Global Information Assurance Certification Paper

MalwareD

A study on network and host based defenses that prevent malware from accomplishing its goals.

GIAC (GREM) Gold Certification

Author: David R Walters, contact@malwared.org

Advisor: Mark Stingley

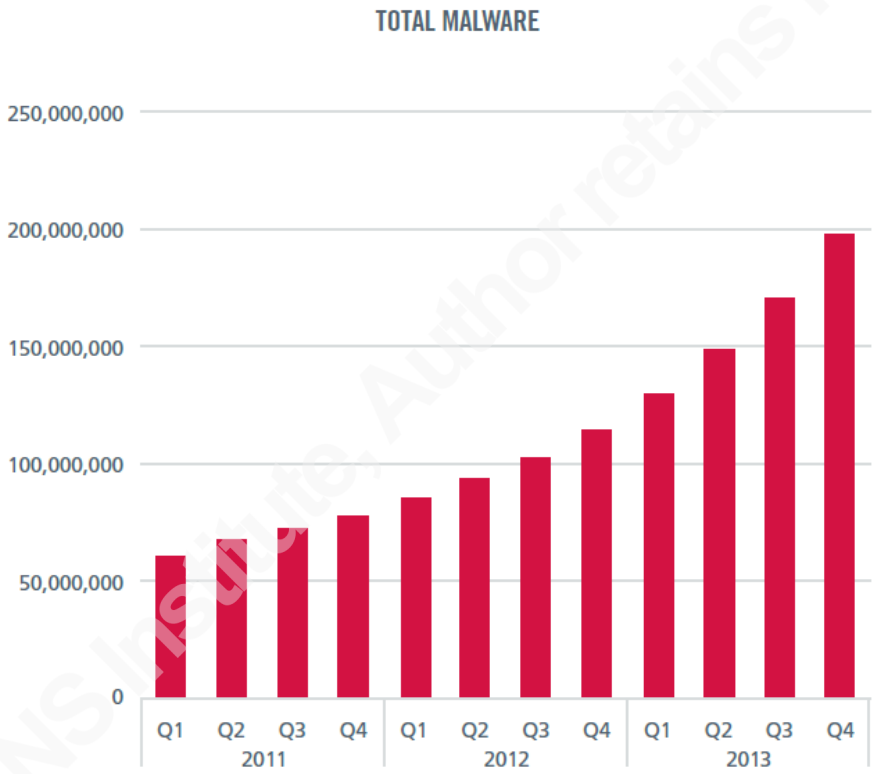
Accepted: September 16th, 2014

Abstract:

Computers pose a risk to companies due to the nature of the information they store. Most organizations battle computer based threats on a daily basis. Malware is the attacker's vehicle. It is diverse, evolving, and capable of any attack a programmer can dream up. Routing, DNS and the principle of least privilege are three critical defenses to combat malicious software. Enterprises who tune these technologies to a more protective stance have a greater chance at successful defense. A test of a few thousand malware samples has demonstrated that these critical defenses work. As a result, a security analyst can better focus their investigations on events that they know have success in their environment.

Introduction:

Malware is an ever-growing problem on the Internet. Organizations struggle to prevent, detect, and responds to malware threats. As networks grow in size and complexity this problem increases every year. The result is an inclining trend in malware. Trends observed by the EC3 (European Cybercrime Centre) across member states in 2013 include substantial increases in intrusions, malware, phishing, grooming, DDoS, espionage, and botnet activity (2014 Verizon DBIR).



Graph 1 McAfee Labs Threat Report - 4th Qtr. 2013

Malware is multi-faceted, diverse, and capable of subverting even the most well defended computer networks. Its prevalence and capabilities are nothing new to the information security community. Any seasoned security analyst would tell you it is highly unlikely that there isn't a corporate network which doesn't regularly exhibit significant indication of compromise. As the variants of malware grow and evolve, so do the number of indicators. There is no shortage of indicators publically available to inform analyst and researchers about a compromise. Malicious IP address lists, malicious domain lists, web proxy categorization lists, A/V signatures, IDS alerts, and behavioral based analysis events are a few examples of tools and technologies that can provide indication that a host is compromised. There are also a significant amount of tools and

services that try to determine if a suspected sample of code is malicious. It is important to note that the goal of these tools is different than the goal of this study. Their goal is to determine malice. This study attempts to determine if certain defenses are effective in preventing malware success.

Behavioral malware environments typically incubate the sample code to execute in a manner that does not always reflect an environment where malware defenses exist. The possibilities to hide the analysis component when only executing in user space are very limited. For example, hiding a process or a loaded library from all other processes running on the system is usually not possible from user space alone. The just mentioned limitation is eased when the analysis component runs in kernel space (Egele, 2008).

This method of execution has a few effects. First, it improves the ability to determine if the code is malicious. Second, it does not consider defenses that are typical in corporate networks. The lack of malware defenses can incorrectly determine if the code was successful in a given network. As a result, analysts are flooded with information and events. This creates a gap between the malware analysis tools and the true workstation within the environment.

This study attempts to bridge some of that gap through testing malicious samples in a behavioral analysis engine built and tuned in a defensive manner. The process of analyzing a given program during execution is called dynamic or behavioral analysis. Static analysis refers to all techniques that analyze a program by inspecting it (Egele, 2008). Commodity malware will take the path of least resistance. This is true throughout the malware lifecycle, from coding to communication with C2 servers. In addition, modern malware samples frequently require some form of Internet access for their Operation (Egele, 2008). *The re-tuning or implementation of a few existing technologies in a more defensive posture can greatly improve network security, thereby reducing organizational risk of intellectual property and personally identifiable information loss as well as reputational damage.*

Malware Defenses:

The focus of this study is to demonstrate controls that prevent malware from accomplishing its goals. The GIAC GREM course educates on the various tools and techniques used to fight malware. It aims to teach analysis techniques that help to respond to incidents, improve forensic analysis performance, and strengthen defenses. This study merges some of the behavioral analysis tools and techniques from the course with defenses learned from experience. Then it attempts to determine the effectiveness of those defenses against malware found in the wild. Properly armed responders and reverse engineers can make better decisions in response situations. In addition, more accurate reporting will result from smarter analysts.

Malware defenses come in many forms. They are not limited to signature based scanning tools on the network or on an endpoint. They are varied and diverse. A corporate network with strong controls should utilize a defense in depth architecture to protect against malicious code. All facets of information security program should contribute to protection. Some of these practices include patching operating systems, forward web proxies, and users operating with reduced privileges. It is critical that malware protection and detection mechanisms be layered and diverse. Relying solely on signature-based tools at the endpoint increases the risk of threats slipping through the cracks. The ability of malicious authors to pack and alter code creates loopholes in corporate networks. On average, they repack specimens every 11 days, and some malware families repack up to twice daily. (Caballero, 2011) One of the most common attack vectors is executed while users browse the web. In other words, from the internally controlled network to the public Internet. Drive-by download is the process by which adversaries infect corporate networks by exploiting users browsing the Internet. Drive-by downloads have become the preferred distribution vector for many malware families (Nappa, 2013).

Technologies, policies, and initiatives that defend against malicious code include:

#	Domain	Description
1	**Management Support**	Oversight, structure, and funding of cyber security programs from managing bodies
2	**Principle of Least Privilege**	Users, services, and systems should run with only the privileges required to perform their jobs
3	**Network Segmentation**	Critical components should be in separate subnets with firewalls in-between
4	**Strict Inbound and Outbound Traffic Control**	Links in and out of the internal networks should be tightly controlled through proxies, firewall rulesets, and routing
5	**Configuration Management**	Systems should be hardened, protected, and monitored according to an industry best practice
6	**Patch management**	Updating software within the timeframe that the business allows
7	**Logging and Monitoring**	Systems, applications, and the network should be centrally logged and monitored for malicious activity
8	**Response Plans**	Once a compromise is detected, the course of action to contain, remediate, recover, and learn from that infection
9	**Signature Based Detection**	Anti-Virus on the endpoints, email, proxy layer

Table 1 - Defense Categories

These are a few examples of controls and technologies mature organizations use in a defense in depth posture. The technologies tested throughout this study include the principle of least privilege, strict inbound/outbound traffic control, and network segmentation.

