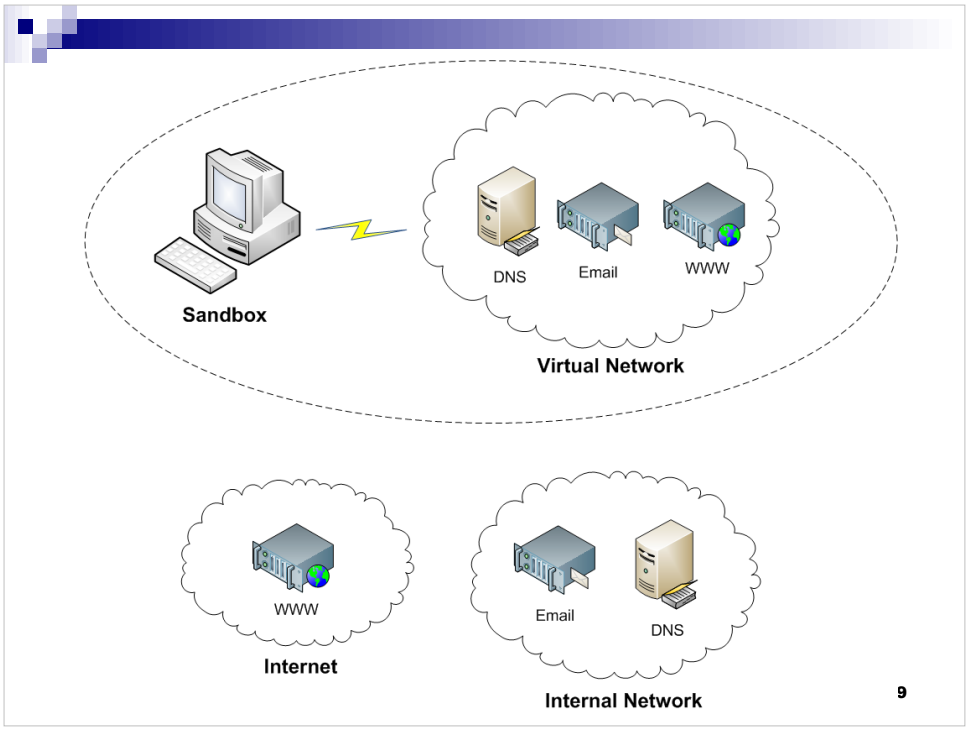
- A virtual network for safely running programs
- Two components
  - Sandbox
  - Network simulator
- No contact with "outside" network

8

Sandnets are a virtual network for safely running programs and are composed of two components – a sandbox and a network simulator.

The main idea behind the sandnet is that the sandbox is on a closed network where no contact, at all, is made with any outside network. Any network connection is to a simulated network where the results are "spoofed" back to the sandbox.

This allows us to execute a program on a system totally segmented from any other network, including the Internet. With the system being segmented, there is no worry about a malicious executable infecting production systems. Also, because we control the simulated resources, we can determine what the responses are and can control the information the program receives. This becomes useful during analysis.



This picture describes how a sandnet works. When the sandbox attempts to contact any network resources, instead of going to the actual resource they contact another program or system which simulates the results back to the program. At no time does the sandbox contact actual resources on the Internet or on an internal network.

Sandbox

- A virtual container for running programs safely
- Network connections are with actual services and machines
- One program or a whole machine
- Two types
  - Virtual
  - Physical

The first component of the sandnet is the sandbox. A sandbox is a “virtual container for running programs safely”. In other words, it is a segmented area where a program can be run without fear of it corrupting the system it is on.

Sandboxes can be a single program or a whole machine. The sandboxes we talk about in this presentation will be the whole machine variety. Some examples of sandboxes include:

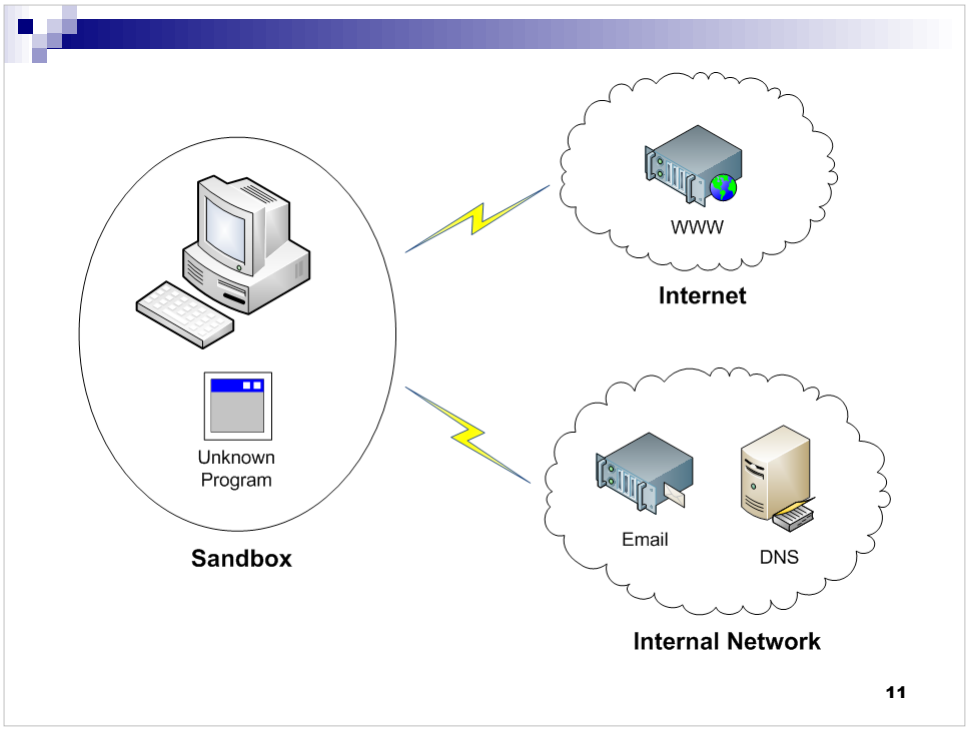
• Java virtual machine, which sandboxes all Java code into an environment where it (supposedly) cannot access the underlying system.

• sandboxie (www.sandboxie.com) which puts programs executed within it to into a virtual area where all file and registry accesses are temporary, and are removed once sandboxie is closed. This is perfect for safe surfing with IE.

