ARP

Abstract
In this part, you will use your project parser to monitor network flows at the link layer and detect abnormalities in the request-response patterns of the address resolution protocol. While you have learned in chapter 4 how a malicious, spoofed ARP response can be identified, implementing such a policy in practice is actually quite challenging. In “Dynamic ARP Inspection”, you will experience these challenge for the comparatively simple ARP protocol and learn three lessons that are critical to building an effective IPS system: First, there exist differences between operating systems and how they implement a particular protocol that an IPS might take into consideration. Second, as hosts may join and leave the network at any time, each host will have a different view on the state of the network than your global monitoring system. Thus, a request which may look unusual could actually be benign given a host’s local state. And third, if you detect a violation of a policy, planning an effective response strategy requires careful planning, as interventions might cause different effects depending on the history and past interactions the network devices have had.
Finally, this assignment will also modularize the code base you have built in the first project, in order to have well-engineered, interaction-free code.
This project is due Sunday 24 Feburary, 23:59.

The Address Resolution Protocol (ARP) accomplishes the binding between the device addresses of the network interface cards (NIC) and the logical addresses of the hosts to which the NICs are connected. This helper protocol thus forms a bridge between the link layer and the network layer, and enables that network traffic can be delivered to the right host in the local area network. In chapter 4, we have seen that although networking hardware comes programmed by manufacturers with unique link layer addresses, it is trivial to change these device identifiers.

The ability to assume arbitrary link layer addresses and broadcast ARP responses for these spoofed addresses creates a number of security issues. Impersonating the link layer address of another host, a malicious host could inject packets on behalf of a victim. After injecting specially crafted ARP replies that change the bindings between victim IP and MAC address, an adversary can also launch a man-in-the-middle attack, thus intercepting all traffic between two IP addresses at the link layer. Finally, if an attacker is flooding the network with malicious ARP packets, the availability and proper functioning of hosts and network infrastructure may be compromised.

Within the context of link layer security, we have introduced a number of countermeasures that can also help address above risks about the address resolution protocol:

- Static ARP tables do not allow that a correct binding between IP and MAC address is overwritten by a malicious advertisement. Alternatively, hosts may be configured to ignore gratuitous ARP replies. While effective against an impersonification and MITM attack, statically configured ARP tables come at the expense of additional configuration overhead and somewhat limit the deployment of resilience strategies such as failover service setups.

- Port security functionality in switches can enforce that only specific MAC addresses or a maximum number of MAC addresses may be active on a specific port. While this addresses the risk of a denial-of-service attack on the network hardware, port security does not stop a malicious changing of IP to MAC address mapping.

- In dynamic ARP inspection, the networking hardware monitors the ARP traffic to compare it against a reference list of official IP-MAC bindings. Any deviations from the reference may be configured to trigger an alarm to the administrator or some other corrective action of the network. Dynamic ARP inspection is a standard feature in most layer 3-enabled switches, as well as present in network-based intrusion detection systems.

Abnormal ARP Traffic

In this project part, you will create a specification and design plan for dynamic ARP inspection (DAI), and develop a package that implements DAI for your intrusion prevention system. Before we can turn to any implementation, it is essential to first specify in detail the kind of ARP packets and ARP exchanges that can be deemed normal, and from this compile a list of abnormal behavior to monitor for.

Recall from the discussion in chapter 4 the normal progression of an ARP lookup. A host in need to find the link layer address that services a particular IP sends out an ARP request to the link layer broadcast address. The host associated with that IP address returns an ARP reply to the requesting host,

based on the link layer address in the request packet. Hosts may also proactively announce their MAC-to-IP binding by sending an unsolicited ARP reply to the network. Such gratuitous ARPs are commonly used in high availability setups, when a backup assumes the network-layer address as part of a failover, but also a common practice when a host joins a network to detect possible IP address conflicts.

As the handshake in the address resolution protocol is very generic, it is necessary to clarify which situations should be classified as abnormal. First of all, when looking at ARP from a protocol perspective, it is necessary to ensure that all ARP packets on the link are properly formatted, and that the included MAC addresses are valid. A host trying to bind an IP address to the link layer broadcast address is clearly not permitted. When monitoring for "illegal" advertisements, it is however necessary to have a baseline of which behavior is considered normal. In case of valid MAC-IP mappings, switches and IDSes contain a list of "normal" allocations, with any ARP packet violating these bindings raising an error message.

Although actually not part of the ARP specification in RFC826, some operating systems have adopted a number of additional requirements for ARP packets. For example, systems may assume that ARP requests have to be sent to the broadcast address, ARP responses reply to the unicast address of the sender and are not returned to the link layer broadcast address, or that ARP packets are internally consistent in that the MAC addresses of the link layer header match those included in the ARP packet. While according to the ARP protocol specification, packets with these properties are technically allowed, they are discarded by some ARP implementations. We thus want to consider these scenarios as part of our intrusion prevention system as a noteworthy event: if some hosts discard such packets as illegitimate, they are not included in the ARP caches of these particular network devices, thus leading to different perspectives on the local network among different devices. Instead of raising an error message, the DAI module of the IPS will generate a notification for such deviations instead of a hard error message.

> **Task 1** *Based on the description of normal and abnormal ARP traffic above, compile a list of scenarios. Each scenario is described by a specific request-response constellation and/or certain packet features or packet field values. Associate to each scenario the labels "permitted", "error", and "notice", that define the action of your IPS module.* ∎

Architectural Design of Intrusion Prevention Systems

Before we dive into a concrete implementation of ARP detection, we will first review some general principles for the design of intrusion detection and prevention systems. As you will add more functionality over the coming weeks, a well-structured design will help you maintain an efficient and side effect-free code base.