Network Architecture:

The next few sections describe some basic protocols and architectures. The network diagram depicted in figure 1 serves as a reference to these basic protocols and architectures and should be relatively self-explanatory to any network engineer. There are a few subnets with segmentation between critical parts of the network. The network layer separates the server zone and workstation zone. The stub zone in-between the router and the firewall/proxy has its own subnet. Each segment off the router is isolated at layer 3 to its own broadcast domain. This network architecture is the configuration used to execute all the malicious samples tested throughout this study.

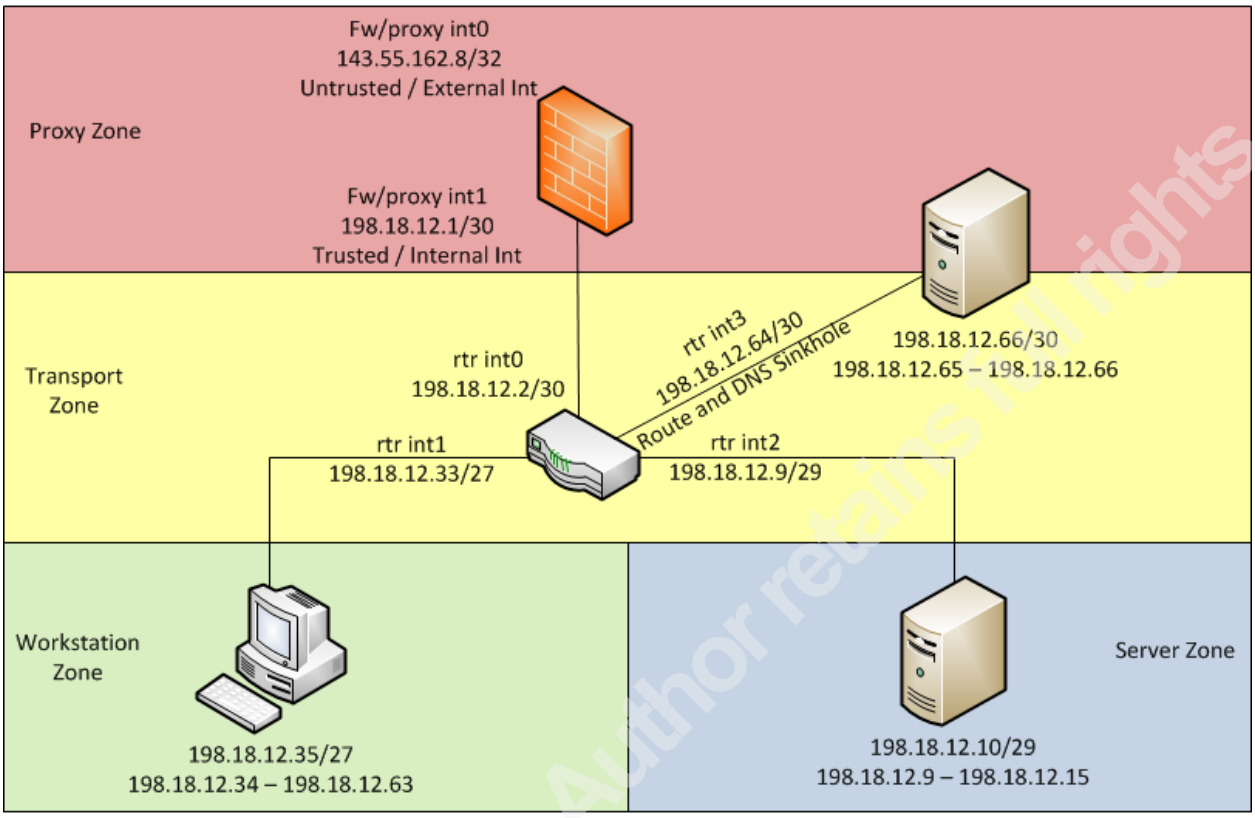


Figure 1 - Network Architecture

DNS:

The domain name system is a protocol designed to map human readable names to numeric addresses, which are more suitable for a computer to process. It is a backbone protocol for today's Internet. The DNS RFC specifies information exchange over UDP port 53, or when larger amounts of information are required, TCP port 53 is used. DNS is relevant to malware because it facilitates communication between compromised hosts and C2 servers. Malware authors program a domain name, or an algorithm to randomly compute a domain name, that maps to an active IP address of a C2 server. This C2 server can order commands, distribute updates, or point to other C2 servers. Frequently changing domain names is a common practice in an attempt to avoid detection. This strategy provides a remarkable level of agility because even if one or more C&C domain names or IP addresses are identified and taken down, the bots will eventually get the IP address of the relocated C&C server via DNS queries to the next set of automatically generated domains (Antonakakis, 2012).

Figure 2 is a typical example of DNS and web browsing in an environment that resolves external domain names. In step 1, the client workstation makes a DNS query to an internal DNS server. Step 2 has the internal DNS server responding with the IP address of the domain requested. In step 3, the client initiates a TCP connection with the IP address that it received from the DNS server. If this process completes, the internal workstation has a full TCP connection with an external and untrusted host. RFC 1035 describes the protocol in more depth.

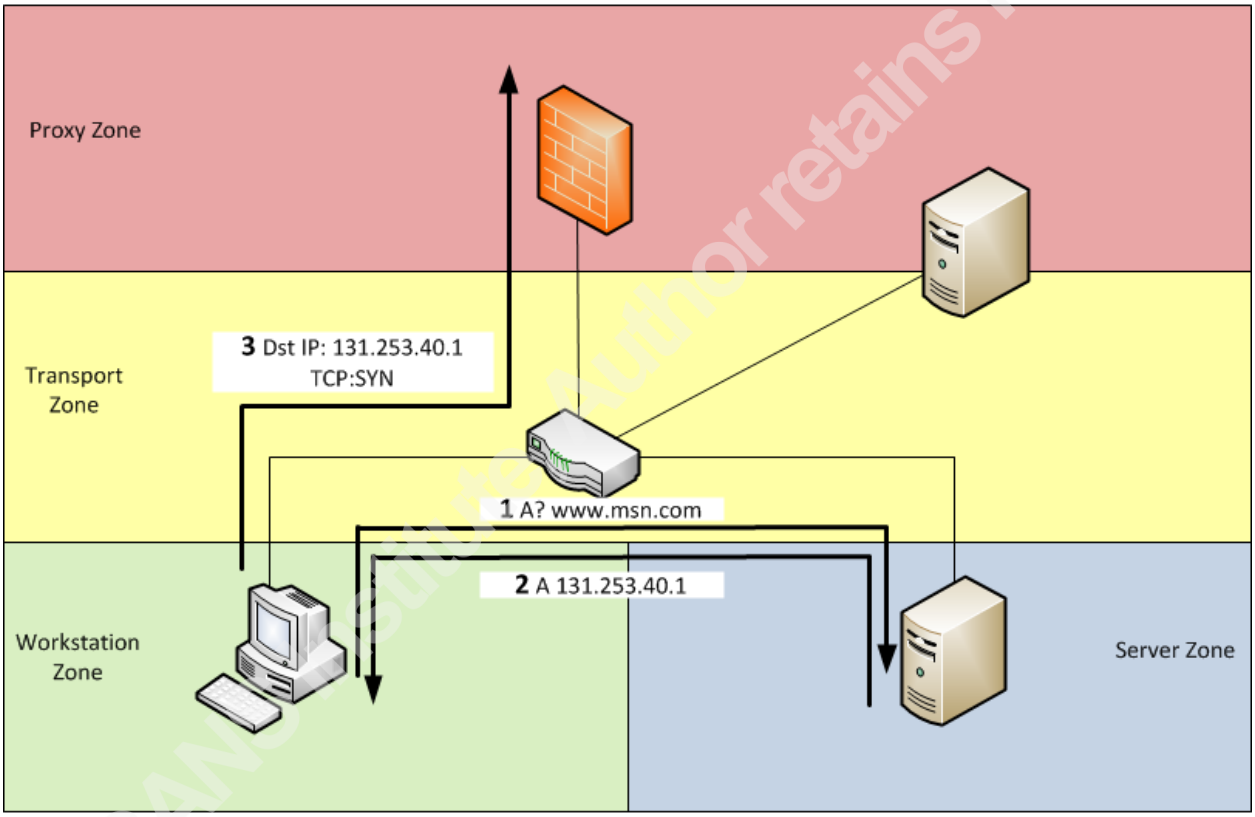


Figure 2 - DNS External Resolution

DNS-Sinkholing:

DNS-sinkholing is where the recipient of a DNS response is given a bogus address. For the purposes of malware, malicious code receives a false address, rendering communication with the originally intended resource unsuccessful. Figure 3 is an example of DNS-sinkholing. The workstation makes a request as normal in step 1. The DNS server receives the request in step 2. The server is pre-programmed to filter the response based on the name requested. This filtering can be through whitelisting or blacklisting depending on the environment. A whitelisting

approach requires additional infrastructure because traffic leaving the internal LAN needs to go through a through a proxy. The proxy layer can then, communicate directly to the Internet resource, at the TCP layer. Step 3 demonstrates the attempted communication between the workstation and the sinkhole server. The server simply logs and discards the packet.

DNS-sinkholing has the benefit of preventing the attempted communication of internal endpoints with untrusted Internet resources. It also has the ability to log and track when these types of communication occur. This logging can help find compromised workstation or unauthorized software. From a malware perspective, DNS-sinkholing can greatly break down command and control channels. Combining a DNS-sinkhole with other defenses increases the ability to detect and prevent compromised assets from communicating with unwanted hosts or domains on the internet (Bruneau, 2010). Further, it can prohibit the ability of malicious code to successfully perform its programmed job.

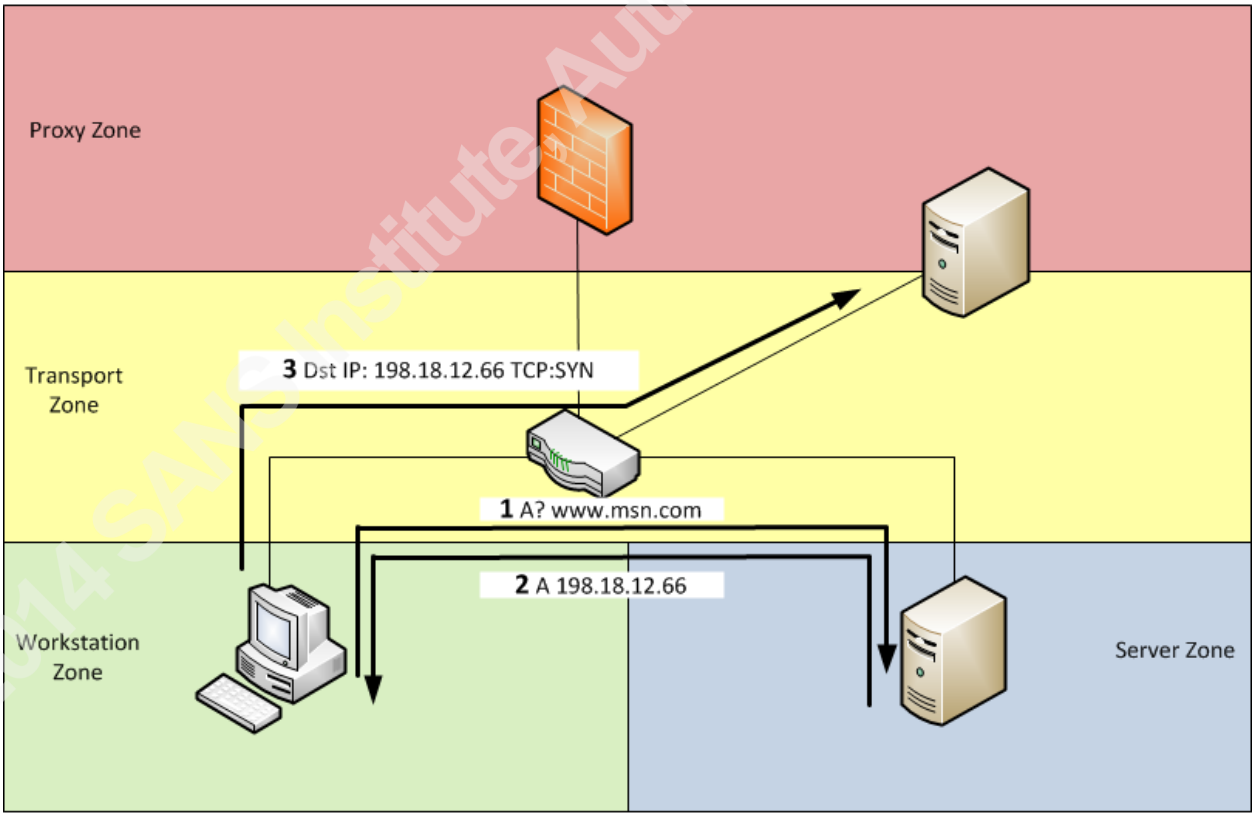


Figure 3 - DNS Sinkhole

Routing:

Routing is the process of moving packets from point A in a network to point B. Destination IP addresses serve as the basis of routing decisions. Routing is a core component of the Internet and every corporate network. Workstations, servers, and routers make routing decisions all the time. Many algorithms make routing decisions based on a multitude of factors. Some of these factors include link congestion, number of hops, or reliability. For this study, the router chose paths solely based on destination IP addresses.

Figure 4 is an example of a typical internal router. The router decides the path for packets to both the corporate and external networks. If external DNS resolution is allowed, the client receives a DNS response with an external IP address. After comparing this IP address with the network portion of its own IP address, the workstation sends the packet to the default gateway. Step 1 designates this process. Once the router receives the packet it must decide how to get the packet to its destination. It performs a lookup for the destination network in its routing table. Step 2 depicts this calculation. The router has all the local subnets configured in its routing table, 198.18.12.0/30, 198.18.12.8/29, 198.18.12.32/27, and 198.18.12.64/30. The last route is the default route. In the case where the workstation attempts to communicate with 131.253.40.1, the router sends the packets out its default route. As pictured in step 3, packets are sent to the firewall/proxy device. This is a common architecture for many enterprise networks and is the most simple to set up.