• VMWare, or other virtualization programs, can even be thought of as a type of sandbox

The key is that for the sandbox, if network connections are allowed, they are made to the **actual resource**. When the sandbox wants to look up an address via DNS, it contacts an actual DNS server.

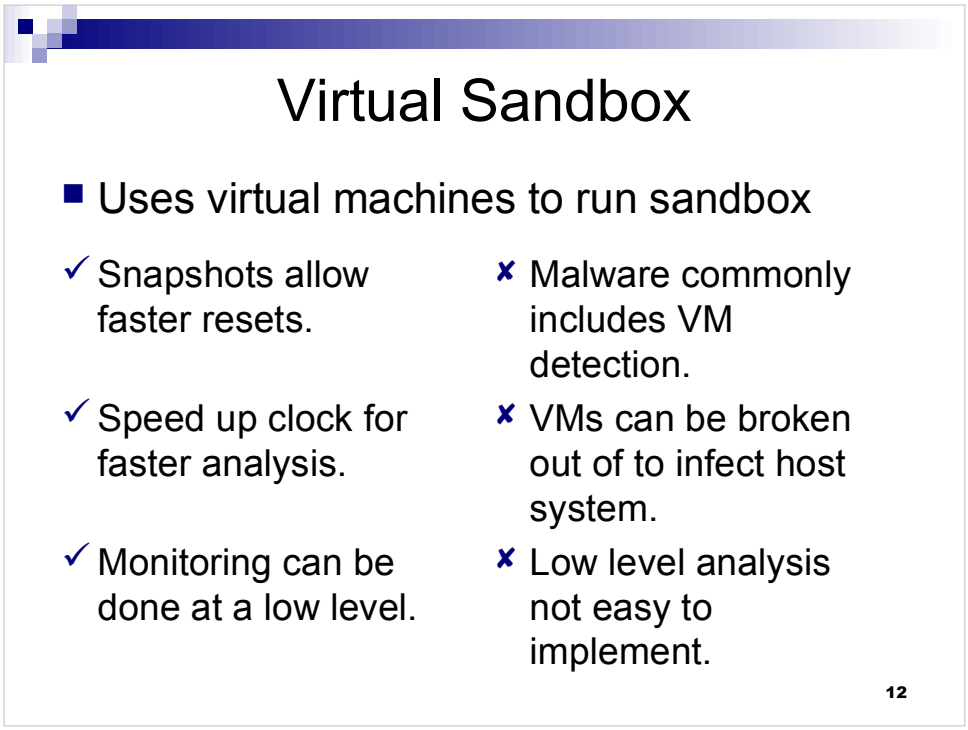
There are two types of sandboxes: virtual and physical.



This picture shows an example of how a sandbox works. The sandbox system contains an unknown program whose behavior we wish to determine. If, and when, the programs makes any network connections it does so to those actual resources.

If it wants to resolve an address, it connects to the local DNS server. If it wants to send an email, it connects to the local SMTP relay. If it wants to go to www.myspace.com, it actually connects to www.myspace.com.

Compare this to the previous picture of the sandnet. As you can see, a sandnet is a logical extension of the sandbox. Instead of contacting the actual network resources, we spoof the network to the sandbox allowing it to execute in a safe environment.



Virtual sandboxes are ones which use virtual machines, such as VMWare or QEMU, to run the sandbox. This has both advantages and disadvantages to it.

First, virtual sandboxes are easy to reset. Most VM programs offer a "snapshot" functionality which allows the user to save the system at a point in time. This could be used to snapshot the point just before an unknown program is run. Whenever a new program is ready to be run, the baseline snapshot can be returned to quickly.

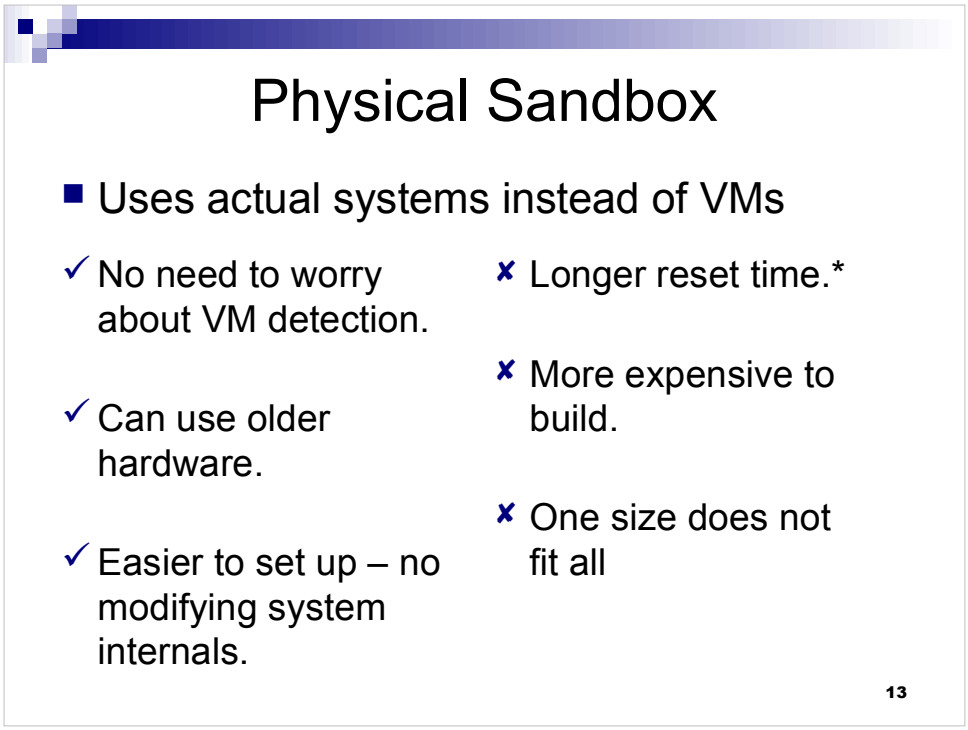
Since the sandbox is done through a virtual machine, the virtual system clock can be sped up to allow execution to be done much faster than normal. This allows faster analysis to occur. Note that this can usually only be done on VMs which have the source code available.

Finally, on VMs with available source code, the low level CPU, kernel and system functions can be "hooked" in order to log what is occurring at a certain point it time. This is useful for defeating things like rootkits which attempt to hide this information from monitoring tools.

