How to test an IPS

Renaud Bidou

RADWARE
renaudb@radware.com

Abstract.

Intrusion Prevention Systems are new technology-based products that intend to detect and block attacks before they reach the target network. As they are to be deployed inline into the internal network they must be tested. However the lack of standard methodology often lower the accuracy of tests, and several issues may be raised once the device is put in production.

Therefore, this document intends to provide methodology and publicly available tools that will help security experts to test IPS in the right context.

Pre-requisites

Before performing tests, it is necessary to understand the purpose of IPS and their mission into the global security of the infrastructure.

Understanding IPS

Mission of an IPS

An IPS is not an IDS. IPS are designed to block attacks and make sure that the part of the infrastructure they protect remain safe. Therefore reporting is still necessary but report's accuracy may be less important than for IDS, as long as blocking is justified.

As an example a tool like [snot] is designed to flood IDS with packets containing attack signatures. Those packets are out of any TCP session and will either be blocked by the stateful of signature engine of the IPS, depending on its setting and design. Then, whatever is reported, the IPS blocked illegitimate packets.

As a full part of its mission, an IPS must be capable to help enforcing specific security policy aspects, such as forbidding *root* loggings on insecure and unencrypted channels. However, this must not be set as a default behavior. Every organization has a different security policy and security devices must help enforcing it. They are not meant to define it! It is then necessary that the IPS is flexible enough to match specific cases of the security policy.

Operations performed by IPS

Three main families of operations are to be distinguished. Their implementation highly depends on needs and all of those capabilities are not necessarily to be tested.

1. Intrusion attempts blocking: mitigating attacks that may lead to privilege escalation and/or remote execution on target servers and applications.
2. DoS mitigation and traffic policing: protect infrastructure from attacks launched to interrupt totally or partially the service and/or flood communication links.
3. Tunnels investigation: detect side and covert channels intended either to make information leak possible, bypass security policy or have compromised system obey to an external system.

Obviously tests must be performed on each and every feature that will be needed.

False positives issues

Policy regarding false-positive must be trivial: no false-positive is acceptable. On the other hand the zero false-positive out-of-the-box can not be guaranteed. Tests have then to be performed with real production traffic in order to identify risks of false-positive and how solving the issue will impact the global security level.

In some cases, operations to remove false-positives may have limited and acceptable impact. However, it may happen that such operations will dramatically

reduce the capacity of the IPS to accomplish its missions, usually turning it into an IDS.

Basics of IPS testing

There are some rules to be followed in order to properly test an IPS, with the objective to evaluate its capacity to efficiently protect the network from attacks.

The need for a context

Defining a context for the tests is mandatory. IPS have a lot of capabilities and they can be tested in a lot of various ways. However, the purpose of the test is to evaluate IPS behavior in a specific environment, with identified needs in terms of protection, reaction and performance.

Testing performance issues at 300 Mbps when the targeted architecture is a 100 Mbps link is useless, time-consuming and will definitely lead to inappropriate results as well as performing tests on the internal network with traffic that will have been blocked by the firewall.

Then the very first task is to clearly define the targeted environment with the following criteria.

1. Traffic typology, which includes type of protocols, size of packets and bandwidth that will pass through the IPS.
2. Protected infrastructure, including the number of protected systems, their nature (OS, application etc.) and services they will be running.
3. Attacks exposure, for each and every system and application, which are the threat the IPS should protect against.

Without such qualification there is no chance tests will be performed in appropriate conditions. Results will then not be reliable and IPS is likely not to properly operate as expected once in production.

Setting up the testing environment

Once the context has been defined, it is necessary to build an infrastructure that will appropriately simulate the targeted infrastructure. A traffic generator, such as [tomahawk], is to be used in order to replay legitimate traffic from capture performed on the production network, at the location the IPS will be implemented. It may be needed to modify part of the packets, usually MAC and IP addresses in order to make sure that packets are correctly switched and sent to the target. [tcpreplay] is well designed for such operations.

Offending traffic will have to be added to this background traffic. Then on one side of the IPS the infrastructure to be protected must be copied, ideally with systems that have identical characteristics. Cost of such laboratory may quickly become unacceptable, and simulation becomes mandatory. The way to simulate the network depends on the kind of threat the IPS will protect.

1. Scanning and worm propagation: [honeyd], which simulates networks, hosts and basic exposure to vulnerabilities on one signe system will be enough.

