jSONAR
Simplifying Security

SONARG

Options for Ingesting Data into SonarG

SonarG is a powerful platform for capturing, managing and analyzing TB's of data using its fully integrated high performance analytics engines and highly efficient storage. The underlying data store within SonarG leverages a JSON-native architecture in order to easily and rapidly ingest data from a wide variety of sources and exploit the tremendous flexibility of NoSQL "schema on read" to create an unlimited variety of collections. This capability is especially valuable in the context of DB activity monitoring and security portals, as customers would like to blend data from sources including DAM, VA, classifiers, CMDBs and many other sources in order to stitch together disparate pieces of data into a cohesive, 360° view of the security and compliance profile for all databases.

In order to feed the SonarG Big Data analytics warehouse, there is a critical need for easily accessible and automated mechanisms for ingesting data into the SonarG platform and making it available for downstream reporting, analytics, etc. This paper outlines the various methods and processes associated with moving data into SonarG.

TECH NOTE #7201

Steps Covered in this Technote

LIST OF FIGURES

1. SonarG Ingestion Options

1.1: BUILT-IN ETL LAYER

SonarG provides a fully integrated ETL layer (Figure 1) that is readily accessible for enabling external data sources to "push" data into the SonarG system. This is the primary mechanism used to facilitate the transfer of data mart extracts from the Guardium collectors into SonarG, but can also be used for other external systems to automatically push data into SonarG.

