Detection of illegal gateways in protected networks

Risto Vaarandi and Kārlis Podiņš
Cooperative Cyber Defence Centre of Excellence
Tallinn, Estonia
firstname.lastname@ccdcoe.org

1. Introduction

In this paper we discuss possible solutions for illegal gateway detection in protected networks. We describe several techniques for addressing this problem, and also provide pointers to a number of commercial and open-source products that can be harnessed for illegal gateway detection. Although there are some promising patents in this area (e.g., see [1, 2, 3]), the relevant scientific research seems to be scarce (to the best of our knowledge, there are no solid journal or conference papers published on the topic). For this reason, the paper focuses on industrial-grade software solutions and practical detection techniques.

The remainder of the paper is organized as follows – section 2 provides the problem statement, section 3 is the main part of the paper which describes a number of techniques and software solutions for illegal gateway detection, and section 4 concludes the paper.

2. Problem statement

During the last decade, the complexity of computer networks has increased considerably – both in terms of size and technology. In today's networks, the administrators often have to manage a large number of devices with different operating systems, and human errors during the device configuration are not uncommon. These errors could not only cause network malfunctions but could also open illegal routes to public networks. Moreover, during the last decade the user laptops and workstations have evolved from mere desktops into devices with a routing capability. For example, consider a scenario where a laptop is connected to a protected network via Ethernet interface, and the user also connects his/her laptop to a public WiFi network, thus opening an opportunity for data exchange that bypasses network perimeter defense. This scenario illustrates that illegal gateways are easy to set up, and they pose a considerable security threat to protected networks.

In this document we restrict our discussion to TCP/IP networks and define the illegal gateway in the following way – illegal gateway is a device with multiple network interfaces which is connected to the protected network, where at least one interface is also connected to a network outside the protected network. Note that the presence of an illegal gateway does not necessarily mean that a data leakage or illegal packet exchange will occur (e.g., a laptop from the protected network could accidentally connect to the Internet without sending or receiving any packets).

In this paper, we consider two major classes of illegal gateways. First, illegal gateway can act as an IP level (Layer 3) gateway. Second, illegal gateway could forward data at the application layer, without having IP level routing capability, and run a proxy for a certain application layer protocol. For example, illegal gateway could act as an HTTP proxy which mediates HTTP traffic between the

protected and outer networks.

We also consider three different classes of methods for detecting illegal gateways:

- detection by monitoring of the creation of unwanted interfaces to outside networks,
- detection by monitoring illegal traffic that goes or comes to/from illegal gateways,
- detection by scanning the network and nodes for illegal open proxy ports, unexpected network routes, unexpected device types, etc.

3. Methods for detecting illegal gateways

3.1. Monitoring of the creation of illegal interfaces

The creation of an illegal interface to an outside network is a prerequisite for creating an illegal gateway, since otherwise no data transfer to/from unauthorized networks can occur. Monitoring of the creation of illegal interfaces can be carried out in various ways, but the monitoring roughly falls under two categories: polling-based monitoring and notification-based monitoring.

With polling methods, the central monitoring server sends queries periodically to all nodes in the protected network, in order to get the list of all configured interfaces from each node. The collected lists are then checked for the presence of unwanted interfaces. With notification-based methods, a node sends a notification message about a newly configured interface to the central monitoring server.

Note that the above methods make several assumptions about the nodes in protected networks:

- each node must run an agent that responds to server queries or sends notifications,
- the nodes must run a secure configuration where the user can not interfere with the agent (e.g., deinstall or alter the agent)

The main disadvantages of the polling methods over the notification based approaches is larger network bandwidth consumption and the presence of a detection lag – periodical polling might require a considerable amount of network bandwidth, and must be carried out frequently enough. Despite this, there could be many polling rounds that yield no results about illegal interfaces, thus wasting the bandwidth; also, if an illegal interface is created between two polls, it is detected only after the second poll. With the notification based approach, a network bandwidth is only used when new interfaces are created. However, this approach has the following drawback – if a transmission error occurs when sending the notification (e.g., due to a temporary network disruption), the central monitoring server will not get the information about the new interface. Therefore, it often makes sense to use polling and notification based monitoring in parallel (this allows for larger intervals between subsequent polls, thus saving the network bandwidth) – if a notification fails to arrive to the monitoring server, the new interface will be detected later by the regular poll.