There are a few problems with virtual sandboxes. First, malware commonly includes code to detect if it is in a virtual machine. If it is, the malware will act differently than normal – something which is opposite to our goals and increases the time it takes us to analyze the malware.

Virtual machines run where the virtual system sits on top of a host system. Because of this, the possibility exists that a malicious program could break out of the virtual machine and infect the host machine. This is not just theoretical – in 2007 Tom Liston and Ed Skoudis gave a presentation where they demonstrated a number of ways a VM could be broken out of.

Finally, modifying system and kernel internals for monitoring is not for the faint of heart. This is not an easy task and could easily result in an unstable system. Granted it is not necessary to do this as normal monitoring tools can still be used.



Physical sandboxes use an actual system instead of one within a virtual machine. The main advantage is that there is no need to worry about VM detection within the malware. This provides a more "truthful" environment for the malware, making it more likely it will act like it normally would.

Since physical sandboxes do not need to worry about sharing system resources with a host OS (like VMs do), high-end requirements for the box are not needed. This allows the reuse of older hardware which may be laying around not being used (and saves on the budget for your system).

To monitor a program in a physical sandbox, monitoring software is used. Using monitoring software makes systems easier to set up – there is no need to modify system internals. Additionally, desktop images can be installed and used on the systems to get an accurate view of what a program would do in the actual, corporate environment.

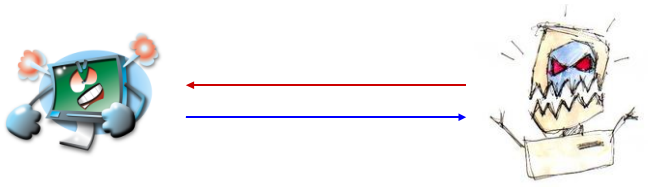
Of course, there are disadvantages to this approach as well. Primarily, it can take longer to reset the sandbox to a known good state as there are no a reliable "snapshot" features by default and re-installing the OS from scratch each time is too time consuming. Fortunately, there are hardware and software solutions available. CoreProtect has a hardware solution, named CoreRestore, which allows a hard drive to be placed back to its original state after a reboot. Software solutions using imaging programs (both corporate and open source) can be used to speed up the process as well. - http://www.coreprotect.com/core_restore.html

If you don't have hardware laying around you can use to build the sandbox you'll need to purchase some. This tends to be more expensive that just buying an OS license for a virtual machine. Fortunately, as previously discussed, you do not need bleeding edge hardware in order to build a physical sandbox – refurbished or used hardware tends to work fine.

Finally, unlike virtual sandboxes, one sandbox may not be able to be used for every configuration you wish to test. For example, if you have a sandbox which meets the requirements for Windows XP, you may not be able to test Windows Vista with it because it lacks certain higher-end hardware.

Simulating Network Services

- Emulates the Internet
- Hack the kernel/APIs
  - Send responses “internally” to the program
- Use a second machine which responds to the sandbox
  - Scripts/programs act as network services



14

Often, malicious binaries will communicate with services on the Internet to download additional malware, communicate to their “masters” or to attack other computers or spam. Watching what theses binaries do can provide us a lot of clues into what they do and how they work. Network simulation is usually done in one of two ways.

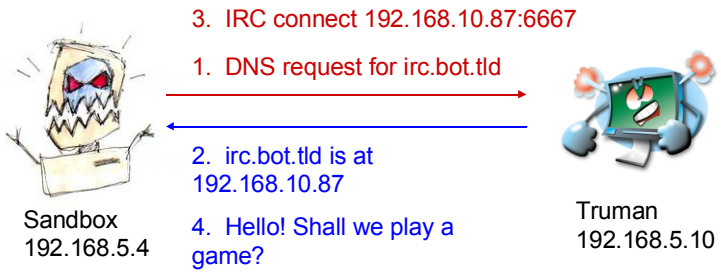
The first way is to hack the kernel or networking APIs such that when they see specific requests for network services (ie. DNS, HTTP, IRC, etc) they respond internally and network traffic never actually leaves the system. By doing so, this allows you to create a one-system sandnet. However, most people do not have the skill necessary to take this approach.

The second, and more common, way to do this is to set up a second machine connected to the sandbox via a hub or cross-over cable. This second machine runs programs and/or scripts which respond back to the sandbox. This approach is much easier to do, especially since there are numerous freely-available or open-source programs for just about every protocol imaginable.

The good news is if you don’t have the time or knowledge to configure or code your own services, it has already been done for you!

Truman

- Written by Joe Stewart of SecureWorks
- Simulates DNS, FTP, IRC, SMTP, SMB, MySQL
- Contains collection scripts and setup instructions

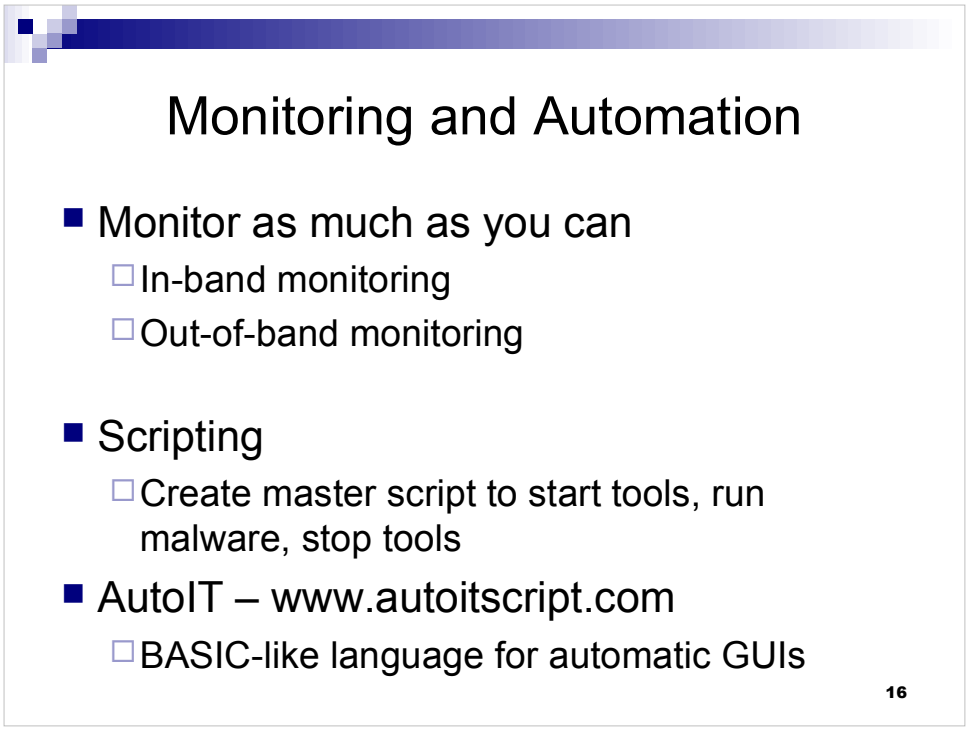


15

Joe Stewart of SecureWorks (formerly LURHQ), wrote a package named Truman. Truman contains instructions on how to set up a sandnet, complete with scripts to simulate DNS, FTP, IRC, SMTP, SMB and MySQL servers.