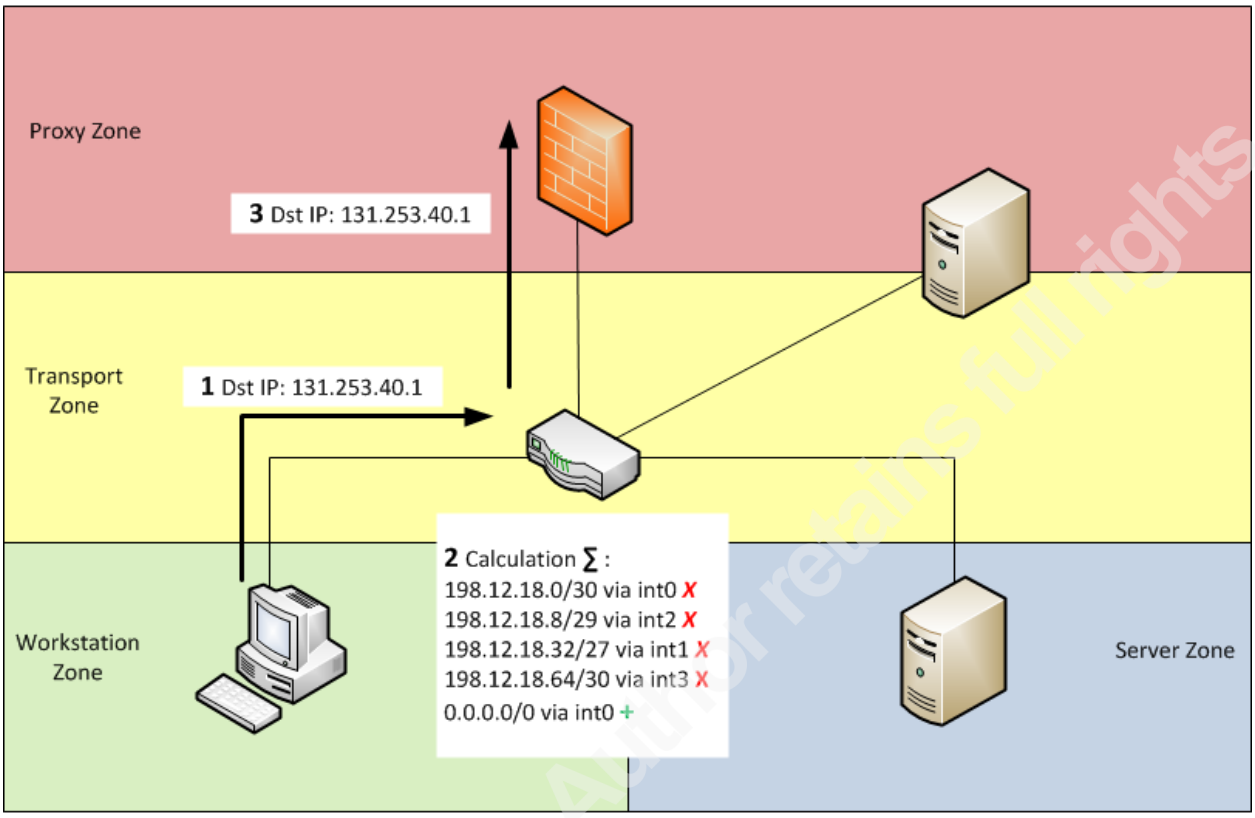


Figure 4 - Routing

Route-Sinkholing:

There are a few methods to implement Route-sinkholing. One method is to insert routes of known bad networks that point to an internal server. Remotely triggered black hole (RTBH) filtering is a technique that provides the ability to drop undesirable traffic (Cisco, 2005). RTBH, as described in the Cisco whitepaper, has very similar goals as the route-sinkholing outlined in this paper. The sinkhole server will simply discard, drop, or log the packets when processing. This method is cumbersome to maintain because routing tables frequently grow to sizes that are difficult to manage.

The second method has the default route configured with an internal server as the next hop. When computed, the packets are logged and discarded. This method has the advantage of reduced routing table entries to manage. In order for route-sinkholing to work, a proxy infrastructure must be in place. This infrastructure interfaces with the outside world. The proxy layer is necessary because internal resources must be able to communicate out to the Internet. In

order to communicate externally, they send the packets to the proxy layer instead of the IP address of the external resource.

Figure 5 is an example of a packet destined for an external IP address. The router is configured to sinkhole all packets for which it does not have a configured route. Step 1 has the client sending the packet to the router, just the same as in figure 3. In step 2, the router performs the same calculation to figure out which interface to send the packet. The result of the calculation has the router sending the packet out the int3 interface to the sinkhole server. The router is unaware that the packet will not reach its true destination. If DNS-sinkholing is used, the only time the router receives a packet with an external destination IP address is when the software has an IP address hardcoded. If the software relies on DNS, then the DNS response would include the configured DNS sinkhole address.

Route-sinkholing is a strong defense when combating malicious code. It will stop all samples that hardcode the IP address in order to communicate to command and control servers. However, it does require additional infrastructure to support external communication. This additional layer provides increased filtering and logging capabilities that greatly enhance awareness and intelligence on internal to external communication.

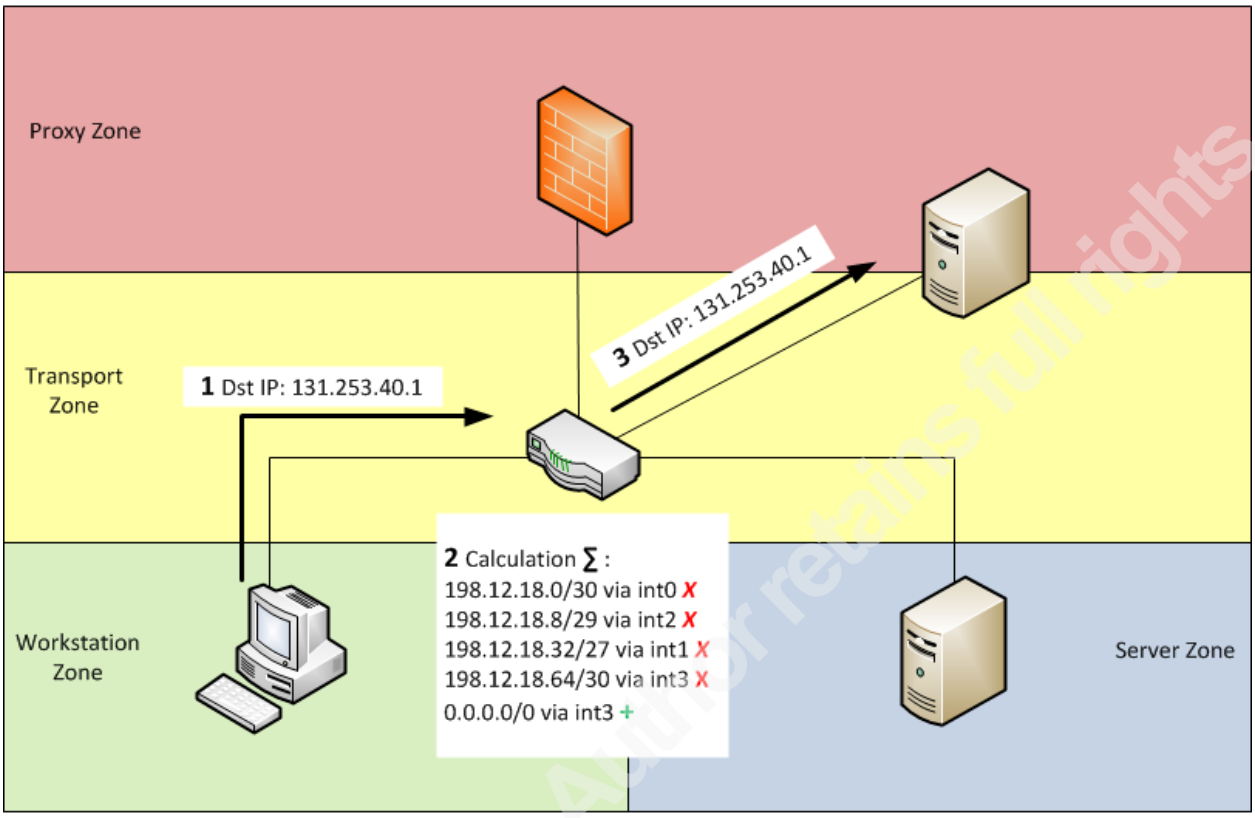


Figure 5 - Route-Sinkhole

Web Browsing Direct:

In an environment where workstations are allowed application layer access to external resources, the request will look something like figure 6. The client has a raw transport layer connection with the external resource. When the client tries to browse to an external website, it first requests the IP address from the DNS Server. Second, it sends the packet to the router. Third, the router sends the packet out its default interface. The diagram in figure 6 depicts an HTTP GET request. This is the typical HTTP method used to request data from web servers. The destination IP address is that of the true webserver and the TCP port is 80. This method of browsing requires that internal routers have default routes configured to allow external access. It also requires that DNS resolution of external domain names be resolved with their true addresses.