Note that the monitoring described above can be implemented with prominent network management frameworks like HP OpenView [4], Tivoli [5], and OpenNMS [6] (the first two are commercial products, while the last one is an open-source solution). These network management frameworks employ SNMP (Simple Network Management Protocol) for monitoring network nodes, and if a node is running an SNMP agent, the network management framework is able to collect status information from

the node. Among other commonly collected things, network management frameworks fetch interface information from each node (this is stored to a central database and is used for determining the network topology). The status is collected from nodes typically in fixed time intervals (e.g., once in 12 hours). Apart from periodical status collection, SNMP agents running at nodes can be configured to send SNMP notifications to central network management server, including the SNMP linkUp notification (this notification is generated when an interface becomes active on a node). By processing the status information and SNMP notifications from nodes, illegal interface creation can be detected with network management frameworks. If custom scripts/programs are needed for active interface detection, there are several solutions available for this. The Net-SNMP package [7] is a widely used open-source suite of SNMP tools which includes an SNMP agent. The SNMP agent works on a wide range of platforms (majority of UNIX flavors and Windows), and is also highly extensible – the user can easily augment the agent with external programs in various ways. Apart from these features, the agent has an automonitoring capability – it can be configured to monitor its own SNMP variables and send an SNMP trap to a central network management server if a variable has a certain value. If the use of an SNMP agent is not desired and the local platform is UNIX, another solution for monitoring local network interfaces is Monitord. Monitord [8] is a lightweight and extensible monitoring agent for UNIX systems which has been developed by one of the authors of this paper. Unlike SNMP agent, monitord does not open a network port on a local node, but rather executes custom monitoring scripts after preconfigured time intervals. If a script detects an error or other abnormal condition, it executes a predefined command (e.g., sends an SNMP trap or produces a syslog message which can be received at the central network management server).

3.2. Monitoring the traffic to/from illegal gateways

Once an illegal gateway has become operational and has started forwarding traffic between protected and outside network, it creates a footprint in overall network traffic that can be detected by careful analysis. For example, if an illegal gateway works at the IP level (i.e., it is a Layer 3 gateway), then IP addresses of outside networks can be observed in the headers of IP packets which are going to or coming from the illegal gateway. An open issue is the use of public IP addresses in the protected network – this might complicate the detection of illegal gateways, since legitimate nodes are reported using IP addresses from outer networks. Fortunately, if the analysis includes interface IDs that the packet has used, the problem can be addressed. For example, if a network switch receives a packet from the interface that is used by a regular workstation and the source IP address in the packet does not belong to the workstation, it is an indication that the workstation is acting as an IP level gateway for other nodes.

If the gateway works at application layer (or works at Layer 3 with both source and destination NAT enabled), the detection of illegal traffic is somewhat more complex, since IP addresses of unexpected networks don't show up in IP packet headers anymore. However, communication with illegal gateway can still be discovered by applying behavioral analysis. In that case, we can observe packets from unexpected nodes going to a node that is not supposed to provide any services in the protected network.

In order to carry out network traffic monitoring, several well-known protocols (e.g., NetFlow and sFlow) have been widely used during the last decade. With these protocols, routers, switches and other network devices are configured to send information about forwarded traffic (or other traffic that has been seen) to network management server where it is stored and analyzed. Usually, the traffic information is arranged into tuples, where each tuple describes a transmission of network packet(s) from a sender to a receiver. The sender and receiver are identified by IP addresses, communication

ports, and transport protocol ID (note that some transport protocols like ICMP don't use ports). In addition, input and output interface of the network device that the packet has traversed are also included in the traffic information. This data allows for identifying unexpected communications between nodes that should not occur normally.