2. Attack detection and mitigation: systems must be vulnerable to recent attacks, then OS and application must be completely simulated. Tools like [xen] or its commercial equivalent [vmware] make possible to simulate different systems on one single physical server.
3. Denial of Service: in this last case, it may not be necessary to setup the while infrastructure. However, one typical system with identical physical and software characteristics must be setup. In this kind of test it is also mandatory that links have the same bandwidth than in the reality.

Common pitfalls

The most common pitfall in IPS testing is to use IDS testing methodology and tools. IPS are new, and as there is a lot of misunderstanding in their very mission most methodology in testing IPS are just old IDS testing ones moved to IPS. Such misconception of tests usually allows testing of IPS in "non-blocking" mode. This is definitely non-sens. Reporting was critical for IDS as it was supposed to provide information about possible compromise of a resource and lead any manual reaction. Mission of an IPS is to block, and this is this capability that must be tested first.

Another usual trick is either testing out of the targeted context and / or not testing attacks that will have an impact on the context. IPS is designed to block real attacks, not theoretical ones. It is then mandatory to make sure that attacks will have an impact on the targeted system before the IPS is put in-line. This is particularly true for DoS attacks and worms propagation. For intrusion attempts it is useless to test too old attacks, such as the classical phf vulnerability [CVE-1999-0067] which appeared in 1996. Also the use of vulnerability scanners such as [nessus] is not appropriate, as they do not launch real attacks but probes to evaluate the potential exposure to some vulnerabilities.

Last, there is only one good configuration for a given context. Once the configuration has been set up it should not change during the test. It is common sense, configuration will not be changed on the production environment according to the attack we expect to see next.

Testing response to intrusion attempts

Scans mitigation

Scans are usually the first operation performed before an intrusion. Therefore they can be considered as a threat in some conditions.

Types of scans

Two main types of scans are to be distinguished: identification scans and vulnerability assessment scans.

The first kind of scans is done in order to provide information about the nature of the target. Information gathered are the list of hosts on the network, their operating system and applications they are running. They are usually the first operation performed when an intruder attempts to get into the network.

The second type of scan, the vulnerability assessment, is aimed at providing target's exposure to known attacks. Some tools perform large scans over thousands of potential vulnerabilities. Another type of vulnerability assessment tools only tests for one specific vulnerability. These tools are usually developed to quickly evaluate exposure to a new and critical threat.

The need for mitigation

Scan detection and prevention issues

Identifying accurately and blocking scans is probably the most difficult and dangerous task an IPS can perform. Several issues are to be considered with such scans.

1. Most scans are at least part of legitimate operations, making detection difficult as there may be no specific signature;
2. Scans do not necessarily rely on high volumes, and detection based on threshold needs accurate tuning;
3. Source of scans can be spoofed, and then in case of blocking the IPS may block legitimate sources.

Role of the IPS

An IPS primary function is to block real attacks targeted at the network. Therefore, and as long as scans are not attacks but identification operations, and that the risk of false positive is relatively high, the need for blocking scans is not obvious.

On the other hand detection and alerting may seem to have an interest in order to build the complete schema of an attack. However the high number of portscan that may be reported will limit the interest of such operation. IPS will move to IDS and, as

such, will definitely need to be coupled with an external log analysis and correlation tool.

Testing for anti-scanning

There are two main steps in scanning a target network.

1. Identification, intended to discover hosts and services that may be reachable by the attacker. This operation will provide results such has IP addresses and open ports on the target infrastructure.
2. Qualification, performed to provide detail information about operating systems and applications. Such information usually includes name, version, patch level and configuration details.

Identification scans

To detect hosts three techniques are available.

1. Layer 2 identification (only on local network), performed by ARP requests. It can be tested with a tool such as [thcrut];
2. Layer 3 identification, by sending ICMP ECHO REQUEST on all possible IP addresses, with a tool such as [fping].
3. Layer 4 identification, which relies on TCP session establishment and can be performed by any port scanner such as [nmap].

Then identifying running application is usually done by a port scanner. Many techniques can be used. Most of them are already implemented in [nmap]. The most interesting point is to test for different scan parameters, as there is always a way to fool an IPS. Most common options to be tested with [nmap] are :

- -P0 : disable the nmap ping which can easily be detected. It is not necessary as hosts running have been detected in the preceding stage;
- --scan_delay <delay> : tell the scanner to wait a specific amount of milliseconds between each port to be scanned. This operation will usually bypass detection based on threshold.