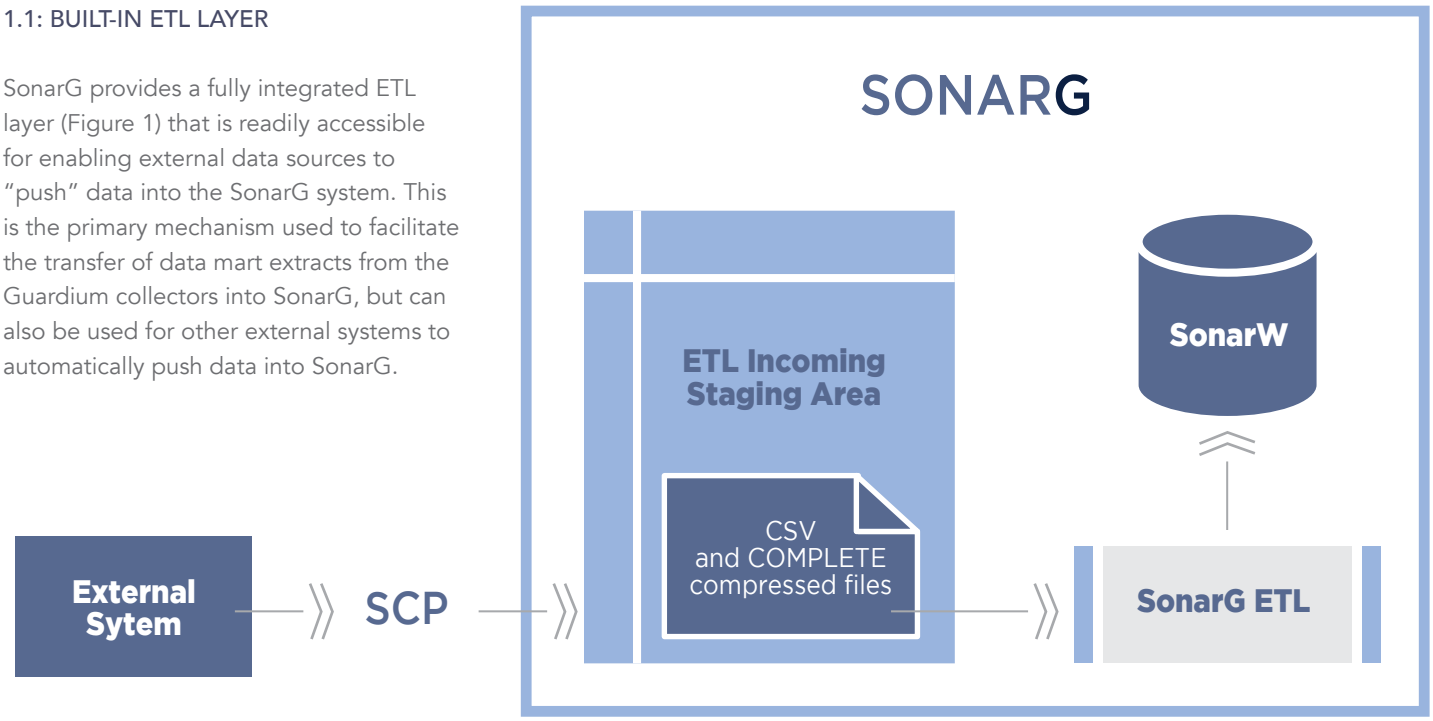


Figure 1 - SonarG Primary ETL Mechanism

The process flow for using this mechanism is outlined below in an example scenario that is modeled on linking a subset of Guardium activity data with privileged account information being managed within the CyberArk Privileged Account Management tool.

Note that there is no preparation needed on the SonarG NoSQL system to accept this incoming data. Any and all data is ingested and indexed automatically into SonarG. Separate data collections can be used for each system's incoming data stream or all ingested data can be consolidated into the same collection. This is completely up to the user and various implementation considerations.

1. The CyberArkExtract1.csv file shown in Figure 2 below represents a sample of credential information exported from the CyberArk tool.

CyberArkExtract1.csv

```
_id,Safe,Device type,Platform ID,Target system address,Target system user name,Group name,Last accessed date,Last accessed by,Last
modified date,Last modified by,Change failure,Verification failure,Failure reason
1,ABC,,,10.205.118.22,PSMConnect,,,,10/7/08 15:52,Aaaaaaaaaa,No,No,
2,ABC,,,10.205.118.22,PSMsecConnect,,,,10/7/08 15:52,Aaaaaaaaaa,No,No,
3,ABC,,,10.205.118.23,PSMsecConnect,,,,10/8/08 12:05,Aaaaaaaaaa,No,No,
4,ABC,,,10.205.118.24,PSMConnect,,,,10/8/08 12:05,Aaaaaaaaaa,No,No,
5,DEF,Operating System,WinDomain,10.205.118.25,AT887401@business.net,,9/1/11 13:22,secistrator,10/10/08 10:49,bbbbbb1,No,No,
6,PMWBTicketingSystem,Database,MSSql_remedy,192.168.1.137,CALE,,4/19/11 9:44,ggggggg,3/12/09 14:29,Aaaaaaaaaa,No,No,
7,PMWBTicketingSystem,Database,MSSql_remedy,192.168.1.137,PINK,,3/13/12 13:13,usssssss1,5/2/09 8:51,Aaaaaaaaaa,No,No,
8,UDB,Database,UDB_DB2UnixSSH,9.70.147.209,DB2ADMIN,,,,6/11/09 10:22,cccccc1,Yes,No,First login - Failed to verify user after
switching. code: 8008
9,UDB,Database,UDB_DB2UnixSSH,9.70.147.208,SCOTT,,,,10/12/09 21:04,PasswordManager,No,No,
10,SecurityEngineering,Security Appliance,Firewall1,10.218.109.67,id0000001,,,,11/18/10 13:53,ncash,No,No,
11,SecurityEngineering,Application,PointSec,,adm1,,,,3/19/11 16:10,ddddd,No,No,
12,SecurityEngineering,Application,PointSec,,adm2,,,,3/19/11 16:12,ddddd,No,No,
13,SecurityEngineering,Security Appliance,Guardium,All Prod 8.2,sec,,11/4/11 12:49,ihhhhhh,3/26/11 9:30,ddddd,No,No,
```

Figure 2 - Sample CyberArk data in CSV

2. To use the SonarG ETL layer, files need to be delivered to a specific SonarG directory and need to follow specific naming conventions: a) all exported CSV files must begin with EXP_ and b) include a timeframe specifier.

