Test GSE-LLR-102

" Lógica de Handshake (Falha)
Em caso de timeout, pacote TFTP inesperado (opcode/bloco) ou chave divergente, a rotina deve registrar AUTH-ERRO e retornar False."

Procedimento realizado

Metodologia de Análise estática de código
Análise no arquivo `gse/backend/protocols/tftp_client.py`

Resultados Obtidos

```
# ------------------------------------------------------------------------

def perform_authentication(self, gse_key: bytes, expected_bc_key: bytes) -> bool:
    """
    Realiza handshake de autenticação com o BC (TFTP 4-etapas).
    Implementa: GSE-LLR-098 a GSE-LLR-103
    """
    self.log("[AUTH] Iniciando handshake de autenticação (DATA/ACK)...")
    if not self.sock:
        self.log("[AUTH-ERRO] Socket não está conectado.")
        return False

    original_timeout = None
    try:
        # REQ: GSE-LLR-099
        original_timeout = self.sock.gettimeout()
        self.sock.settimeout(5.0)  # Timeout curto para handshake

        self.server_tid = None  # Reseta TID

        # --- PASSO 1: Enviar chave GSE para o BC ---
        # REQ: GSE-LLR-100 (Parte 1)
        self.log(f"[AUTH] Enviando chave GSE (DATA 1) para porta {TFTP_PORT}...")
        pkt = struct.pack("!HH", TFTP_OPCODE.DATA.value, 1) + gse_key
        self.sock.sendto(pkt, (self.server_ip, TFTP_PORT))
        self.log("[✓] DATA(1) com chave GSE enviado")
```

```
        # --- PASSO 2: Aguardar ACK(1) do BC ---
        # REQ: GSE-LLR-100 (Parte 2)
        self.log("[AUTH] Aguardando ACK(1) do BC...")
        # Usamos a nova helper function (GSE-LLR-133)
        ack_block = self.recv_ack_packet()
        if ack_block != 1:
            self.log(f"[X] ACK(1) não recebido, recebido Bloco={ack_block}")
            return False  # REQ: GSE-LLR-102

        self.log("[✓] BC aceitou nossa chave - ACK(1) recebido")

        # --- PASSO 3: Aguardar chave do BC (DATA 1) ---
        # REQ: GSE-LLR-101 (Parte 1)
        self.log("[AUTH] Aguardando chave do BC (DATA 1)...")
        # Usamos a nova helper function (GSE-LLR-137)
        result = self.recv_data_packet()
        if not result:
            self.log("[X] Chave do BC não recebida (timeout ou erro)")
            return False  # REQ: GSE-LLR-102

        block, bc_key = result
        if bc_key != expected_bc_key:
            self.log("[X] Chave do BC inválida")
            self.log(
                f"[AUTH]   Recebido: {bc_key.hex() if isinstance(bc_key, bytes) else bc_key}"
            )
            self.log(f"[AUTH]   Esperado: {expected_bc_key.hex()}")
            return False  # REQ: GSE-LLR-102

        self.log("[✓] Chave do BC válida")
```

```
            self.log("[✓] Chave do BC válida")

            # --- PASSO 4: Envia ACK confirmando a chave ---
            # REQ: GSE-LLR-101 (Parte 2)
            # Usamos a nova helper function (GSE-LLR-141)
            if not self.send_ack(block):
                self.log("[X] Erro ao enviar ACK para chave do BC")
                return False  # REQ: GSE-LLR-102

            self.authenticated = True
            self.log("[✓] Handshake de autenticação concluído com sucesso!\n")
            return True

        except Exception as e:
            # REQ: GSE-LLR-102 (parcial)
            self.log(f"[X] Erro durante autenticação: {e}")
            return False
        finally:
            # REQ: GSE-LLR-103
            if original_timeout is not None:
                self.sock.settimeout(original_timeout)
                self.log(
                    f"[AUTH] Timeout do socket restaurado para {original_timeout}s."
                )
```

O requisito está rastreado e implementado através dos passos 3 e 4 da função perform_authentication no arquivo destacado no procedimento realizado.