Usually, scans will be detected but rarely blocked, as explained before.

Qualification scans

The first kind of qualification scan is aimed at detecting OS. Three main techniques are used to achieve this operation. The first one is based on TCP exceptions and initial values (such as the TCP window size), and is implemented in [nmap]. The second one relies on handling of specific ICMP packets, and can be performed with [xprobe]. Last, the analysis of the retransmission delay of SYN/ACK packets can reveal the OS of the server. This technique is made available as a patch for nmap [nmap-cronos].

Then applications versions can be identified in two main ways:

1. Grabbing banners and testing commands, which is a basic technique but still quite efficient and implemented in [amap];
2. Testing for contents of answers to specific requests, as provided by [httprint].

Testing intrusions blocking

Basics of intrusion testing

IPS are made to block intrusions, and there is no reason why letting an attack reaching a target would be acceptable. Therefore the analysis should not be performed on the capacity of detection but of blocking.

As such, sniffers must be set on both sides of the IPS and all traffic related to an intrusion must be detected. It will then provide information about which part of the attack has made its way through.

In terms of blocking the main concern is the different possibilities offered by the IPS. Depending on the nature of the attack, it should at least provide the capacity to:

- Drop the packet;
- Reset the destination of the attack, to make sure that the server is not handling open sessions which are not active anymore. This method should be mandatory to avoid side-effect denials of service generated by sessions left open when attack occurred;
- Reset both source and destination, useful in case of a worm propagating over the network, to make sure that infected servers will not consume all resources on open sessions which have been silently dropped.

In summary intrusions mitigation testing must check the following points:

1. Blocking of real, effective and context related attacks;
2. Sealing of malicious traffic that must not go through the device;
3. Blocking and resetting capabilities.

Generating appropriate exploits

One common difficulty is to find appropriate exploits to test the effectiveness of the detection engine.

It is then necessary to use real malicious programs that make possible to take over a system. The most flexible one is definitely the Metasploit Framework [metasploit], which provides a unified interface to generate recent exploits with different payloads.

In terms of signature database update, it is necessary to test very recent exploits and vulnerabilities. Most of recent exploits can be found either on [frsirt], [securiteam] or [milw0rm] web sites. Testing a few of recent exploits (between 1 or 2 weeks old) is the best solution to evaluate the capacity of the vendor to maintain an up to date signature database.

About vulnerability scanners

As discussed before, the use of vulnerability scanner is not a solution as those tools don't launch real attacks. The issue is the same as the one encountered with identification scans. Blocking such operations would definitely lead to false positives, which is not acceptable.

On the other hand, one may want to be aware of such behavior in order to establish a more global view of a complete attack sequence. Then, and if a decision to couple the IPS with a log analysis and correlation tool is taken, some common scanners

should be tested. [nessus] should be first as it is built as a global framework performing identification, qualification and vulnerability assessment stages in a single tool, potentially relying on other scanning tools such as [nmap].

More specific, vulnerability oriented scanners exists. Common ones are ususaly provided by [eeye] and [foundstone]. However, each time a new vulnerability comes out a scanner is developed to test it. Some research on security web sites such as the ones quoted earlier will usually provide such code.

Testing evasion

There are numerous evasion techniques, organized in four main categories:

- Fragmentation: splitting the attack in different packets
- Substitution: formatting part of the attack in such a way that the server understands it but may confuse the IPS
- Insertion: inserting fake or non standard packets may also confuse an IPS while the server will simply drop them
- Confusion: the use of encryption, especially with SSL, and tunneling technologies will bypass most IPS which are not able to handle such traffic.

Fragmentation

Fragmenting attacks can be performed at two levels. The first one is at layer 3, by splitting one packet in smaller ones. The second one is at layer 4 and is performed by splitting a command or a data in several packets. On top of those basic techniques it is possible to confuse IPS by adding out of session packets or messing established connections. Such techniques are implemented in [fragrouter] which behaves like a router and performs most fragmentation techniques discussed above.

However, it is mandatory to make sure that exploits sent through the fragrouter are effective against the target. Specificities of IP stacks may prevent some attacks from working properly and there will be no use for an IPS to block them.