   $ mv CyberArkExtract1.csv EXP_CyberArkExtract1_20160610000000.csv

   CSV files are pushed as a compressed tarball and adhere to a file naming convention that defines the name of the data, the timeframe and the source of the data. For this example the data comes from a system called CyberArk with an identifier of 12345, and the files are packaged as below:

   $ tar cvzf 12345_CyberArk_EXP_CyberArkExtract1_20160610000000.gz
   EXP_CyberArkExtract1_20160610000000.csv
   EXP_CyberArkExtract1_20160610000000.csv

   Each data file to be delivered to SonarG must be accompanied by an empty "COMPLETE " file, as this is the mechanism used to inform the SonarG ETL system that all data has been properly and completely transferred via SCP from the external source to SonarG and the ETL job can begin. For this example, the pair of files delivered to SonarG for ingestion would be:

   -rw-rw-r-- 1 ubuntu ubuntu 38093618 Jun 12 13:09
   12345_CyberArk_EXP_CyberArkExtract1_20160610000000.gz

   12345_CyberArk_EXP_CyberArkExtract1_20160610000000_COMPLETE.gz
   -rw-rw-r-- 1 ubuntu ubuntu 0 Jun 12 13:14

   All files to be ingested by SonarG must be delivered to the **var/lib/sonargd/incoming** directory. Shown below are the contents of this directory and notice that the CyberArk files are co-mingled with various extract files coming from Guardium collectors.

   ubuntu@ip-172-30-0-114:~$ ls /var/lib/sonargd/incoming
   12345_CyberArk_EXP_CyberArkExtract1_20160610000000_COMPLETE.gz
   12345_CyberArk_EXP_CyberArkExtract1_20160610000000.gz
   1454978685_gmac01_EXP_GROUP_MEMBERS_20160519040000.gz
   1455013343_gmac03_EXP_BUFF_USAGE_20160302010000.gz
   1455013343_gmac03_EXP_EXCEPTION_LOG_20160302010000.gz
   1455013343_gmac03_EXP_POLICY_VIOLATIONS_20160302010000.gz
   1455029471_gmac02_EXP_ACCESS_LOG_20160302190000.gz
   426441011_china2_EXP_BUFF_USAGE_20160612120000_COMPLETE.gz
   426441011_china2_EXP_BUFF_USAGE_20160612120000.gz
   426441011_china2_EXP_EXCEPTION_LOG_20160201130000.gz
   90373858_ghp02_EXP_GROUP_MEMBERS_20160229090000.gz

3. Following the delivery of the extract file and its corresponding COMPLETE file, the SonarG ETL process will automatically ingest and map the data into a JSON collection and the completed process will be confirmed in the SonarG ingestion log. The CyberArk data is now within SonarG and easily accessible for queries, joins, reporting, etc., as shown in Figure 3 below.

