Intrusion Detection and Malware Analysis

Signature-based IDS

Pavel Laskov
Wilhelm Schickard Institute for Computer Science
Misuse detection systems

- **Expert systems** (NIDES, Emerald): rule-based decisions, rather slow, binary decisions only.
- **Signature matching** (Snort, Bro, Cisco Secure IDS, ISS RealSecure): pattern matching, policy scripting.
- **State transitions** (STAT suite): description of attacks by assertions over a state transition diagram.
- **Data mining**: automatic extraction of attack description rules from labeled examples.
- **Machine learning**: automatic extraction of “black-box” classifiers from labeled examples.
Snort highlights

- Initial open source release (December 1998):
  - plain sniffer, no rules
- 1.0 release (June 1999)
  - basic rules language, stateless packet processing
- 1.5 release (December 1999)
  - packet pipeline architecture used up-to-date
- 1.8 release developed by Sourcefire (mid-2001):
  - IP fragmentation and TCP reassembly
- Current release (2.8):
  - highly stateful, 3000+ rules, protocol anomaly detection
Snort design criteria

- A **lightweight** intrusion detection tool
  - cross-platform portability
  - small footprint
  - easy installation and configuration
- A **simple language** for rules
- **High efficiency** and low memory and CPU consumption
  - packet-level detection (with no support for TCP stream reassembly in early versions)
  - packet filtering using BPF and rule hierarchy
- An **open source** alternative to expensive commercial IDS
2. Then it sends them through a chute to determine if they are coins and how they should roll (the preprocessor).

3. Next, it sorts the coins according to the coin type. This is for storage of quarters, nickels, dimes, and pennies (on the IDS this is the detection engine).

4. Finally, it is the administrator’s task to decide what to do with the coins—usually you’ll roll them and store them (logging and database storage).

Packet sniffer interacts directly with a network card using libpcap.

Preprocessing, detection and alert components are implemented as plugins.

Various front-ends are available for logging (DB, Prelude meta-IDS, GUI).
Snort sniffer

- Operates in promiscuous mode: passes all traffic to OS.
- Performs basic packet filtering using BPF.
- Decodes packet headers using pointer casts.

```c
typedef struct _EtherHdr {
    u_int8_t ether_dst[6];
    u_int8_t ether_src[6];
    u_int16_t ether_type;
} EtherHdr;

/* lay the ethernet structure over the packet data */
p->eh = (EtherHdr *) pkt;
```
Plugin architecture enables dynamic plugin configuration.

Preprocessor functions:
- Stream reassembly (stream4)
- Packet defragmentation (frag2)
- Protocol decoding/normalization (HTTP, RPC, telnet)
- Alternative (non-rule) detection modes (portscan, arpspoof)
Snort detection engine

- Rules are parsed into internal data structure.
- Rule matching is prioritized according to matching complexity:
  - IP header rules
  - TCP header rules
  - Application protocol header rules
  - Content rules
- Multiple matches are possible: the highest priority alert is reported.
- **General format:** header (options)
- **Header:**
  - fixed format
  - present in every rule
- **Options:**
  - variable format
  - not always necessary
- **Example:**

  ```plaintext
  alert tcp $BAD any -> $GOOD any
  (flags: SF; msg "SYN-FIN scan");
  ```
- **General format:** `action proto srcaddr srcport dir dst addr dstport`
- **Example:** `alert tcp $BAD any -> $GOOD any`
- **Actions**
  - alert, pass, log
- **Protocols**
  - tcp, udp, icmp, ip
- **Directions**
  - `->` (unidirectional), `<>` (bidirectional)
- **Variables, wildcards and expressions can be used, e.g.**
  - `!$HOME, any, etc.`
**Snort rules: options**

- **General format:** (keyword: value;)
- **Example:** (flags: SF; msg “SYN-FIN scan”;
- **Basic options:**
  - content: pattern matcher
  - pcre: Perl-compatible regular expression
  - msg: alert message
  - flow: test for TCP connection state, traffic direction
- **Content options can be combined using the conjunction.**
Snort rules: advanced options

- **Informational options**
  - sid (Snort ID), priority, ref (reference, usually to CVE’s)

- **Advanced payload options**
  - byte_jump, byte_test, distance, within, depth, offset, nocase, rawbytes, uricontent

- **Protocol analysis options**
  - Keywords for IP, TCP, ICMP protocols

- **Stateful rules**
  - flowbits, threshold, flow

- **Regular expressions**
  - use with care: expensive!
alert tcp any any -> any any \
    (flow: established, to_server; \ 
    content: "foo"; msg: "detected foo");

alert tcp $EXTERNAL_NET any -> $SMTP_SERVERS 25 \ 
    (msg: "SMTP exchange mime DOS"; flow:to_server,established; \ 
    content: "charset = |22 22|"; nocase; reference:bugtraq,1869; \ 
    reference:cve,2000-1006; reference:nessus,10558; \ 
    reference:url,www.microsoft.com/technet/security/bulletin/MS00-082.m 
    classtype:attempted-dos; sid:658; rev:11;)

alert tcp $EXTERNAL_NET 80 -> $HOME_NET any \ 
    (msg: "EXPLOIT Netscape 4.7 client overflow"; flow:to_client,established; \ 
    content: "3|C9 B1 10|?|E9 06|Q<|FA|G3|C0|P|F7 D0|P"; \ 
    reference:arachnids,215; reference:bugtraq,822; \ 
    reference:cve,1999-1189; reference:cve,2000-1187; \ 
    classtype:attempted-user; sid:283; rev:10;)

Snort rule examples
Snort summary

- A de-facto standard IDS in the practical security community
  - More than 3,000,000 downloads
  - About 200,000 users
- A light-weight, easily configurable IDS
- Good performance and reliability
- Moderate expressivity of rule language
- Numerous appliances available
  - Database, logging and alert interfaces
  - GUI tools
  - Intrusion prevention and firewall interfaces
  - Shared object rules: more complex functionality