Substitution

Substitution techniques are mainly used to have HTTP attacks bypass prevention systems. They use several encoding and formatting of malicious URLs to confuse IPS. An interesting point is that some systems are able to handle properly each technique but get confused when they are cumulated. A tool like [nikto] will perform atomic tests for such techniques. [http-mutate] will test them in different combinations.

Insertion

The purpose of insertion techniques is to add fake data in the middle of a data stream and have the IPS handle them while they are discarded by the server. They are usually used in combination with fragmentation and may be designed in different forms. The most common one used to bypass IPS is at the application level. It simply uses overlapping sequence numbers and bad TCP checksums, rarely checked by IPS for performance saving issues. They are implemented and can be tested with [http-insert].

Confusion

Most simple way to test for confusion is to launch an attack on a web server through port 443 (e.g. HTTPS). Specific settings and / or hardware is usually needed to give IPS the possibility to analyze such traffic.

However, depending on the target architecture other confusing techniques may be used such as the use of GRE tunnels or IPv6/IPv4 encapsulation. There is no mean to test for such protocol is they are not implemented into the target infrastructure. On the other hand it becomes mandatory to validate the capacity of analysis if such protocols can be found on the network, especially if some gateways (that could be use by an attacker to terminate its tunnels) can be found on the internal network.

Denial of Service and Traffic policing

Mitigating Denial of Service attacks

Issues in testing DoS attacks mitigation

In order to test for DoS attack mitigation it is mandatory to understand the different types and effects of such attacks. A detailed description is given in [dos].

However, once again it is critical to stick to the context of the target platform and have tests performed in a similar one. As an example it is useless to test for a SYNFlood on port 135 when this port is blocked by the firewall. It may seem obvious in some cases but it is sometimes more difficult to simulate a 10 Mbps link on a LAN, when this is precisely the bandwidth of the uplink to the Internet.

Another difficulty is to evaluate the efficiency of those attacks in a simulated environment. When the target system or application crashes success of the attack is obvious. Same when CPU utilization reaches 100%. But in many cases results are in between and it is also necessary to evaluate if a legitimate client can get access to the application and data in acceptable conditions, depending on the nature and the criticity of the application.

Last, simulating a distributed source of attacks is relatively difficult for session-based attacks, relying on operations performed after a TCP session is established. In such case it is no longer possible to use spoofing capabilities of most packets generators as the TCP session must be completely established. The only way is either to use multiple sources or to use [xen] to have multiple systems running on a single computer. However, the real impact of a powerful botnet will not be reached and efficiency of mitigation upon multiple source will have to be evaluated from obtained results.

Generating DoS attacks

Depending on the nature of the attack two main categories of tools are to be used: packet generators and clients-like software.

The first type is used for most packet-rate based attacks, such as SYN, anomaly based or UDP floods (refer to [dos] for explanation and details). [hping] is a great tool to simulate such operations and can be used at some point to control the rate of packets sent over the network. Many options are available to increase the payload, create bad or abnormal packets. As an example flooders using short TCP header (header of 20 bytes) can be simulated as is with no option. On the other hand correct TCP SYNs can be generated simply by using the --tcp-timestamp option.

The second type of tool simulates legitimates client in the same way DDoS agents would, creating legitimate connections. This kind of tool will make possible to simulate session pending attacks and data floods, only aimed at flooding uplinks.

[webdevil] will test session pending while [http-flood] will continuously establish multiple connections on the home page.

Getting the results

Traces of attack mitigation on the IPS in one thing. Effectiveness of mitigation is another. If no connection is made available when the IPS pretends attack is under control it may mean that IPS operations are not appropriate, but it may also point out a global misconception of the security architecture design. The best example is the one of an uplink on which a SYNFlood is launched. The IPS may behave properly and protect the server, if the link before the IPS is filled up with the offending SYNs, connection will not be possible to the server anyway.

Once architecture is considered as appropriate it is necessary to simulate the behavior of a legitimate client and evaluate the impact of the attack and the mitigation techniques on operations, either in terms of performances or in terms of miss or errors. Variation in results may appear considering most DoS mitigation technologies rely on threshold. At first some connections may be lost or performance sensibly lowered. Once the protection mechanism is activated, performances should improve, but they will rarely reach the level they were at before the attack. For HTTP a tool like [openload] will provide simple and reliable results.

Traffic policing

Reducing the impact of attacks

Limiting the overall bandwidth dedicated to an application is a solution to reduce the impact of data flooding attacks. Then in worst cases, only the targeted service will be downed, or have its performances drastically lowered, while critical traffic, such as voice over IP will keep on running, unaffected by the attack.