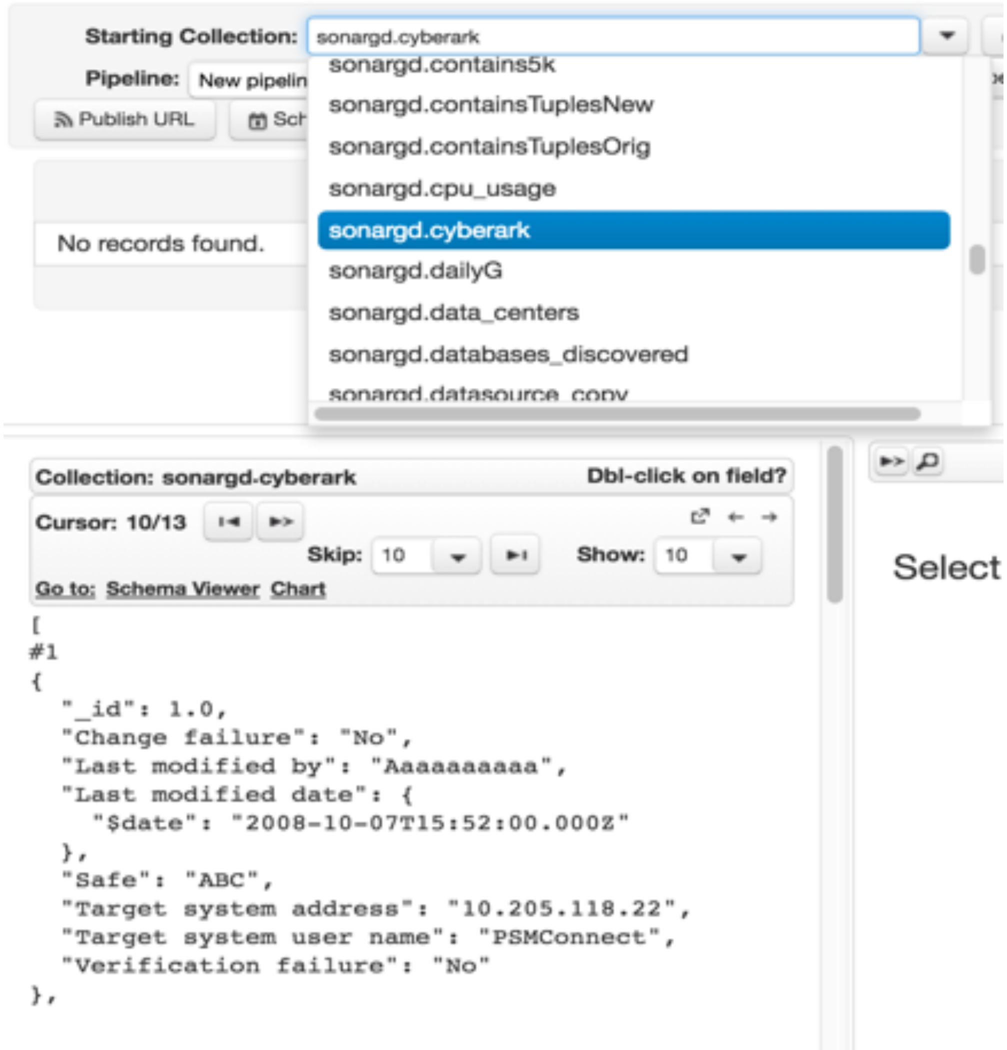


Figure 3- CyberArk Data in SonarG

4. Using a join condition enables CyberArk data to be easily linked to Guardium activity data as shown in Figure 4 below, which depicts the Guardium activity associated with the privileged IDs generated by CyberArk.

Session Start	Session End	Server IP	Platform_ID	DB User Name	Last_modified_by	Safe	Change_failure	Service Name	Source Program	Analyzed Client IP
2016-06-12 00:04:52	2016-06-12 00:04:52	9.70.147.209	UDB_DB2UnixSSH	DB2ADMIN	cccccc1	UDB	Yes	DB2	DB2HMON	9.70.147.209
2016-06-12 00:04:52	2016-06-12 00:04:52	9.70.147.209	UDB_DB2UnixSSH	DB2ADMIN	cccccc1	UDB	Yes	DB2	DB2HMON	9.70.147.209
2016-06-12 00:04:52	2016-06-12 00:04:52	9.70.147.209	UDB_DB2UnixSSH	DB2ADMIN	cccccc1	UDB	Yes	DB2	DB2HMON	9.70.147.209
2016-06-12 00:04:52	2016-06-12 00:04:52	9.70.147.209	UDB_DB2UnixSSH	DB2ADMIN	cccccc1	UDB	Yes	DB2	DB2FMP64.EXE	9.70.147.209
2016-06-12 00:04:53	2016-06-12 00:04:53	9.70.147.209	UDB_DB2UnixSSH	DB2ADMIN	cccccc1	UDB	Yes	DB2	DB2HMON	9.70.147.209
2016-06-12 00:05:52	2016-06-12 00:05:52	9.70.147.209	UDB_DB2UnixSSH	DB2ADMIN	cccccc1	UDB	Yes	DB2	DB2FMP64.EXE	9.70.147.209
2016-06-12 00:05:52	2016-06-12 00:05:52	9.70.147.209	UDB_DB2UnixSSH	DB2ADMIN	cccccc1	UDB	Yes	DB2	DB2HMON	9.70.147.209
2016-06-12 00:05:52	2016-06-12 00:05:52	9.70.147.209	UDB_DB2UnixSSH	DB2ADMIN	cccccc1	UDB	Yes	DB2	DB2HMON	9.70.147.209
2016-06-12 00:09:52	2016-06-12 00:09:52	9.70.147.209	UDB_DB2UnixSSH	DB2ADMIN	cccccc1	UDB	Yes	DB2	DB2HMON	9.70.147.209
2016-06-12 00:09:53	2016-06-12 00:09:53	9.70.147.209	UDB_DB2UnixSSH	DB2ADMIN	cccccc1	UDB	Yes	DB2	DB2HMON	9.70.147.209
2016-06-12 00:09:53	2016-06-12 00:09:53	9.70.147.209	UDB_DB2UnixSSH	DB2ADMIN	cccccc1	UDB	Yes	DB2	DB2HMON	9.70.147.209
2016-06-12 00:09:54	2016-06-12 00:09:54	9.70.147.209	UDB_DB2UnixSSH	DB2ADMIN	cccccc1	UDB	Yes	DB2	DB2HMON	9.70.147.209
2016-06-12 00:11:53	2016-06-12 00:11:53	9.70.147.209	UDB_DB2UnixSSH	DB2ADMIN	cccccc1	UDB	Yes	DB2	DB2HMON	9.70.147.209

