VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ISA PROJEKT

REVERSE-ENGINEERING NEZNÁMEHO PROTOKOLU

Jakub Hlava (xhlava52)

Analýza zachycené komunikace – popis neznámého protokolu

Informace o neznámém protokolu, pomocí kterého spolu komunikuje referenční klient a server (pro přehlednost jej budu označovat jako protokol ISA) jsem získal experimentováním s příkazy, které klient podporuje a zachytáváním komunikace.

Z experimentů je patrné, že protokol ISA využívá na transportní vrstvě protokol TCP a že výchozím portem pro komunikaci je port 32323.

Všechny požadavky i odpovědi v protokolu ISA jsou ohraničeny pomocí znaků 0x28 a 0x29, tedy kulatých závorek ( ). Tyto znaky se používají i uvnitř odpovědí pro oddělení logických celků, např. pro oddělení položek seznamu přijatých zpráv.

Požadavky i odpovědi v protokolu ISA mají obdobný formát. U požadavků je uvnitř závorek nejprve příkaz, ten je následovaný případnými argumenty, které jsou v případě řetězců navíc ohraničeny uvozovkami. U odpovědí serveru je uvnitř závorek nejprve informace o úspěchu či chybě, následovaná případnými daty, o které si klient příkazem požádal, příp. stručným zdůvodněním nastalé chyby.

Protokol ISA podporuje jednoduchou autentizaci – registraci a přihlašování uživatelů, proto většina příkazů (mimo příkazy pro registraci a přihlášení) má jako první argument přihlašovací token, který se získává ze serveru při přihlášení (detaily viz popis příkazů níže).

Platné příkazy protokolu ISA

`register`

Formát:	**`register "jmeno" "kódované heslo"`**
Popis:	*Zaregistruje nového uživatele* `jmeno` – unikátní jméno uživatele kódované heslo – uživatelovo heslo, které klient před odeslání na server kóduje pomocí base64
Odpovědi:	Registrace proběhla v pořádku: `(ok "registered user jmeno")` Chyba při registraci: `(err "user already registered")`

`login`

Formát:	**`login "jméno" "kódované heslo"`**
Popis:	*Přihlásí uživatele* `jmeno` – unikátní jméno uživatele `kódované heslo` – uživatelovo heslo, které klient před odeslání na server kóduje pomocí base64
Odpovědi:	Přihlášení v pořádku: `(ok "user logged in" "token")` Chyba při přihlašování: `(err "unknown user")`

list

Formát:	**`list "token"`**
Popis:	*Požádá server o seznam zpráv, které byly uživateli zaslány.* `token` – token získaný jako odpověď na příkaz login
Odpovědi:	Žádné zprávy: `(ok ())` Zprávy ve schránce: `(ok (1 "odesilatel" "předmět"))` Více zpráv: `(ok (1 "odesilatel" "předmět") (2 "odesilatel" "předmět"))`

send

Formát:	**`send "token" "příjemce" "předmět" "text zprávy"`**
Popis:	*Odešle zprávu uživateli* `token` – token získaný jako odpověď na příkaz login `příjemce` – jméno uživatele, kterému má klient odeslat zprávu `předmět` – předmět zprávy pro příjemce `text zprávy` – tělo zprávy pro příjemce
Odpovědi:	Odeslání proběhlo: `(ok "message sent")` Chyba při odesílání např. neznámý uživatel: `(err "unknown recipient")`

fetch

Formát:	**`fetch "token" ID`**
Popis:	*Načte celou zprávu ze serveru* `token` – token získaný jako odpověď na příkaz login `ID` – číslo zprávy podle příkazu list
Odpovědi:	Pro platné ID: `(ok ("odesilatel" "předmět" "tělo zprávy"))`

logout

Formát:	**`logout "token"`**
Popis:	*Odhlasí uživatele* `token` – token získaný jako odpověď na příkaz login
Odpovědi:	Vždy: `(ok "logged out")`

Návrh disektoru pro protokol ISA

Disektor pro protokol ISA je implementovaný v jazyce Lua. Pro identifikaci paketů protokolu využívá výchozí port referenčního klienta/serveru (32323), tzn, že v případě, že zachycená komunikace proběhla na jiném portu, pak není bez úprav disektor schopen protokol rozpoznat.

Pro rozpoznání typu zprávy slouží funkce `get_message`, ta porovnává úvodní úseky dat s příkazy uvedenými výše a s odpověďmi ok a `err`. Pro usnadnění rozboru protokolu slouží funkce `get_params` a `get_data`. Funkce `get_params` pomocí regulárního výrazu vybírá z dat paketu řetězce, obvykle parametry příkazů – text ohraničený závorkami a mezerami. Funkce `get_data` slouží opět za pomoci regulárních výrazů k rozložení odpovědi na příkaz list na jednotlivé zprávy, které lze poté ve Wiresharku snadno zobrazit.

Za pomoci těchto funkcí je v hlavní funkci dissectoru nejprve detekován typ zprávy pomocí `get_message`, poté podle typu zprávy jsou zpracovány odpovídající počty a typy parametrů příkazů, případně dále vyzískána všechna zajímavá data, která lze z dat získat. Tato jsou poté prezentována přidáním přehledných podstromů do popisu protokolu paketu v programu WireShark.

Limitace klienta

Ačkoliv referenční klient podporuje i IPv6, tato implementace podporuje pouze IPv4.
Další limitace oproti referenční implementaci nejsou známy.