For the reasons of efficiency, the traffic information is extracted from network packet headers only without doing any packet payload analysis. For increasing efficiency even further, a network device can be configured to use packet sampling – the extraction of traffic information from a certain fraction of packets only (e.g., only every 100^{th} or 1000^{th} packet is processed). Also, sometimes specialized network probes are used for extracting traffic information from network packets. However, in order to have an adequate monitoring system for detecting illegal gateways that provides a complete picture of network traffic, the following conditions must be met:

- traffic information must be collected from all network segments where illegal gateways might be installed,
- traffic information must be collected without packet sampling (i.e., for all packets which traverse network segments of interest).

There are several widely used open-source solutions for traffic information collection and analysis which deserve to be mentioned. Flow-tools [9] and SiLK [10] are widely used software packages for the collection and analysis of NetFlow data sent by network devices (SiLK also supports the IPFIX protocol). Fprobe [11] is a lightweight Netflow probe software which runs on a number of UNIX platforms, listens for all the network traffic on one of the local interfaces, and emits NetFlow data about this traffic to the specified collector.

Note that in some cases the collection and analysis of packet header data might not reveal the presence of an illegal gateway – sometimes nodes in the protected network frequently provide services to each other. For example, if an illegal application layer gateway gets installed in one of the nodes and uses a legal service port, it is hard to distinguish malicious traffic from legitimate packets. However, in that case a protocol analysis techniques might help. With these techniques, the network packet payloads are analyzed for establishing an application layer protocol of the traffic, and if unexpected application layer protocol is detected, an alarm is sent to the security staff. Although such packet analysis requires a lot of computing power, industrial solutions for accomplishing such tasks are available on the market, e.g., SourceFire RNA probes [12].

Finally, sometimes IP level gateways advertise their connectivity to outer networks with routing protocols (e.g., OSPF or RIP). Therefore, the monitoring of such traffic from unexpected sources can be helpful for illegal gateway detection. Also, unexpected changes in the routing tables of legal routers could be a symptom of illegal gateway activity in the network.

3.3. The scanning of protected networks

The methods for detecting illegal gateways which have been described in sections 3.1 and 3.2 assume that security analysts can collect and access various monitoring information from the nodes of the IT system (e.g., SNMP traps from workstations, or NetFlow records from routers and switches). Unfortunately, if the local site does not have any monitoring practices implemented, creating a system for collecting relevant security data from the whole IT system is a time-consuming and difficult task. In

such cases the scanning of protected networks for unexpected/abnormal/malicious configurations often provides a partial but quick solution to the problem of illegal gateway detection – instead of changing the configuration of hundreds or thousands of workstations and network devices, only a few dedicated nodes are deployed in the local network which run the network scanning software. However, it should be emphasized that this solution alone is far from being complete, since mere scanning will not reveal several types of illegal gateways (one such scenario is described below).

There are several different ways for scanning protected networks – some methods/tools aim at detecting illegal application layer gateways, while others attempt to find illegal IP level gateways. The scanning of protected networks for illegal open proxy ports is a method for discovering illegal application layer gateways. With this method, a security administrator periodically runs a scanning tool like Nmap [13] or Nessus [14] for all nodes in the protected network, in order to detect unexpected open ports on nodes which could potentially be used by illegal proxies. The main disadvantages of the method are high network bandwidth consumption and inability to detect an open port if the proxy has protected it with a firewall rule (e.g., a rule allows access to an illegal proxy for certain nodes only). The main advantage of the method is the ability to find illegal proxies when they are passive and have not transmitted any data to/from outside networks yet – in that case, all methods that find illegal gateways by monitoring their traffic obviously don't work.