Figure 4 – Merged CyberArk and Guardium Data

1.2: DIY ETL

A second ETL option for moving data into the SonarG system is to exploit the 100% compatibility between the SonarG data store and MongoDB, as shown in Figure 5. SonarG can accept connections using any MongoDB ETL and import tool and therefore you can easily ingest data via the mongoimport for CSV, TSV and JSON files. There are many tools and drivers available to facilitate this process and a list can be found at http://mongodb-tools.com/

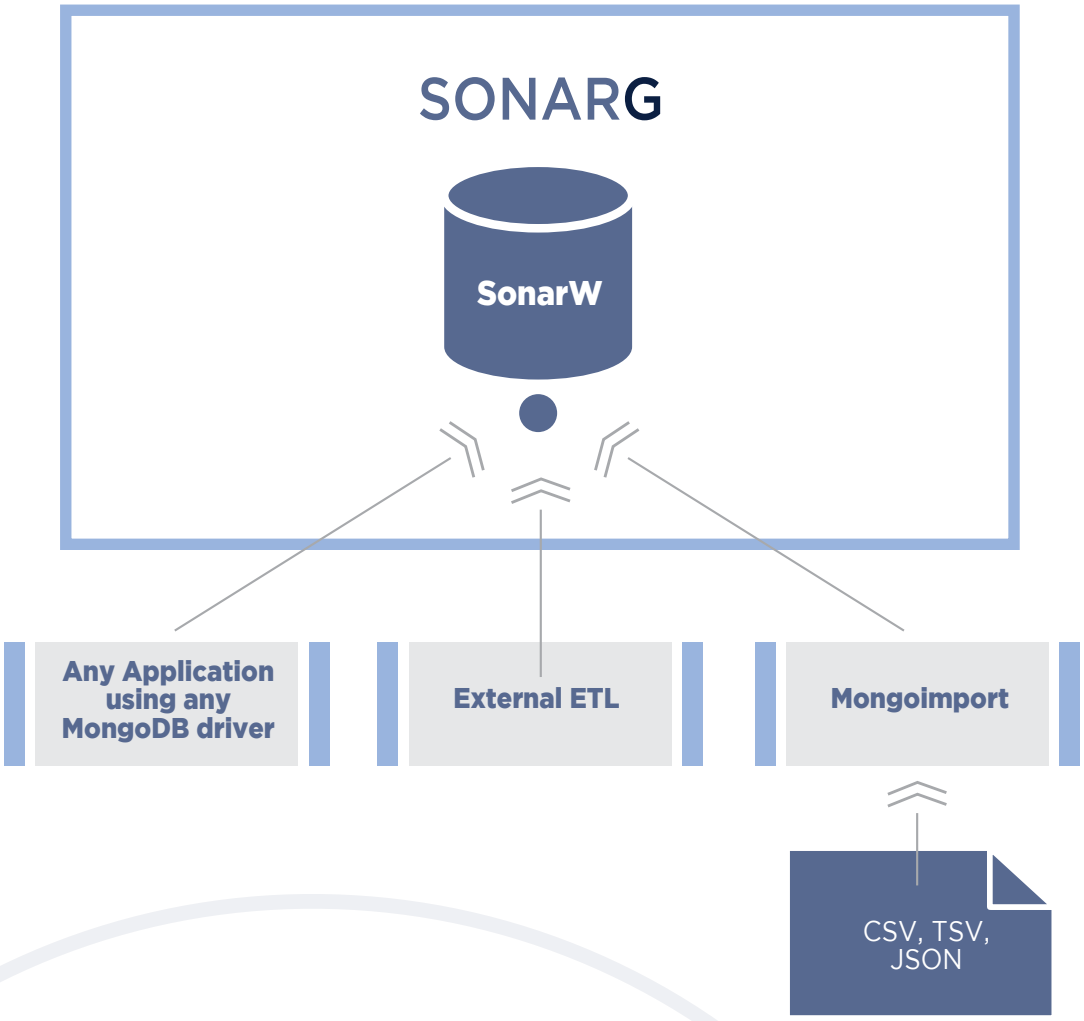


Figure 5 - DIY ETL Diagram

As an example if you have a CSV file called foo.csv and wish to ingest it into the sonargd database you can use:

mongoimport –h <the sonarg hostname> --port 27117 –u <your username> -p <your password> --authenticationDatabase admin –d sonargd –c <the name of the collection you want the records inserted to> --type csv –headerline foo.csv