This can be considered a less secure browsing method than a proxied browsing environment because of the inability to control the flow and type of data in and out of the organization. Proxy servers support logging individual Transmission Control Protocol (TCP) sessions and blocking

specific Uniform Resource Locators (URLs), domain names, and Internet Protocol (IP) addresses (NIST SP 800-53).

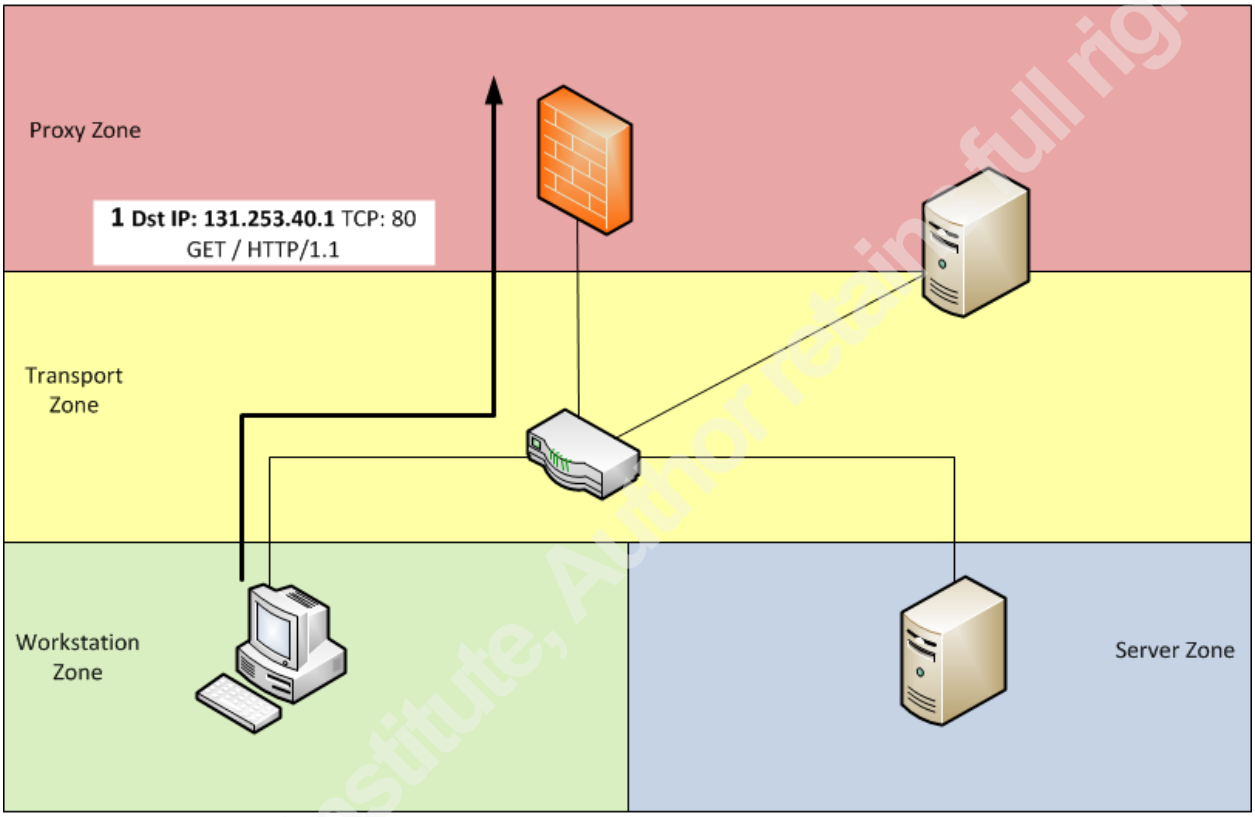


Figure 6 - Web Browsing Normal

Web-Browsing Explicit Proxies:

Web browsing proxies can be implemented a two different ways, explicit or transparent. Transparent is where the packets are intercepted at some point along the path. Typically, at an internet choke point. In a transparent proxy deployment, the user's client software (typically a browser) is unaware that it is communicating with a proxy (Websense, 2014).

Explicit proxies are configurations in client browsers that tells them to only communicate with the proxy in order to reach external resources over the HTTP or HTTPS protocols. In explicit proxied environments the transport layer connectivity is only between the client and the proxy. The proxy establishes its own transport layer connectivity to the true web service. The client is effectively shielded from the outside world. This is an important distinction because it allows

network administrators to sinkhole the route of last resort. In transparent proxied environments, the default route cannot be sinkholed. This is because the client thinks that it is communicating with the real IP of the web service.

Figure 7 illustrates both HTTP and HTTPS requests when explicit proxies are in place. The top request shows a GET request from the client to the proxy for the website www.sdf.org. Notice how the destination IP address is that of the proxy and not of the real IP address of www.sdf.org. The second request is a typical HTTPS request using the HTTP CONNECT method. The client is requesting the proxy connect to ma.sdf.org over port 443 and then pass all subsequent data through, acting like a TCP proxy.

This tunneling mechanism was initially introduced for the SSL protocol [SSL] to allow secure web traffic to pass through firewalls, but its utility is not limited to SSL (Luotonen, 1999). Again, notice how the destination IP address is that of the proxy. In both cases the destination transport layer port is 8080, because this is used as an alternate HTTP port. This is arbitrary and any TCP port configuration works. As long as the service port on the proxy and the client match. If protocol analysis is not being performed on applications layer protocols, arbitrary protocols can be run utilizing HTTP CONNECT. The RFC does not require TLS or SSL be used after the CONNECT method. SSH or SMTP could be used just the same. It can be used as a method of data exfiltration or bypassing filtering engines.

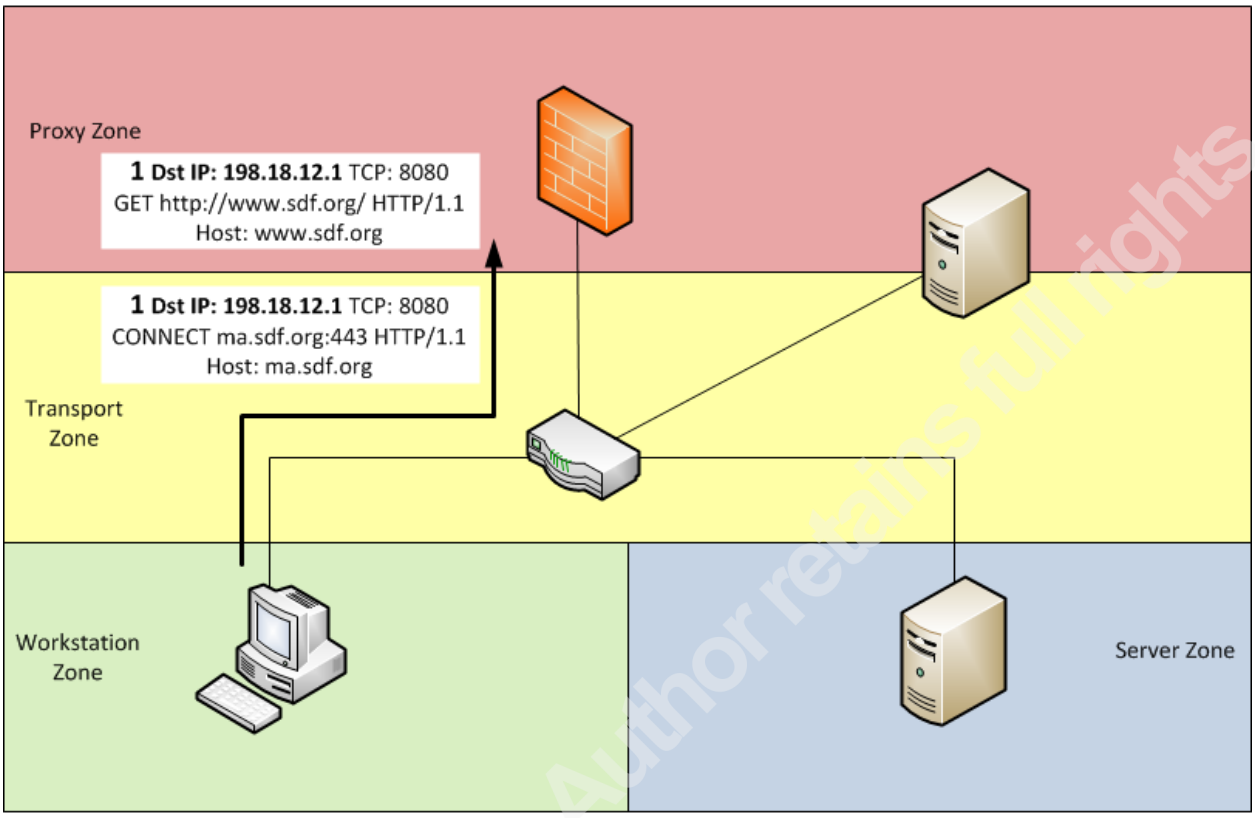


Figure 7 - Web Browsing Explicit Proxy

Putting it All Together:

When used in conjunction with each other, these technologies create a narrow and controlled path for packets leaving the internal zones. There are great benefits to this level of control, from a malware defensive viewpoint. These technologies will not only detect malicious code when it phones home but also unauthorized software that is improperly configured. It can do this because much of this software also phones home to vendor's web sites for updating purposes. If malicious code tries to find its C2 infrastructure through DNS, it will only get a bogus IP. If the IP address is hardcoded, the router redirects the packets to a controlled server that logs and drops the packets. In both cases the malicious code fails to communicate with its desired partners. The only way for malicious code to communicate out to the Internet is through the proxy infrastructure. In order to do this it must be explicitly coded to do so. This requires additional effort on the malware authors part, increasing time and resources required to achieve their goals. In addition, the code footprint increases, which increases the likelihood to detect the malicious code through forensic investigations and A/V signatures.

Principle of Least Privilege:

Every program and every user of the system should operate using the least set of privileges necessary to complete the job (Saltzer, 1975). This is a basic principle that has been around for a long time. However, it is frequently not enforced nor employed. In the Microsoft Windows world this principle can be applied by removing local administrative rights of everyday users. Enterprise environments utilize Microsoft Active Directory (AD) to manage users and groups. Within AD, it is simple to assign proper groups to user roles. However, in larger organizations it becomes challenging to manage and maintain roles and group memberships. This is due to the sheer number of users and roles. Without administrative rights on a system, users cannot install software, create services, or configure administrative settings.

Reduced privileges are highly applicable to malware defense because when malware finds its way onto a system, it typically wants to maintain some persistence by injecting itself into the boot process, installing as a service, or adding itself to the startup section of the registry. Least privilege can be helpful in preventing malware incidents, because malware often requires administrator-level privileges to exploit vulnerabilities successfully (NIST SP800-83). In addition, if the malware is downloaded by a user running with reduced privileges, it will execute with those same privileges.

Table 2 represents a summary of Microsoft Advisories released since 2008. The advisories are broken down by year. The 3rd and 4th columns show the number of advisories per year where Microsoft states "*Users whose accounts are configured to have fewer user rights on the system could be less impacted than users who operate with administrative user rights*". Since 2008, over half of the vulnerabilities identified by Microsoft, could have a reduced impact. A reduction due to a simple implementation of the principle of least privilege. These numbers support that running with limited privileges reduces risk. Reduced risk through privilege management.

Year Advisory Released	Total MS Security Advisories	MS Security Advisories w/Admin Privileges Discussed	Percent
2008	78	48	61.54%
2009	74	39	52.70%
2010	106	60	56.60%
2011	100	48	48.00%
2012	83	44	53.01%
2013	106	50	47.17%
2014	55	22	40.00%
	602	**311**	**51.29%**

Table 2 - Microsoft Security Advisory Analysis – As of September 2014

Malware Analysis Technologies:

Various technologies were used to facilitate this study. A behavioral analysis environment was set up to execute and perform examination of the malicious code. Custom scripts collected samples and configured the environment. Scripts also analyzed the logs produced during the execution. Figure 8 depicts technologies used at each part of the infrastructure. The environment is largely virtual, consisting of two physical machines connected by a switch that is not depicted. The physical machines used are the firewall/proxy and a workstation running Archlinux. The firewall is PFSense 2.1 with squid as the proxy. The virtual host is VirtualBox version 4.3.8 with 2 guests. One guest is a Windows 2008 R2 instance with Active Directory and DNS roles configured. A PowerShell script was developed to sinkhole all top-level domains (TLDs) to the IP address of the sinkhole server. This script can be found for download from www.malwared.org. The other guest is a Windows 7 SP1 instance that is a part of the Active Directory domain managed by the AD server. This machine serves as the malware execution virtual victim. This machine was patched up to December of 2013. Windows defender and DEP have been disabled for the purposes of narrowing down the causes of malware success and failure. The machine was prebuilt to a known good configuration and then a snapshot was taken. At the onset of each malware execution, the Windows 7 machine is reset to a known good state. After the reset, the execution takes place. The user who is executing each sample is running without administrator privileges. The mass execution of malware samples was facilitated by cuckoo malware analysis software. The analysis portion of what cuckoo provides was not used to determine success for this study. The router used to provide inter-subnet communication is

iproute2, which is a routing package built for Linux and supports rule based routing and virtual routing tables. The logical and actual routing tables are depicted in table 3.

Logical representation:	Routing representation:
▪ Server network to proxy layer ▪ Server network to workstation network ▪ Workstation network to server network ▪ Workstation network to proxy layer ▪ Proxy network to server network ▪ Proxy network to workstation network ▪ ***Default to sinkhole***	▪ 198.18.12.8/29 to 198.18.12.0/30 int0 ▪ 198.18.12.8/29 to 198.18.12.32/27 int1 ▪ 198.18.12.32/27 to 198.18.12.8/29 int2 ▪ 198.18.12.32/27 to 198.18.12.0/30 int0 ▪ 198.18.12.0/30 to 198.18.12.8/29 int2 ▪ 198.18.12.0/30 to 198.18.12.32/27 int1 ▪ ***0.0.0.0/0 gateway 198.18.12.66 int3***

Table 3 - Logical and Actual Routing

The sinkhole server is Kali Linux. The log server is rsyslog running on the host machine in addition to an instance of splunk for some increased filtering and export capabilities.