Truman works by pointing all requests for services back to itself. If the sandbox tries to resolve the DNS for an IRC server, Truman responds with a a random IP address in a pre-configured range. When the sandbox connects to the fake IRC server on the IRC port, Truman will intercept the traffic and redirect it to an IRC server it has running where it will record the data the program sends, allowing an analyst to look over it further to determine what the program was trying to do on the server.

One thing to remember is that Truman is no longer maintained so don't plan on any enhancements to be released. Of course, there is nothing stopping you from using Truman as a base and creating your own simulation system.



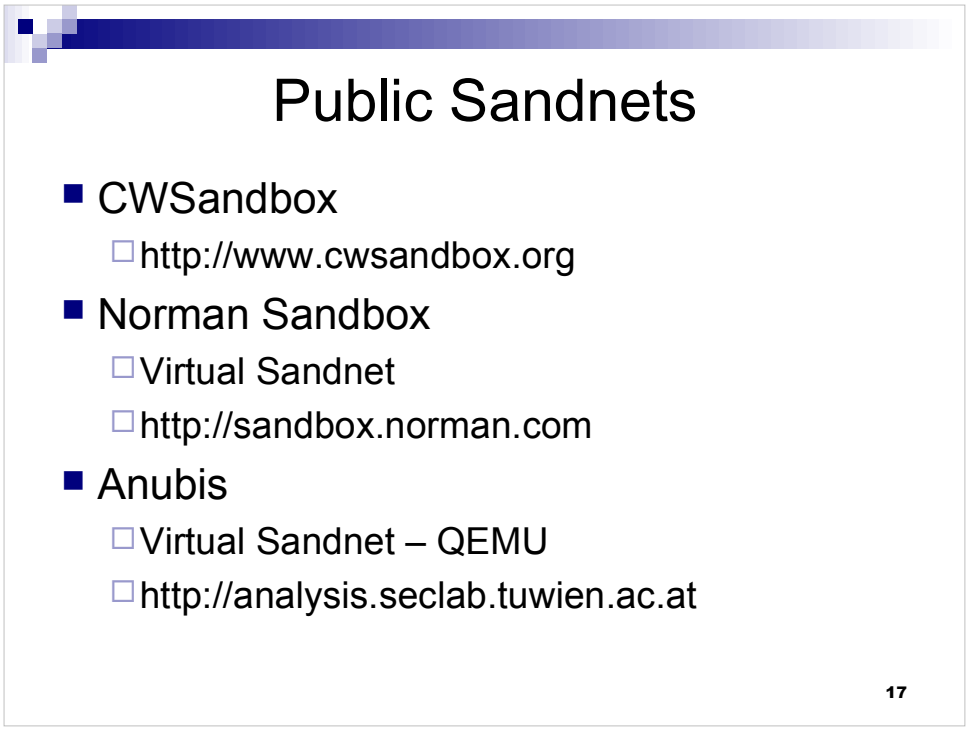
Two of the goals for the sandnet are to monitor what the program does and automate it to allow analysis to be done faster.

Monitoring is easy. There are lots of great tools out there – use them! Which tools to use is beyond the scope of this presentation, but searching the Internet for malware analysis tools will yield a treasure trove of tools. To get started, check out Process Monitor from sysinternals.com and InCntrl5, a tool used to show you changes that occurred on a system between two points in time.

Automating the tools and analysis of the system should be a major goal of any sandnet. This will allow you to have your tools run the same way every time you analyze a program, giving you consistent feedback. In the end, the analyst should be able to just submit the executable to the sandnet and wait for the results.

This can be accomplished easiest by using scripting. There are many scripting languages available for OS'. Take advantage of these to create a master script which starts tools, runs the malware, stops the tools and analyzes the results. While it may take some time to create, its worth the effort.

For Windows GUI-based tools which cannot be scripted, look at AutoIT. AutoIT is a basic-like language which allows the scripting of actions within a GUI – clicking on a button, selecting a menu, etc. http://www.autoitscript.com



Fortunately for those who don't have the time, knowledge or resources to create their own sandnet, there are ones available on the Internet for you to use. Public sandnets offer an interface for you to submit your file as well as a way to send you the results – usually through email. Keep in mind there is no guarantee on the time it will take to analyze though.

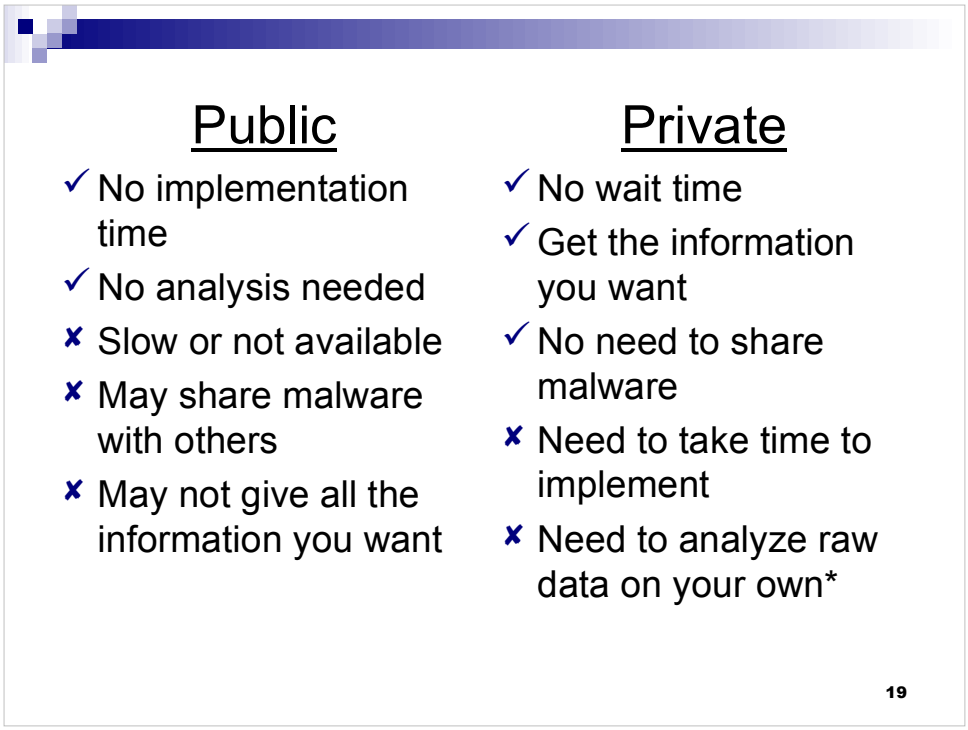
Note that these sandnets are not available for download, although they may be able to be purchased. They are privately held and you can only use them to get results – not to install within your own network. Other than Truman, I do not know of any publicly available sandnet (or sandnet code) which does this.