1.3: Pull CSV Files

The third ETL option is to use SonarG's ability to schedule and dispatch jobs via the invocation of a web service, as in Figure 6. SonarG has a scheduler/dispatcher module that runs jobs according to a cron schedule. A job can be scheduled that periodically makes a call to an external system that returns CSV files that are then ingested into SonarG. This assumes that the external system offers an HTTP/HTTPS-based service capable of generating CSVs.

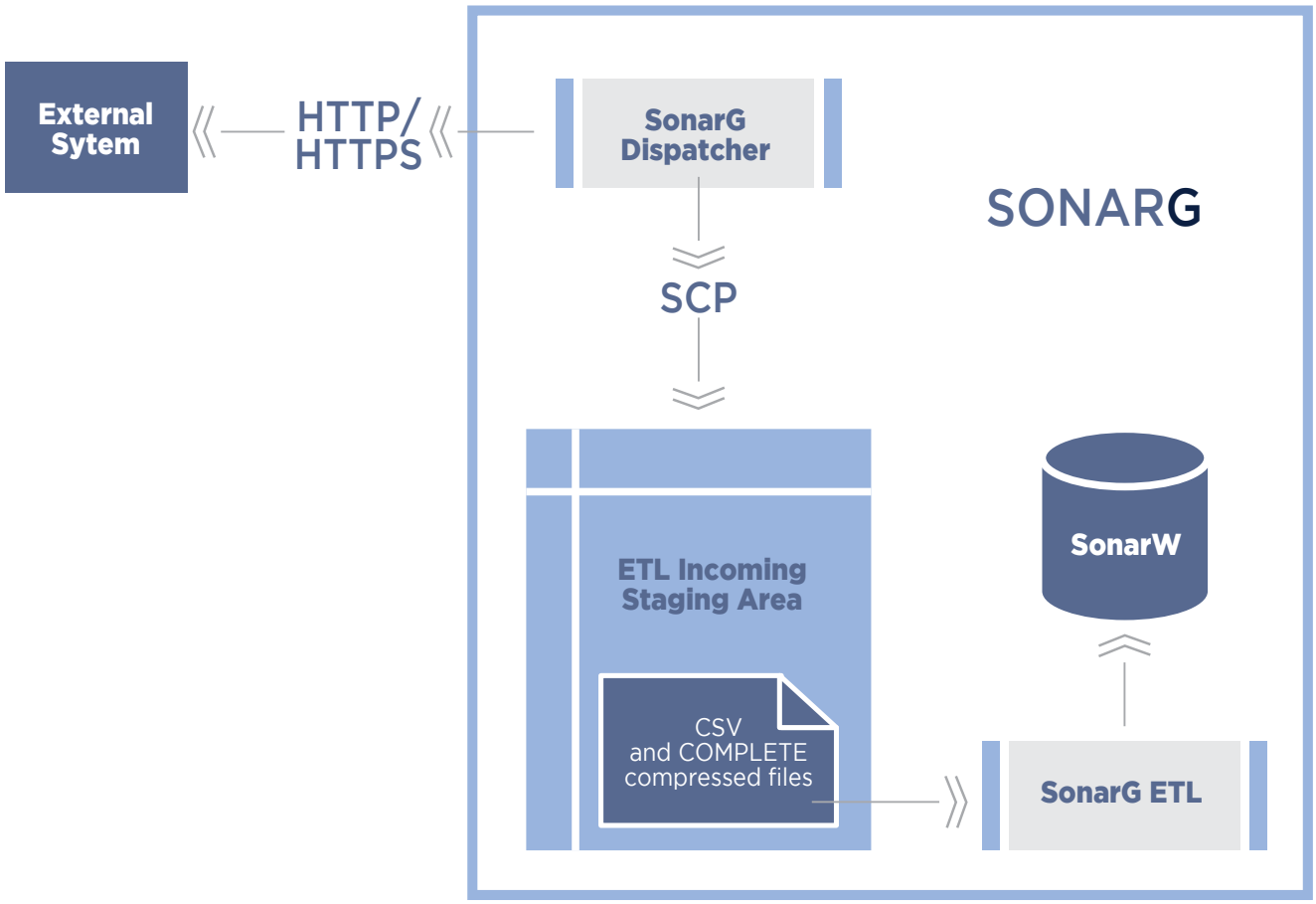


Figure 6 – Scheduled Pull of CSV Files

The following steps need to be followed in order to use this ETL process:

1. Setup the external system by adding two entries into the dispatcher.conf file located in /opt/sonarfinder/sonarFinder. One entry is to add a section to the config file (e.g. [ExternalSystem]) that sets the authURL value to specify how you will authenticate to the external system. The second section is to specify what collection the new data will go into and the directory location of the built-in SonarG ETL. This process uses the same ETL process as outlined in option one, but relies on the dispatcher to pull the CSV file into SonarG and feed this into the built-in ETL layer.

An example copy section is:

```
[externalCopy]
copy_host = localhost
copy_port = 22
copy_username = ..
copy_password = ..
copy_keyfile =
copy_dest = /var/lib/sonarg/incoming
filePattern = my_collection_name
```

2. Setup the scheduled job with a specific reference to the external system so that the dispatcher knows to use the correct authURL. To add a schedule you insert a document into the lmrm__scheduled_jobs collection in the lmrm__scheduler in SonarG. The two important things in this document are: a) the copies section which has the value of the copy section in dispatcher.conf (external Copy in this example) and b) the URL have a parameter name db that references the value specified in dispatcher.conf which is used to get the authURL (externalSystem) in this example.