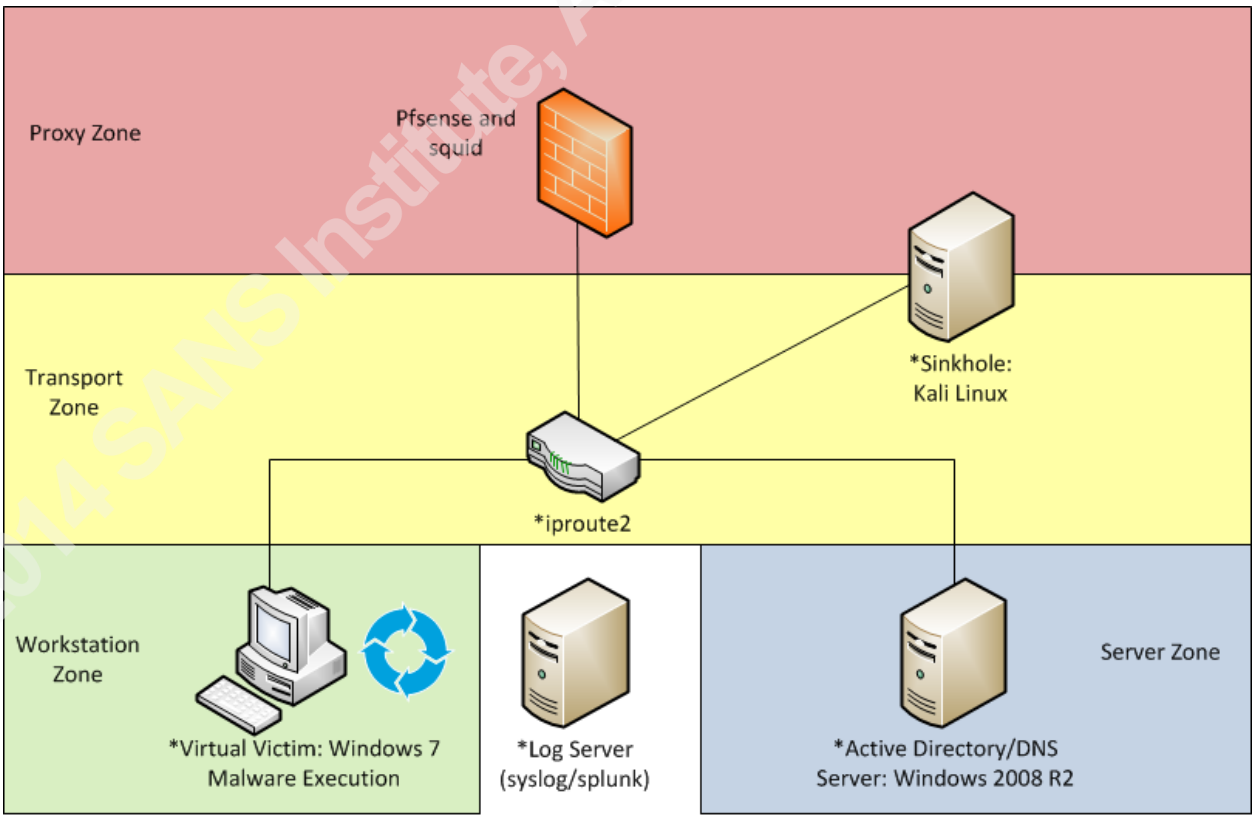


Figure 8 - Analysis Technologies

Collection of Samples:

All of the samples executed are malicious. The process to collect the samples was to download various md5 hash lists from VirusShare. Each of these lists has approximately 130,000 entries. To avoid any challenges whether or not the samples are truly malicious, each list was vetted against VirusTotal. To qualify, the hash had to have greater than 40 of ~50 A/V engines flag the hash as malicious. The number 40 was chosen because it allowed enough samples to be collected in a timely fashion. The files were downloaded from VirusShare after the list was compilied. A list of all hashes can be found at www.malwared.org, along with the scripts used to collect and vet them.

Analysis:

The execution environment had a defensive posture. External DNS resolution was disallowed and requests for external domains would be responded to with the internally controlled sinkhole address. The router's route of last resort was configured with an internally controlled next hop. If the samples tried to communicate directly to an external IP, the packets would never leave the internal network. The Windows 7 virtual victim was stripped of privileges through Active Directory. All samples were executed with domain user privileges. In addition, the Windows 7 virtual victim has had its internet options configured to exclusively use the web proxy. Log events were collected and centralized from every critical point in the environment where possible. Figure 9 represents the log sources collected and analyzed during each execution. Table 4 shows the log source at each point in the architecture along with a short description of the targeted event.

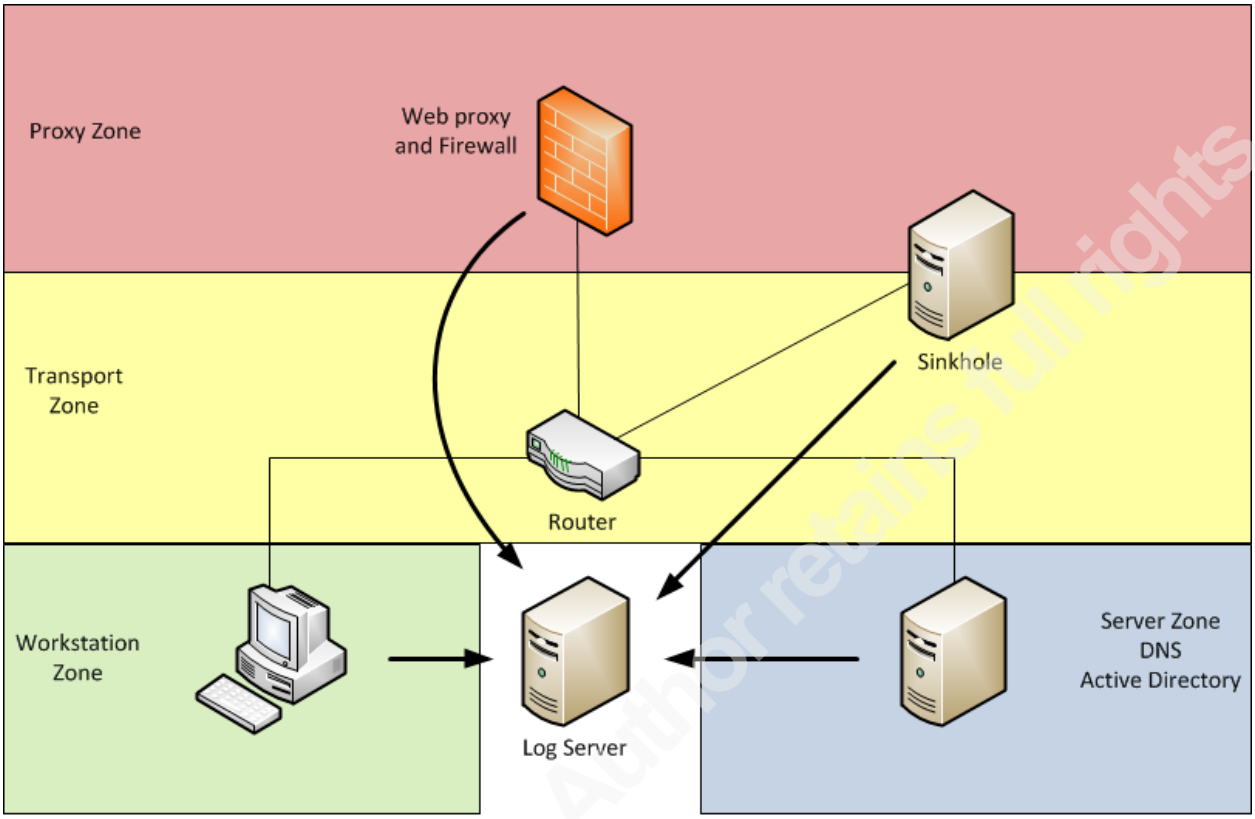


Figure 9 - Event Logging Flow

Log Source	Targeted Event Description
Workstation	Audit and system Logs on the Windows 7 host – shows privilege violations and application crash events.
DNS Server	DNS queries from Windows 7 host – shows attempted DNS resolution of C2 servers by malware.
Route Sinkhole	All traffic from Windows 7 host to any external IP address – shows attempted communications directly to an IP address by malicious code when IP address is hardcoded into the sample.
Web Proxy	CONNECT and GET HTTP requests from Windows 7 host to the proxy – shows the ability to successfully communicate to an explicit proxy.

Table 4 - Execution Event Types

As each sample was executed on the Windows 7 virtual victim, any of the above events could be generated. The following criteria determined if the malicious code was prevented from successfully executing or communicating outside of the local network:

1) If the sample tried to resolve an external domain.
2) If the sample tried to communicate directly to an external IP address.
3) If the sample tried to perform some task that the current user does not have sufficient permissions.

The following criteria determined if the malicious code was successful at communicating outside of the internal network:

1) If the sample was capable of communicating with an HTTP CONNECT or GET request to the forward web proxy.

The result table below shows the summary of all the samples tested. The definitions for each category are as follows:

1) **Success**: Web Proxy event observed.
2) **Explicitly Stopped**: Privilege, RouteSink, DNSSink, or Crash event observed and no Web Proxy event observed.
3) **Unknown but no network IOCs**: No Web Proxy, RouteSink, DNSSink, Privilege, or Crash event observed.