Before testing traffic policing it is mandatory to evaluate which traffic is critical and which bandwidth must imperatively be reserved to it. It is important to keep in mind that limiting the bandwidth of a specific type of traffic may lead in dropping legitimate packets. The use of TCP retransmission mechanisms will keep connections alive, but performances on this specific traffic will be lowered. In this case traffic policing is a “last chance” operation and threshold must be set high enough not to impact normal operations.

Tools used for application layer DoS as well as packet crafting tools can be used efficiently to create this kind of attack. In order to evaluate efficiency of policing it is necessary to use specific tools to measure performance of the traffic of critical applications. Then [openload] can be used for HTTP when [sipp] will appropriately provide performance information about SIP protocol and [opensta] will focus on CORBA based application performance. Other tools can be downloaded from [testing].

Simulating worms

Effective worms propagation simulation implies that two main aspects are taken in account. The first one is the vector. This is the exploit used by the worm to compromise the target. It is a basic intrusion test to be performed according to techniques described above. The second effect is the load on the network generated by the propagation itself.

The capability to reduce such traffic must be tested on the IPS to evaluate its capacity to protect internal network from flooding. Based on [worms], 4 types of propagation are to be tested:

- TCP based : attempting to establish as much TCP connections as possible, up to the maximum available bandwidth. [witty] was one of the most lethal one.
- UDP based : same as TCP based but just sending out UDP datagrams. The most famous is [slammer].
- Mass mailing worms : propagating through SMTP. Resolution of MX entries require a huge number of DNS requests. Load is then a mix of TCP (SMTP) and UDP (DNS). [sobig] is a famous one.
- ICMP generators : In order to optimize their propagation, some worms such as [welchia] do *ping* IP addresses before attempting connection. At a large scale the ICMP traffic quickly floods the network.

Specificity in traffic policing tests is that real exploits or worms are not to be launched, it is just a matter of generating traffic and checking for appropriate bandwidth reservation of predefined business critical streams. A tool such as [hping] with the *--flood* option will help in generating such traffic while performance testing tools mentioned above will be used to evaluate the results.

Tunnels Investigation

Covert Channels issues

What are covert channels?

Covert channels are hidden communication streams intended to bypass security policy at different level:

- Make unauthorized procotol connections possible;
- Allow sensitive information leak;
- Create bi-directional communication tunnel between internal and external systems.

There are different techniques to create such channel as described in [covert]. The most common is to use legitimate or unsuspicious traffic to embed data or commands. ICMP used to be fairly exploited in such way with tools such as [itunnel], as it is not always filtered, especially for the ECHO REQUEST and ECHO REPLY messages. At upper level HTTP, SMTP and DNS (UDP/53) are also often used as the security policy usually let this kind of traffic out of the internal network.

Implementation may differ. Sometimes communication stream is just built over TCP or UDP ports belonging to such applications. A simple tool such as [netcat] is able to create such channel. The major drawback is that at application level the communication is not compliant to protocol standards. More specific tools, like [httptunel] have been designed to embed data into completely legitimate HTTP traffic, making detection far more difficult.

Another way to create channels is to use specific fields of layer 3 or 4 protocol. As an example the IPID (use in [stegtunnel]), provides two bytes of data that can be discretely transferred within a simple SYN packet. Options, source ports, checksums, and when using reflection source and or IP addresses are also fields usable to create channels.

Testing covert channels prevention

Tests for covert channel prevention must be performed in two ways:

1. Test for application layer based channels (including ICMP);
2. Test for network anomaly based channels.

Of course, in order to get relevant results, tests must be done according to the security policy and the context of the IPS implementation. As an example, there is no need to test for tunnels above port 23 if telnet is blocked by the firewall.

Application based channels

Typically, this type of channel is created between a client (located in the protected network) and a server (outside the protected network). Testing non standards compliant tunnels can be done for any application above UDP or TCP thanks to [netcat]. Once the server is launches (*-l* option) client must try to connect and send data entered on STDIN. If the server gets the data, the tunnel made its way through. Otherwise it has been blocked.

The same mechanism is to be used in order to test for application compliant tunnels, except that specific tools are to be found for each and every protocol. As mentioned above [itunnel] and [httptunnel] are appropriate for ICMP and HTTP testing.