This slide and the next list the publicly available sandnets known to me – if you know of any others please let me know and I will add them to the list.

Public Sandnets

- Joebox
  - Actual sandnet running MS Vista
  - http://www.joebox.org
- Threat Expert
  - http://www.threatexpert.com/
- Bleeding Snort Sandnet
  - Registration required
  - http://sandnet.bleedingthreats.net/

18

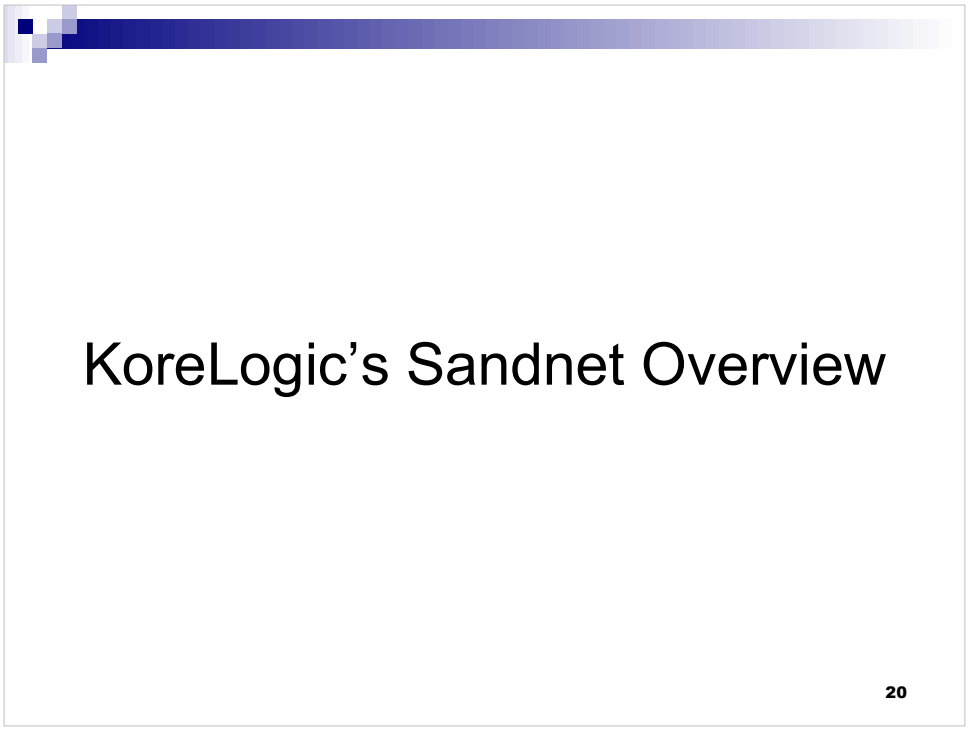


The question which inevitable comes up is do you use public sandnets or do you build your own. It will always boil down to this: do you have the time required to put your own together?

If you do, it is worth the effort. You will not have to worry about waiting for results to get back or if the sandnet you use is even available. Additionally, there could be times you do not want to share the program you are running through the sandnet out – there is no guarantee of this when you use a public sandnet which may send them to AV companies or other researchers.

Of course, using public sandnets saves you the development and implementation time. You also don't need to worry about analyzing the raw data to develop your own reports, which may be difficult if you are not familiar with the system internals of the system you are using.

Remember, there is always the possibility you can contract out to a third party who runs a private sandnet or can build you one. This gives you the advantage of having a private sandnet with higher availability without the need to maintain it on your own.

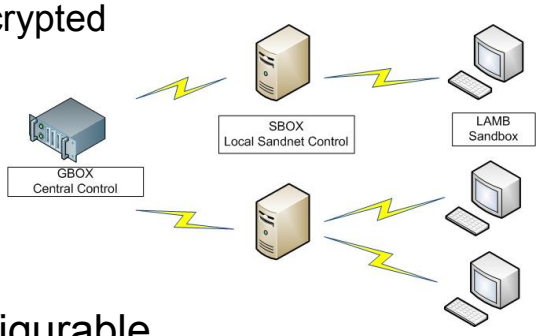


The KoreLogic Sandnet was originally created to allow KoreLogic analysts to test the effectiveness of various AV suites in the removal of malware. However, we soon found that it was extremely useful for a large number of duties and would help our clients a number of times. The following is an overview of our sandnet and how it functions.

This is only one way a sandnet can be put together and is not meant to be a definitive guide or methodology (although we think its pretty good ☺).