The last category is assumed to be a failure because there are no network indicators observed. It was mentioned earlier that many modern day malicious code samples require some form of network communications. This is sufficient to state that the sample was not successful if no network events occurred. There could be a multitude of reasons why this sample failed. It could have been the wrong CPU architecture, a missing expected DLL, or a failed exploit simply due to patching. If the sample cannot communicate out to the Internet then it cannot leak proprietary code, PII, or organization secrets. Figure 10 is a common example of a multistage attack or a multistage persistent threat. This stage-0 loader is a fairly primitive, hard-coded application whose behavior is fairly limited. All it does is contact some remote machine over a cover channel and load yet another loader, called a stage-1 loader. (Blunden, 2013) This diagram shows that even multistage exploits are thwarted when they do not have the capability to

communicate out to public resources.

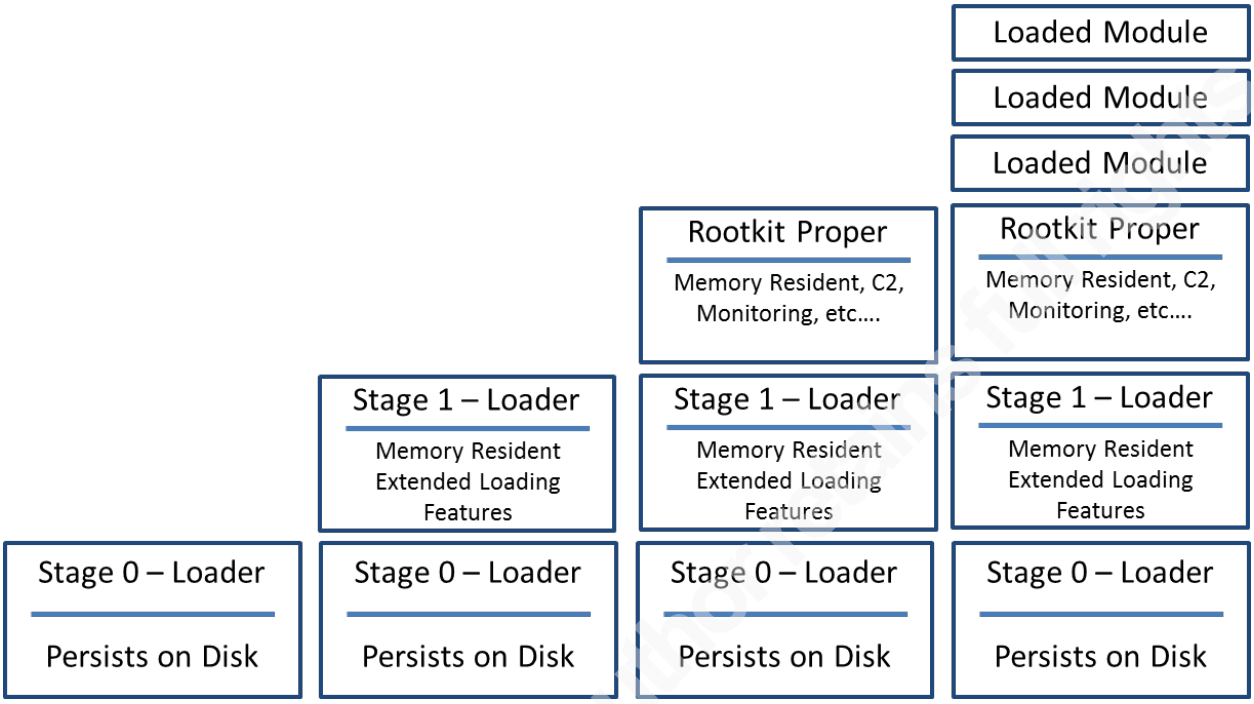


Figure 10 - Multi-Stage Loader (Blunden, 2013)

Category	Count	Percentage
Total	60628	100%
Success	714	1.2%
Explicitly Stopped	9049	14.9%
Unknown but no network IOCs	50865	83.9%

Table 5 – Results

Conclusion:

The results are very interesting, in that, they hint at the fact that commodity malware is just that, commodity malware. It is written for the masses and is expecting limited barriers. It has been very successful over the years. Now, fast forward to the 2014 DBIR. We have more incidents, more sources, and more variation than ever before – and trying to approach tens of thousands of incidents using the same techniques simply won't cut it (Verizon DBIR 2014). It is obvious that new approaches are needed. A more in depth approach should be applied. Malware defenses are varied and should take on many forms, even those that are traditionally thought of as network protocols. Simple improvements to network architectures make a difference. Enterprise environments implement defense-in-depth measures, such as enterprise firewalls that prevent a certain amount of malware from reaching users' computers. Consequently, enterprise computers tend to encounter malware at a lower rate than consumer computers (MSSIR, vol 15, 2013). This study tries to show that breaking into corporate systems is more difficult with stronger controls. The entrants to the game will be limited to those with the most skill and resources if the difficulty is increased, thereby reducing the overall risk that data or intellectual property loss poses to a particular organization. It is still possible to breach any network, regardless of the defenses in place. However, with strong controls and monitored pathways, an organization stands a much-increased chance of detecting the threat and reducing the risk.

References:

A. Nappa, M. Z. Rafique, and J. Caballero. (2013). Driving in the Cloud: An Analysis of Drive-by Download Operations and Abuse Reporting. International Conference on Detection of Intrusions and Malware & Vulnerability Assessment.

Blunden, B. (2013). The Rootkit Arsenal: Escape and Evasion in the Dark Corners of the System. Second Edition. Jones and Bartlett Learning.

Bruneau, Guy. (2010). DNS Sinkhole. Retrieved from https://www.sans.org/reading-room/whitepapers/dns/dns-sinkhole-33523

Cisco. (2005). Remotely Triggered Black Hole Filtering – Destination Based and Source Based. Retrieved from http://www.cisco.com/web/about/security/intelligence/blackhole.pdf

J. Caballero, C. Grier, C. Kreibich, and V. Paxson. (2011). Measuring Pay-per-Install: The Commoditization of Malware Distribution. USENIX Security Symposium.

J. H. Saltzer and M. D. Schroeder. (1975). The protection of information in computer systems. Proceedings of the IEEE, 63(9):1278-1308. Retrieved from http://web.mit.edu/Saltzer/www/publications/protection/Basic.html

Luotonen, A. (1999). Tunneling TCP based protocols through Web proxy servers. RFC Internet Draft retrieved from http://www.web-cache.com/Writings/Internet-Drafts/draft-luotonen-web-proxy-tunneling-01.txt.

Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, David Dagon. (2012). From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware. USENIX Security Symposium.

McAfee Labs Threats Report. (2013). Fourth Quarter 2013.

M. Egele, T. Scholte, E. Kirda, and C. Kruegel. (2008). A survey on automated dynamic malware-analysis techniques and tools. ACM Computing Survey.

Microsoft Security Intelligence Report Volume 15. (2013). January through June, 2013.

NIST. (April, 2013). NIST Special Publication 800-53 Revision 4. Security and Privacy Controls for Federal Information Systems and Organizations Retrieved from http://dx.doi.org/10.6028/NIST.SP.800-53r4.

NIST. (June, 2013). NIST Special Publication 800-83 Revision 1. Guide to Malware Incident Prevention and Handling. Retrieved from http://dx.doi.org/10.6028/NIST.SP.800-83r1

Roberto Perdisci, Wenke Lee, and Nick Feamster. (2010). Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces. USENIX NSDI.

Verizon 2014 Data Breach Investigation Report. (2014).

Websense. (August 2014). Explicit and transparent proxy deployments. Retrieved from http://www.websense.com/content/support/library/web/v75/wcg_deploy/WCG_Deploy.1.3.aspx

http://www.malwared.org – Personal website for this research which includes a few scripts mentioned throughout the paper.