For the term project, we will adopt the design principles presented in the "Common Intrusion Detection Framework"

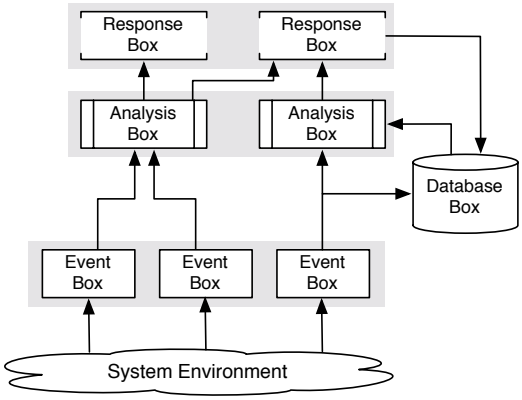

Figure 1. The Common Intrusion Detection Framework recommends to separate data collection, analysis and response into separate connected building blocks which can turn to a database for historical records.

(CIDF) (Staniford 1998), which functionally decomposes the activities of an intrusion detection and prevention system into four building blocks: first, an *event box* which collects data from the environment it monitors, second, an *analysis box* which processes events in search for abnormal behavior and triggers an alarm upon detection, third, a *response box* which initiates a response in case of an alarm, and fourth, a *database box* which stores historical data and may be queried for information.

Modules exchange information based on a common format, in the CIDF specification called a *gido* or generalized intrusion detection object. This allows modules to be freely interconnected to build more complex analysis. A box might consume two different event streams, or push out messages and reports to multiple other modules. Analysis and response boxes can consult historical records from the database, but also update stored data to relate information. Depending on the type and link between boxes, data is either pushed automatically forward (for example from the event to the analysis box) or provided upon request (for example a read from the database). While the flexible interconnection of individual modules allows you to keep each component simple and self contained, their combination enables you to build up some powerful functionality.

Aside from a general IDS/IPS architecture, the CIDF module also specifies a message interchange format. As the term project is meant for you to learn the main concepts and challenges in IPS design, we will only adopt the architectural best practices but not implement the more complicated mechanisms for universal interoperability in our proof-of-concept.

A Dynamic ARP Inspection Module

In this assignment you will use the foundation of the libpcap binding and traffic sniffer you have developed in the first part of the term project, and extend the foundation with a self-contained module that implements the necessary packet

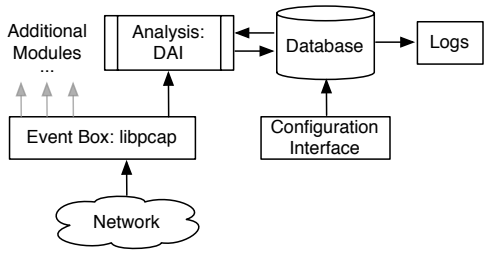

Figure 2. Your dynamic ARP inspection algorithm will run as one of many analysis components in your IPS prototype.

parsing and logic for dynamic ARP inspection. Your DAI module follows the general architecture for IPS designs as outlined above and hooks as an analysis box into the foundation via an interface which we will now call an event box. The pcap event box forwards a copy of any packet that your libpcap connection has received from the operating system to every connected module. Based on the assignment above, you will have noticed that in order to perform proper anomaly detection on ARP, your analysis module will need access to some past information. Create a simple "database" where your DAI module can store and retrieve information from, in the most basic form a simple data structure in memory with the appropriate interfaces to access it.

Design a software interface that allows you to connect event, analysis, database and in the future also reponse boxes. It should be general enough so that additional functionality can be seamlessly added to the intrusion protection system in the future via the same mechanism. In addition, it should be possible to provide modules with some configuration values, which could be read in at program start from a file and stored in the database. To report intrusions and help you in the development, also add some possibility to send error and notification messages to a general logging mechanism. This could be done via the database, or a separate message bus.

So far we have assumed that the dynamic ARP inspection mechanism has an internal list of valid IP-to-MAC allocations, without going into detail where such a list could be obtained from. In practice, DAI uses two sources as a reference: first, monitoring devices draw on an ARP Access Control List (ACL), a curated list that is loaded into the switches or IPS. Second, devices independently derive a mapping of legitimate bindings from the network traffic itself. The most common technique is *DHCP snooping*, where a switch or IPS records the IP address assignment a host has received from a DHCP server and enforces that ARP packets match this prior assignment.

Within the scope of this project part, we will ignore DHCP snooping. Instead, your DAI module will read in an external configuration file that lists valid IP and MAC address allocations. Aside from one-to-one mappings, your configuration format and module should also allow that IP addresses can be assumed by more than one MAC address, for example to enable high-availability setups.

Task 2 *Design and document your configuration format. Implement your DAI module, which will log ARP packets that have generated an error or notice via your IPS's API. To demonstrate your module's functionality, create test suite of ARP packets (for example using scapy) that you can replay as a pcap file to your IPS.* ∎

Aside from detecting abnormal traffic, some IPSes also have the ability to detect unusual deviations in host behavior that could hint at a potential implementation mistake. To visualize this concept, you will implement a similar functionality in your DAI module and in addition to detecting abnormal ARP traffic, check whether all hosts follow the reference in RFC826. Make a plan how you can determine from network traffic whether a host is discarding certain ARP replies.

Task 3 *Extend your DAI module to detect host-specific deviations, and test your network for deviations.* ∎