Network based channels

At network level, channels are created thanks to exploitation of specific header fields. There are few possibilities to detect the initiator of the tunnel, as it relies on authorized network level protocol and manipulates legitimate fields. However, at the terminator level it may be possible to detect and block traffic when reflection is in use.

Simulating this kind of traffic is possible with packet crafter such as [hping] or [sing] for ICMP. Set on the unprotected part of the network the following packets should be generated and sent to a system on the protected network and blocked:

- TCP SYN/ACK with source port matching an outgoing traffic authorized by the security policy (e.g. 80 for HTTP);
- TCP RST with same characteristics as above;
- UDP with source port and destination port 53;
- ICMP replies and errors.

Blocking must be checked at the level of the internal system as some packets may be dropped by the firewall as most of them are typical violation of stateful mechanism.

Encapsulation and encryption

Basics about encapsulation and encryption testing

Encapsulation and encryption tunnels are created either to provide privacy to data transmitted over a shared network or to simulate extension of the LAN over a WAN. SSH, SSL and IPSec are common tunnels of the first kind while MPLS, GRE or IPSec tunnel mode are typical implementations of the second one. Encapsulation and encryption are technically different, but can be used in the same way to bypass security policy. Therefore testing techniques, methodology and expected results are the same.

The main issue with such tunnel is that they are part of the legitimate traffic and, as such, are authorized by the security policy. However, in each case, malicious data can be transferred and avoid blocking by IPS. Therefore it is mandatory, when tunnels are to be expected on the network, to test the capability of IPS to block threats embedded into authorized protocols.

Another important point to notice is that in many cases a system that should be the destination of a tunnel can be turned into a gateway to reach systems belonging to the internal network on any application port. This is particularly true with SSH and SSL servers. Then it is important either to test protection against encrypted malicious traffic and tunneling through encrypted traffic, as it could be the second step of an intrusion.

As an example a web server that can be reached through SSH, could have been compromised because of a vulnerability in the HTTP server. Then the SSH daemon could have been restarted with appropriate options so that connection to SSH service on the web server redirect the traffic to the (supposedly) protected SQL port of a database server.

Testing methodology

As this kind of tunnel is only another vector for transferring data, all the tests realized without tunnels have to be done once again with tunneled data. This is particularly important as tunnel analysis is usually performed by different piece of software (or even hardware for performance issues). Therefore there is no guarantee that DoS protection or anti-evasion techniques behave the same way (or even are implemented) when tunnel investigation is set up.

In terms of tools, the best way is to create tunnels with the same devices or software than those used in the production platform. It is very important as depending on nature and configuration of the tunnel the quality of investigation may differ.