KoreLogic Sandnet

- Globally Distributed / Centrally Controlled
  - All communications authenticated and encrypted



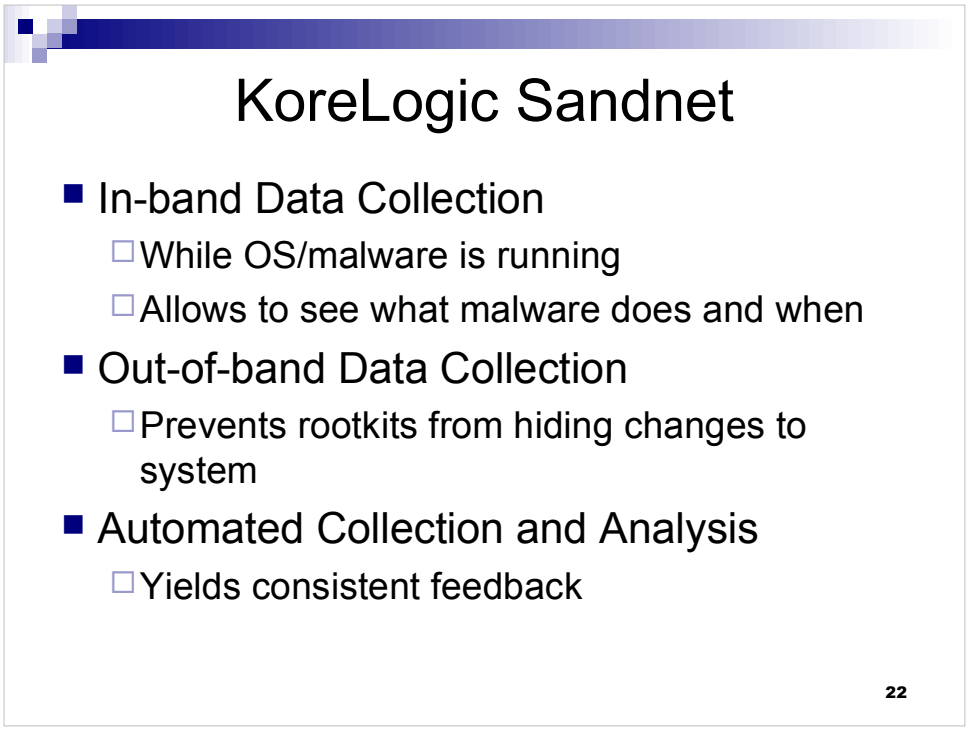
- Configurable
  - Multiple OS' in multiple configurations

21

The Korelogic Sandnet is a physical sandnet composed of two machines – one which acts as the network simulator and local control (sbox) and the other which is the sandbox (lamb). The sandnet was designed in such a way that multiple, distributed sandnets can be controlled by a central server. The central server can submit tests to each of the distributed sandnets and, when the tests have completed, the results are uploaded back to the central server. All communications between all machines are authenticated and encrypted. This allows us to have multiple tests occurring at the same time.

The sandnet was also designed to be extremely configurable. A "dd" image of a sandbox is taken at a specific point in time and labeled as a base image. That base image can be loaded and configured in a different way and then using a custom written binary diff program called "ddp", only the changes from the base image needs to be stored. When that configuration is required, the "ddp" file is then overlaid back onto the base image. The entire process occurs within minutes.

For example, our main malware analysis base image is of a standard Windows XP Pro SP2 system. From the base image we have created multiple "ddp" files which contain different Windows patch levels and with different AV programs installed. By doing so, we are able to test malware against multiple anti-virus programs or software/patch-level configurations – thereby increasing the flexibility of the system.



The sandnet collects data both in-band and out-of-band.

In-band data collection occurs when the OS and malware are running. The data collected during this time period includes all file, registry and network accesses and modifications as well as video of the entire session. This is useful as it allows us to see what the malware is doing and when as well as what processes it calls, what ports it listens on and any files it may temporarily use. This information is not possible to collect in out-of-band analysis.

Out-of-band analysis occurs when the sandbox OS has been shut down and the malware is no longer running. By examining the file system and registry files at this time we can see what modifications have occurred to the system without worry that a rootkit may be hiding their presence. This provides accurate results.

Finally, as soon as the analyst submits the job to the sandnet the entire process – from program execution, data collection and final analysis – is completely automated. This yields consistent feedback for every malware test.

KoreLogic Sandnet

sbox

- Runs network services
- Controls automation
- Stores analysis data

mlamb/vlamb

- Sandbox
- Dual boots between Linux and sandbox OS

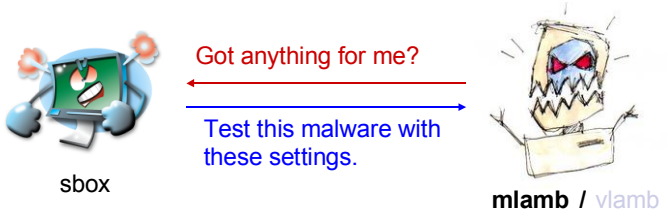
23

The sandnet consists of two machine – connected via a cross-over cable. The machine which simulates network services is called sbox. Sbox runs a modified version of Truman that emulates more services and contains more logging. The automation of the sandnet is also controlled on sbox using an open source program named webjob. Currently, sbox runs Slackware Linux.

The sandbox is run on machine which dual boots between Slackware Linux and the sandbox OS of choice (typically Windows XP Pro). When in Slackware Linux, the machine is known as mlamb (“maintenance lamb”) – in the sandbox OS it is known as vlamb (“victim lamb”).

Images on vlamb are initially installed and then copied using dd. A custom program, known as ddp, allows us to take new images, compare them to the baseline and save off only those bytes which have changed. At any point in time later, we can restore a sandbox to its point of infection.

KoreLogic Sandnet Primer



1. Analyst sets up malware job on sbox
2. Mlamb queries sbox for new test
3. Malware, tools and configuration copied to mlamb
4. Mlamb reboots to appropriate image

24

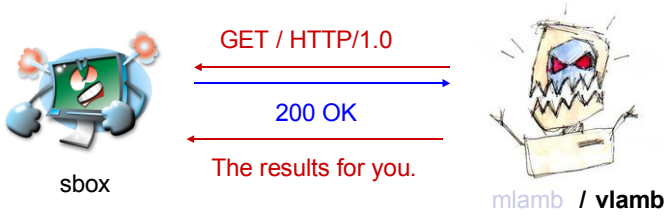
Analysis starts when an analyst submits a malware to process on sbox.

Mlamb, which is constantly querying sbox through webjob for new malware to process, receives a new job and uses scp to copy the malware, configuration files, master control script and latest set of tools to an area which mlamb and slamb can access.

A control script for mlamb runs which copies the correct sandbox OS image (if its not already there) from sbox and applies it to the sandbox. This is done using dd to apply the base image and a custom written program named ddp to apply any configuration changes.

Once completed, mlamb reboots into slamb.