In order to find illegal IP level gateways, several other scanning tools and techniques can be used. A simple method for finding illegal gateways is to have two nodes A and B for network scanning, where A resides in the protected network and B in the outer network (e.g., in the Internet). In order to detect whether node C in the protected network acts as an illegal gateway, node A sends an ICMP echo (ping) packet to C, but sets the source IP address of the packet to B. If node C responds to that packet with ICMP echo reply and the response reaches node B, node C must have an illegal connectivity to the outer network (this example assumes that ping traffic between protected and outer network is dropped by legitimate gateways, which is often the case). Instead of using ICMP echo packets, other packet types could be employed, like sending a TCP SYN packet with a forged source IP address from A to C (if the port is open, C sends SYN-ACK packet to B, otherwise RST packet could be sent).

One of the most widely used commercial tools for this purpose is IP Sonar [15]. IP Sonar employs methods similar to the approach described above for detecting illegal routers that connect the protected network with other networks (so called “network leak detection”) [15, 16]. Apart from finding unexpected routes to other networks, IP Sonar is also capable of network topology and host discovery, and device fingerprinting (detection of device OS, active services, etc.). Device fingerprinting is an important feature, since sometimes illegal gateways provide IP packet routing for a few nodes only from the local network, dropping all traffic from other nodes in the protected network. Therefore, scanning the network for illegal routes might not reveal such gateways. However, device fingerprinting could reveal the manufacturer, OS, hardware type, and other info for the illegal gateway, thus providing a valuable information for their detection. There are also several well-known open-source solutions for device fingeprinting like Nmap [13] and p0f [17]. Nmap supports active fingerprinting – in order to find the device type, the device is probed with specially crafted network packets. P0f implements passive fingerprinting – it observes normal traffic from/to the device, and analyses these data for detecting the device type. Therefore, p0f suits well for the cases where device fingerprinting must be done in a transparent and non-intrusive manner. BackTrack [18] is another valuable open-source toolkit that supports not only device fingerprinting, but contains a large set of network auditing, vulnerability assessment, and penetration testing tools. BackTrack is a Linux distribution which can be used for building a specialized appliance for finding various security issues in the local network, including devices of unexpected type.

4. Conclusion

This paper provided an overview of various techniques and tools for illegal gateway detection. Note that the methods we have outlined are not mutually exclusive, but rather complementary – in fact, for the reasons of completeness, several techniques we have described should be employed in parallel. For example, if an illegal HTTP proxy has been started which has not yet forwarded any traffic to the public internet, this illegal gateway can not be detected by analyzing network traffic; however, scanning the local node could reveal the open TCP port used by the proxy. As another example, if a workstation illegally connects to the Internet via WiFi, the immediate detection of such a connection is possible only if the local node is monitored with an agent. Therefore, if a complete solution is desired for illegal gateway detection, all three detection method classes (outlined in sections 3.1, 3.2, and 3.3) must be employed.

References

[1] http://www.faqs.org/patents/app/20090190602
[2] http://www.wipo.int/pctdb/en/wo.jsp?WO=2009043745
[3] http://www.google.com/patents/about?id=_tqDAAAAEBAJ
[4] http://www.openview.com
[5] http://www.tivoli.com
[6] http://www.opennms.com
[7] http://www.net-snmp.org
[8] http://ristov.users.sourceforge.net/monitord
[9] http://www.splintered.net/sw/flow-tools
[10] http://tools.netsa.cert.org/silk/index.html
[11] http://fprobe.sourceforge.net
[12] http://www.sourcefire.com
[13] http://nmap.org
[14] http://www.nessus.org
[15] http://www.lumeta.com/ipsonar
[16] http://www.infoworld.com/d/networking/lumeta-chief-scientist-checks-network-leaks-ip-sonar-507
[17] http://lcamtuf.coredump.cx/p0f.shtml
[18] http://www.backtrack-linux.org