Addendum

Typical testing architectures

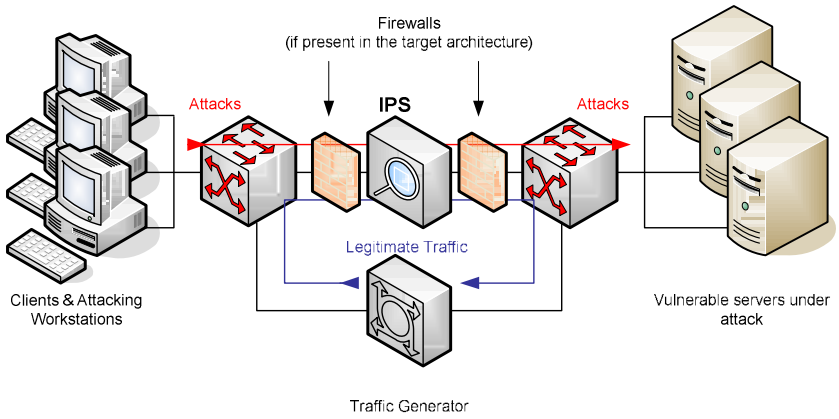

Fig. 1. Generic Intrusion test architecture

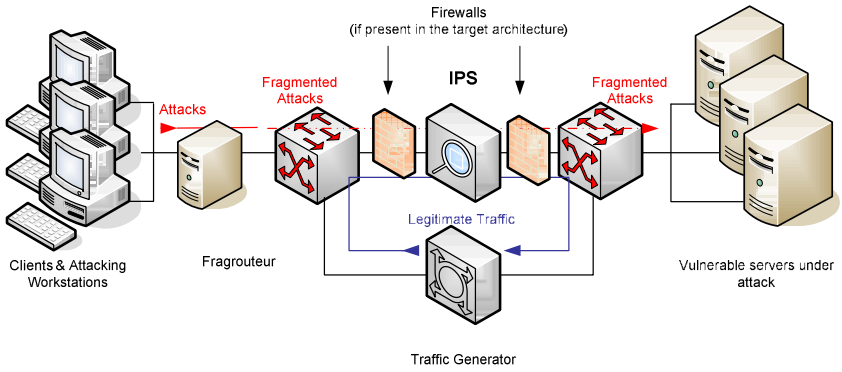

Fig. 2. Fragmentation intrusion test architecture

Resources

Documents

[covert] Covert channels – http://www.iv2-technologies/~rbidou/covert_channels.pdf
[CVE 1999-0067] phf remote command execution vulnerability - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0067
[dos] Denial of Service – Understanding Attack Techniques - http://www.radware.com/content/document.asp?_v=about&document=5613
[sobig] A study of mass mailing worms - http://www.ece.cmu.edu/~jmmccune/papers/worm04.pdf
[testing] Open source performance testing tools - http://opensourcetesting.org/performance.php
[welchia] Virus Analysis 1 - http://www.peterszor.com/welchia.pdf
[witty] The spread of the Witty worm - http://www.caida.org/analysis/security/witty/
[worms] Worms Propagation – Traffic typology

Tools

[amap] Application identification tool – http://www.thc.org/thc-amap
[eeye] eEye Tools - Multiple attack specific vulnerability scanners - http://www.eeye.com/html/Research/Tools/
[fragrouter] Fragrouter 1.6 – IP fragmentation and insertion tool - http://packetstormsecurity.org/UNIX/IDS/fragrouter-1.6.tar.gz
[foundstone] Foundstone Scanning Tools – Multiple attack specific vulnerability scanners - http://www.foundstone.com/resources/scanning.htm
[fping] Ping sweeper – www.fping.com
[honeyd] Virtual Honeypot – http://www.honeyd.org
[hping] hping3 – Multi-purpose packet crafter – http://www.hping.org
[http-insert] HTTP insertion technique – http://www.iv2-technologies.com/~rbidou/http-insert.tar.gz
[http mutate] HTTP confusion tool – http://www.iv2-technologies.com/~rbidou/http-mutate.tar.gz
[httprint] Web server fingerprint based identification tool - http://net-square.com/httprint/
[httptunnel] Full HTTP compliant tunnel - http://www.nocrew.org/software/httptunnel.html
[itunnel] ICMP tunneling tool - http://packetstormsecurity.org/UNIX/penetration/itunnel-1_2.tar.gz
[metasploit] Exploitation Framework – http://www.metasploit.org
[netcat] GNU netcat 0.7.1 – The network Swiss army knife - http://netcat.sourceforge.net/

[nikto] Web server scanner - http://www.cirt.net/code/nikto.shtml
[nmap] The network mapper – Port scanner and OS fingerprinting - http://www.insecure.org/nmap
[nessus] Vulnerability scanning tool – http://www.nessus.org
[openload] Web server performance testing tool - http://openload.sourceforge.net/
[opensta] Open Systems Testing Architecture - http://opensta.org/
[slammer] The spread of the slammer/saphire worm - http://www.cs.berkeley.edu/~nweaver/sapphire/
[sing] Send ICMP nasty garbage - http://freshmeat.net/projects/sing/
[sipp] SIP testing tool and traffic generator - http://sipp.sourceforge.net/
[snot] IDS flooder - http://dc.qut.edu.au/pub/download/TGZ/snot-0.92a.tar.gz
[stegtunnel] IPID based covert channel - http://www.synacklabs.net/projects/stegtunnel/
[tcpreplay] Pcap editor and replay tools - http://tcpreplay.sourceforge.net/
[thcrut] Network layer information gathering tool - http://thc.org/thc-rut/
[tomahawk] Bi-directional traffic replay tool - http://www.tomahawktesttool.org/
[vmware] Virtual machine – http://www.vmware.com
[webdevil] Pending sessions DoS generation - http://packetstorm.linuxsecurity.com/distributed/webdevil-v1.tar.gz
[xen] The xen virtual machine monitor - http://www.cl.cam.ac.uk/Research/SRG/netos/xen/
[xprobe] Xprobe 2 - ICMP based OS fingerprinting - http://xprobe.sourceforge.net/