KoreLogic Sandnet Primer



1. Vlamb starts tools, runs malware, stores results and reboots
2. Mlamb sends results to sbox
3. Sbox analyzes results, produces report

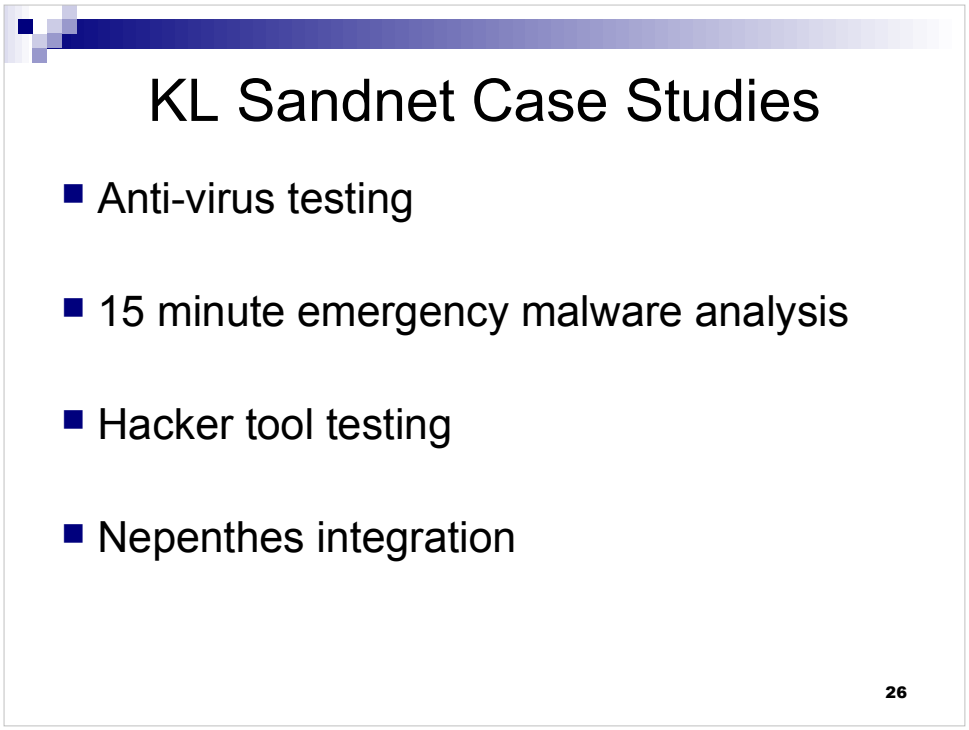
25

When vlamb starts up it finds the master control script and executes it. The master control script starts the monitoring tools and runs the malware. At any point, if the unknown program running attempts to connect to a service, sbox will intercept the connection and respond to it.

Once a configurable amount of time has passed, the control script stops the monitoring tools and saves the results into the shared location. It then reboots back to mlamb.

Once rebooted, mlamb runs its post-analysis procedures. This includes taking a ddp of the image - which allows us to restore the sandbox to the point of infection at any time - and running out of band analysis tools of the file system and registry – which allows us to discover new files that may have been hidden by rootkits.

Once post analysis has been finished, all data files and results are sent back to sbox which stores them on an external hard drive. Sbox analyzes the results, produces a report and uploads it to the central server, gbox.



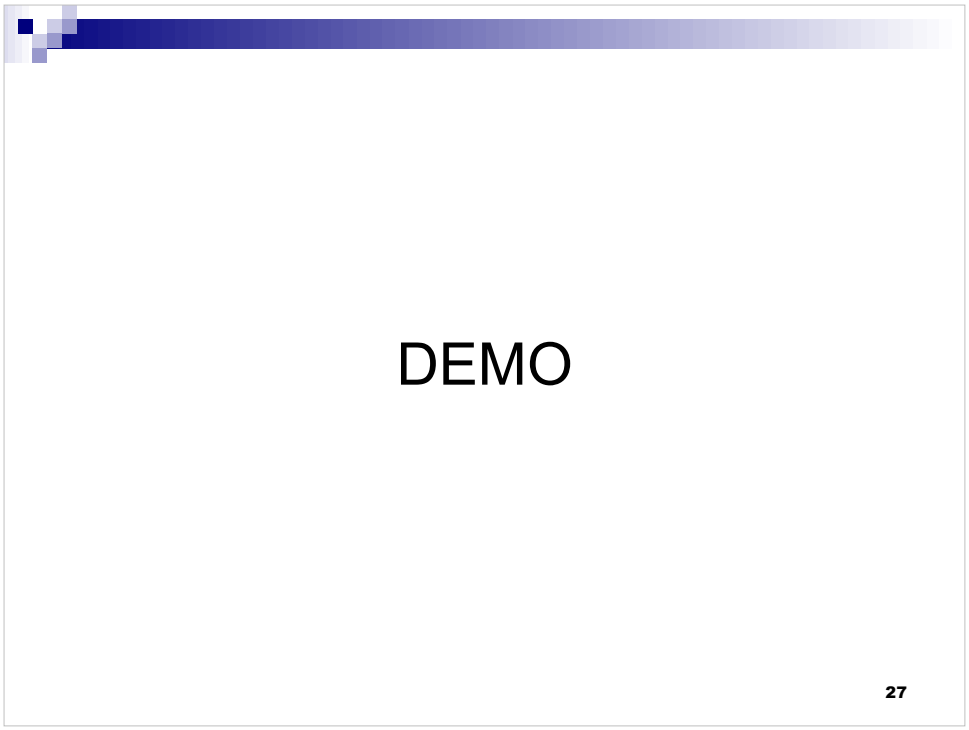
Since its creation, the KoreLogic sandnet has been successfully used for both ourselves and our clients. The following highlights some of the things it has been used for and may give ideas on what alternate tasks sandnets can be used for.

-The original purpose of the sand net was to test a number of major anti-virus programs to see how effective they were at preventing malware from executing. To do this, we created our base image and then a number of ddp images – each of which had a different AV package installed. We then a large number of common, and not so common, pieces of malware through to see how effective the AV software was at preventing the malware and cleaning up the malware's remnants. In doing so, we had 4 sandnets running constantly for over 2.5 weeks – executing over 500 tests.

