Security Workshop

Eyal Kuttner

Road Map

- HW 4 – infrastructure
  - Kernel
  - Proxy in userspace
- Data leak prevention (DLP)
  - C code structure
  - Implementation with regex
- OrientDB vulnerability
  - OrientDB role management (oRole)
  - Privilege escalation
  - Remote code execution
  - Limitations of Firewall in this vulnerability

HW4 – Kernel

- Each packet pass some basic tests (IPv4, not XMAS, etc).
- If Pass, I check if it suitable to some connection in con table:
  - If suitable – promote TCP FSM and ACCEPT.
  - If exist connection that reject it – DROP.
  - If no such connection exist and it's syn no ack – check rules. If valid according to rules – add new connection and ACCEPT.
- If packet accepted – check if it special packet (contain dedicated port).
  - If special packet – identify if came from proxy:
    - Came from proxy: forge source to client/server to keep the proxy transparent.
    - Came from client/server: forger destination to proxy.

HW 4 – Epoll Proxy

- Epoll – a Linux kernel system call for a scalable I/O event notification mechanism. Thus allowing handle multiple connections in parallel.
- Simple passive proxy:
  - Hold two sockets that can forward data between them.
    - C ←→ PS ←===→ PC ←→ S
  - Read from kernel the destination of transferred data (address of server).
  - Update kernel connections to add proxy as client port.
  - Can shutdown connection of both side.
- Inherited by proxies for dedicated protocols. Override forward data to read it and decide if it problematic or not.

Data Leak Prevention

- I created proxies for ports 25 & 80.
- SMTP (port 25): For every message I used python parser to get only the payload(s).
- HTTP (port 80): For every message I simply took the payload after all headers.
- Both cases where sent to dedicated function to conclude if it's a leak.

C Code Structure

- Because it's a leak, the files looks like a regular c code.
- C Code shared similar words with English (if, else, return, include, define).
- But, while English is divided to sentences and paragraphs. C code is divided by lines, which have a defining structure.
- #define and #include will be at the beginning of the line.
- Usually same for: variables, typedef, inline, struct, return, if, else, for, //, etc.
- '{' and ';' will be at the end of line, while '}' will be at the beginning and the end.
- So, I don't look just at defining keys, but look at them only where I expected to find them in C.

Implementation with regex

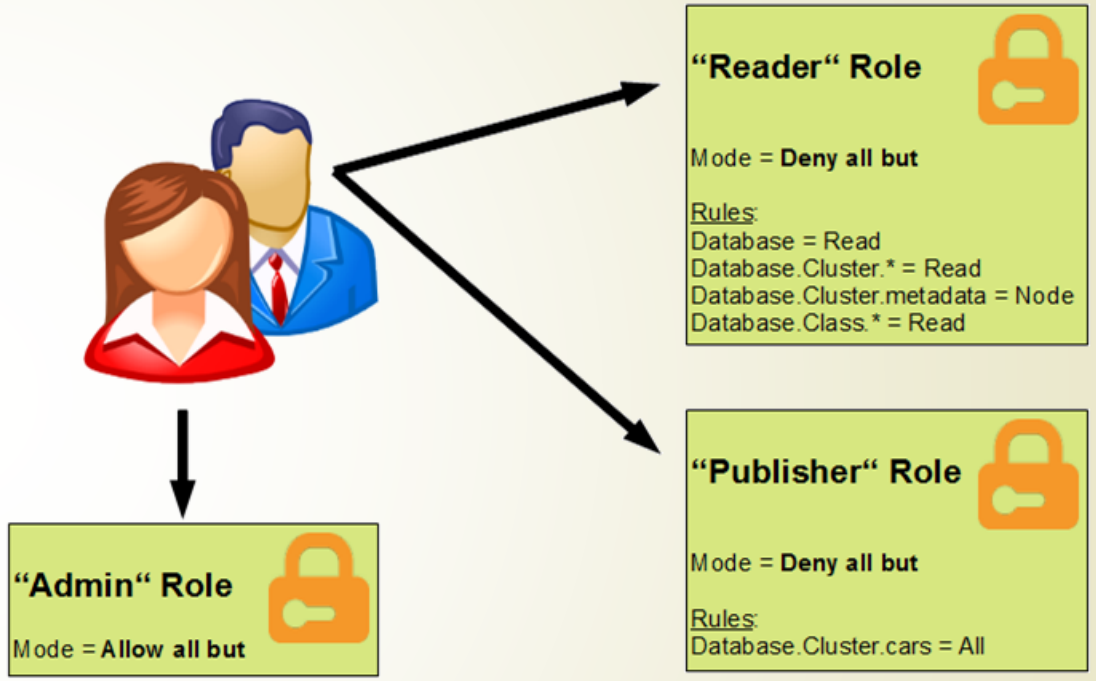
- To analyze the messages, I use python regex (short for regular expressions).
- I create a list of regex key, with attention to location in beginning (followed by optional spaces/tabs) or end of line.
- Although I use some keys (like malloc or printf) without location in line.
- Each key got a score for it.
- The total score = Sum(for each key: key's score * key's count in message),
- I divided that score in number of (non-empty) lines to get normalized score,
- If normalized score pass some threshold then it code C, else it just English.

OrientDB

- Open source NoSQL (Structured Query Language).
- Distributed Graph Database & Document Database.
- Written in Java
- Operate in port 2480.
- OrientDB 2.2.x Remote Code Execution

OrientDB Role Management (oRole)

- Role-based access control (RABC):
  - Each user has role.
  - Each role has different access.
- By Default, each database has 3 users:
  - admin (“admin”)
  - reader (“reader”)
  - writer (“writer”)
- oRole structure handles users and their roles and is only accessible by the *admin* user.
- **The Vulnerability**: In some cases, about oRole, this permission requirement is not required and information is returned to *unprivileged* users.



Privilege Escalation

- GRANT command: grant *permission* on *database* to *role*.
- Using GRANT command, ‘writer’ gets control about databases he shouldn’t have, including:
  - database.class.ouser: handle oRole management.
  - database.function: handle function execution.
- I developed dedicated proxy, which block any GRANT command that gives permission to ‘writer’ on any of these databases.

Remote Code Execution

- OrientDB can execute groovy functions, which don't use sandbox and expose system functionalities.
- No privilege escalation -> no remote code execution.
- Also, admin can run remote code and it's ok.

Limitations of Firewall in this vulnerability

- Other roles than ‘writer’:
  - Can do privilege escalation which leads to remote code execution.
  - We (obviously) don’t have access to oRole database, so we can’t identify when it’s legitimate and when it isn’t.
- Closing the stable door after the horse has bolted:
  - If the database already got privilege escalation, our defense would be pointless.
  - Remote code execution are sometimes legitimate, so we can’t just block them.