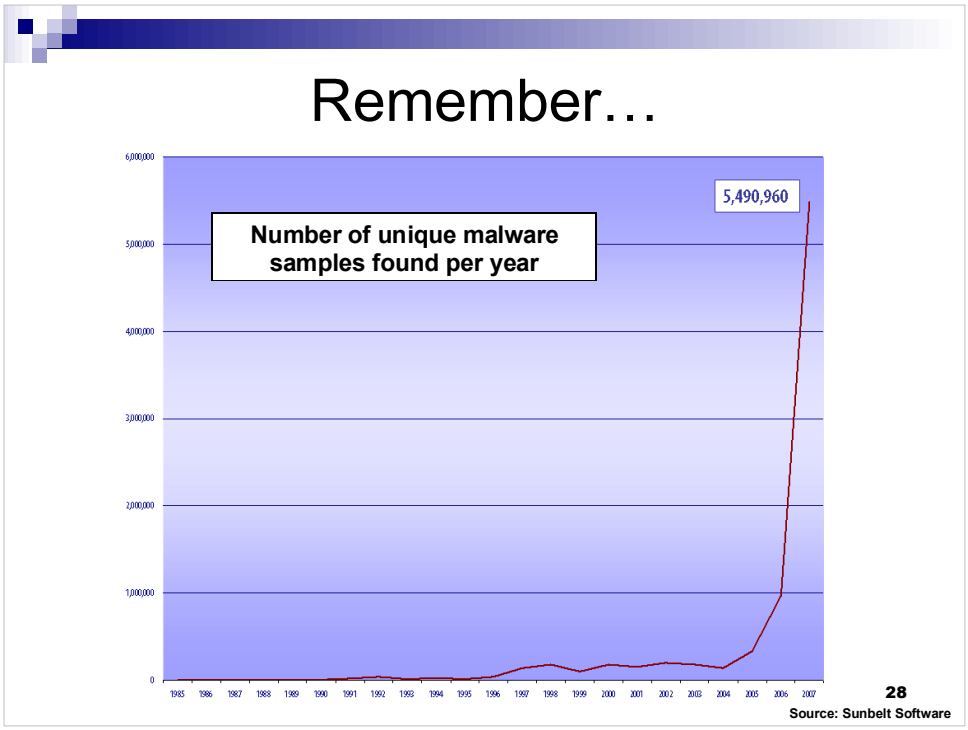
-While onsite at a client for a review, they had a large number of machines infected with malware their AV did not detect. While they had the malicious executable they knew nothing about it and there was no time table from their AV company for a new signature. We asked them to let us look at it and we ran it through the sandnet. Within 15 minutes we had the analysis completed and the results back to the client. They were able to use this information to remove the malware before the AV software had signatures available.

-We often find "hacker" tools on various suspicious websites that promise to do very helpful things. However, before we use them we need to vet them out to ensure they will not compromise our own systems or do anything unintended. In order to do that, we throw the tools into the sandnet and run it a few times. Then, we analyze the state of the system afterwards to see if it dropped any files, threw itself into the registry, attempted connections to external sites, etc. This allows us to be fairly sure we know what the tool is doing.

-Nepenthes (nepenthes.mwcollect.org) is a malware collection honeypot. Essentially, when running on a computer it emulates a number of different, commonly exploited, services and waits for attacks. When these services are attacked, it parses the shellcode and downloads any files the attack would have it get. These files are then placed in a repository for the analysis to later look at. One of the KoreLogic analysts has created a nepenthes plug-in which allows any malware taken to be immediately submitted to our sandnet for analysis. Within 20 minutes we have a full profile on what the malware does and can tell if this is a new variant or one that has been around for a while.

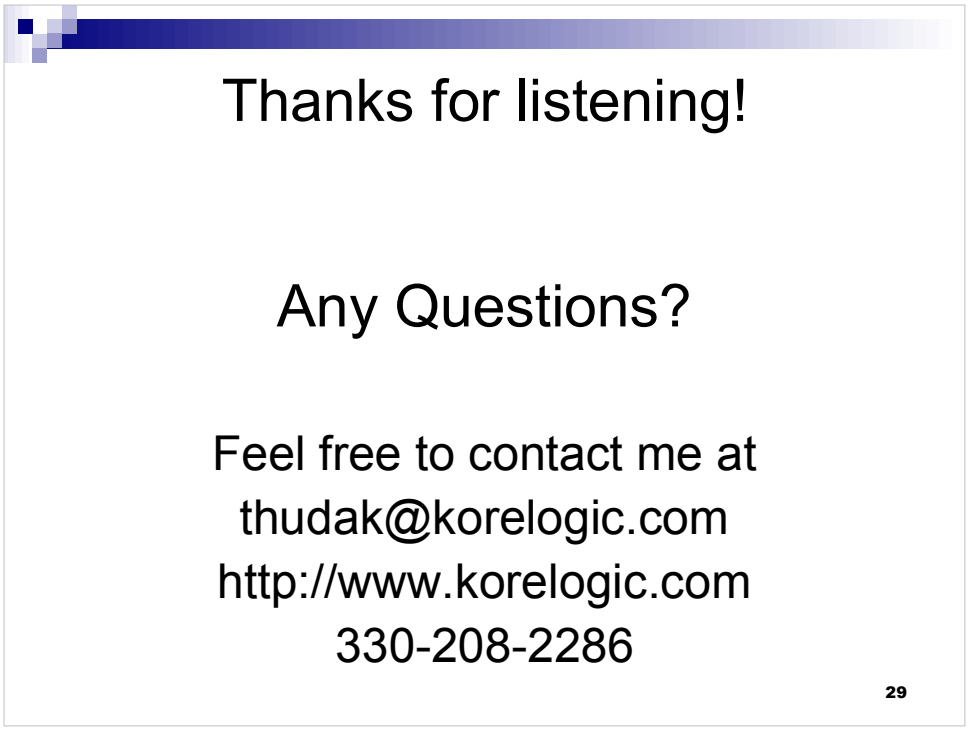


MD5 hash for Generic SDBot used for demo:
11bdf2a9b21166490a4147ef957f2477



I'd like to leave you with this graph once again. Remember, the number of malware in the wild is increasing exponentially each year. Eventually, you will come across some in an investigation. Anti-virus is becoming less effective in the short term, so forensic analysis and incident response practitioners need to develop the skills necessary to analyze malware and determine what they do in a safe, fast and effective way.

The implementation of use of a sandnet will help analysts by allowing them to analyze malicious code in a way which allows them to quickly get their jobs done in a safe manner.



Tyler Hudak
Senior Security Consultant
KoreLogic Security – http://www.korelogic.com
thudak@korelogic.com
330-208-2286