For example, you can login using the shell:

```
mongo <sonarg host>:27117/lmrm__scheduler -u admin -p  --authenticationDatabase admin
```

and insert a document into lmrm__scheduled_jobs such as:

```
{
  “cron” : “0 14 * * * ? *”,
  “header” : “”,
  “footer” : “”,
  “subject” : “”,
  “emailType” : “”,
  “emails” : “”,
  “copies” : “ externalCopy “,
  “type” : “CSV”,
  “bind_collection” : “”,
  “sendIfEmpty” : false,
  “user” : “admin”,
  “name” : “external pull”,
  “ts” : ISODate(“2016-03-16T19:11:49.642Z”),
  “url” : “https://<exteral system host>/<external system url>&db=externalSystem”
}
```

3. Restart the SonarG scheduler using “sudo service sonarfinder restart” to update the schedule definitions in order to include the new schedule introduced in step 2 above.

Additional details on the scheduling process can be found within the JSON Studio documentation at http://jsonstudio.jsonar.com/scheduler.html?highlight=scheduler Note that JSON Studio is embedded within the SonarG solution to provide a variety of capabilities around scheduling, publishing gateways, visualization, etc.

1.4: SonarG Excel Interface for Manual Ingestion

The above three options all provide ETL mechanisms for automatically delivering data from external sources into the SonarG system. Occasionally there may be a need to manually ingest data from an external source and SonarG provides an easy to use, flexible mechanism for interfacing with Excel spreadsheets via the Analyze panel within SonarG. Steps for this flow are outlined below:

1. Invoke the Spreadsheet module within the Analyze panel to get to the configuration options.

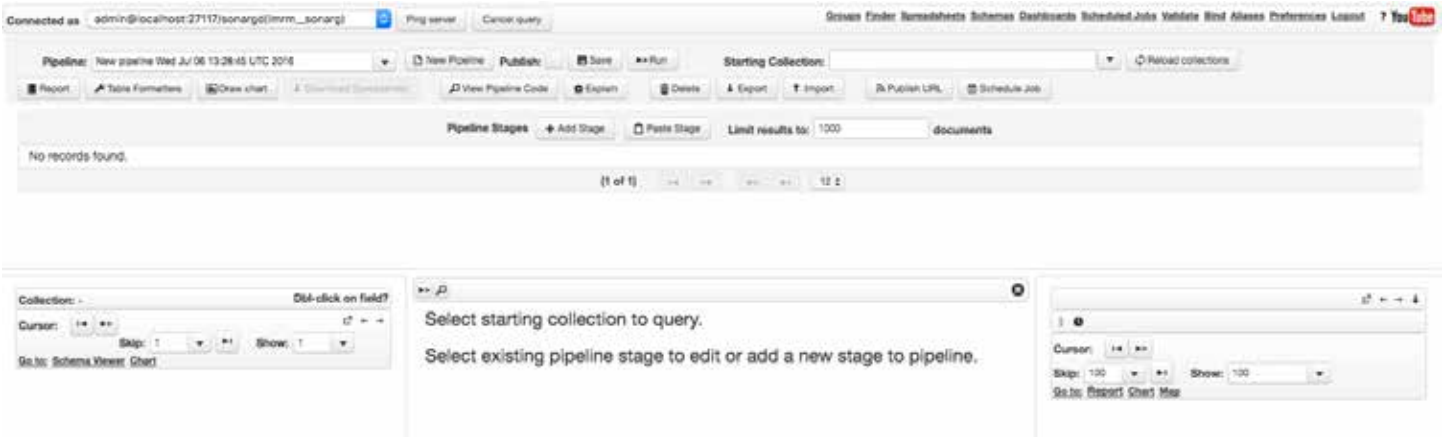


Figure 7 - Invoking the SonarG Spreadsheet Interface

2. Configure the appropriate parameters for Excel filename, target collection, etc. and then click the "Upload Spreadsheet" button.

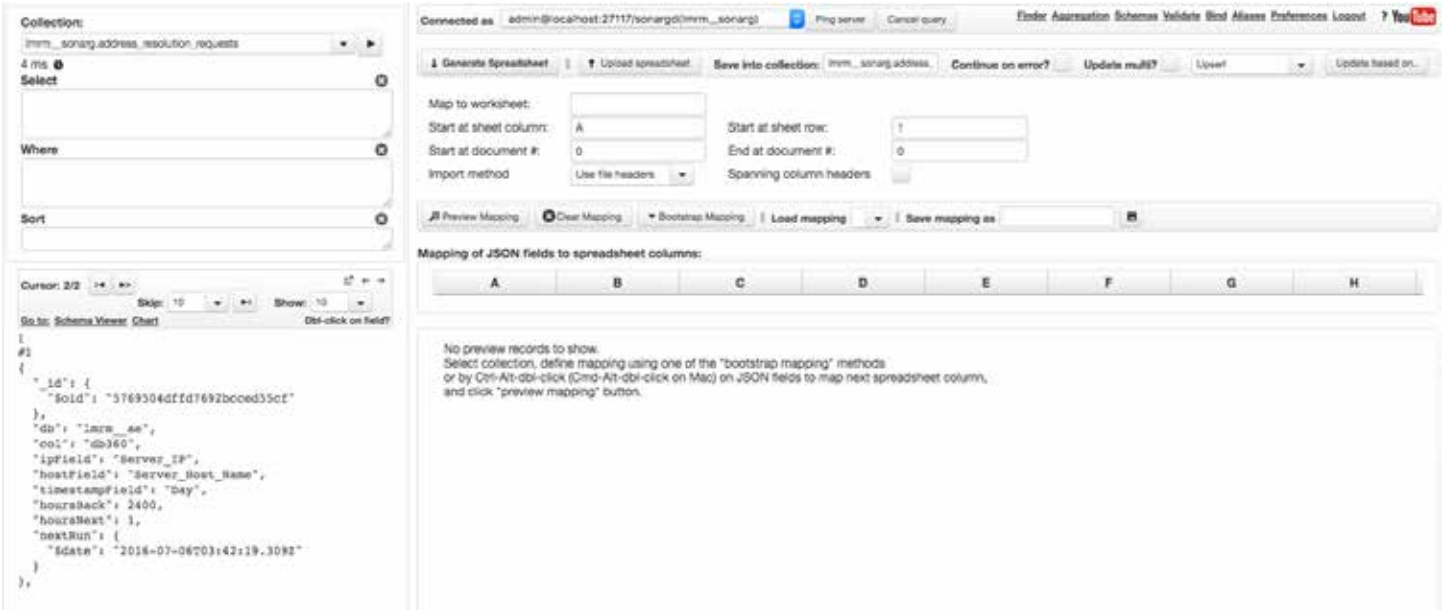
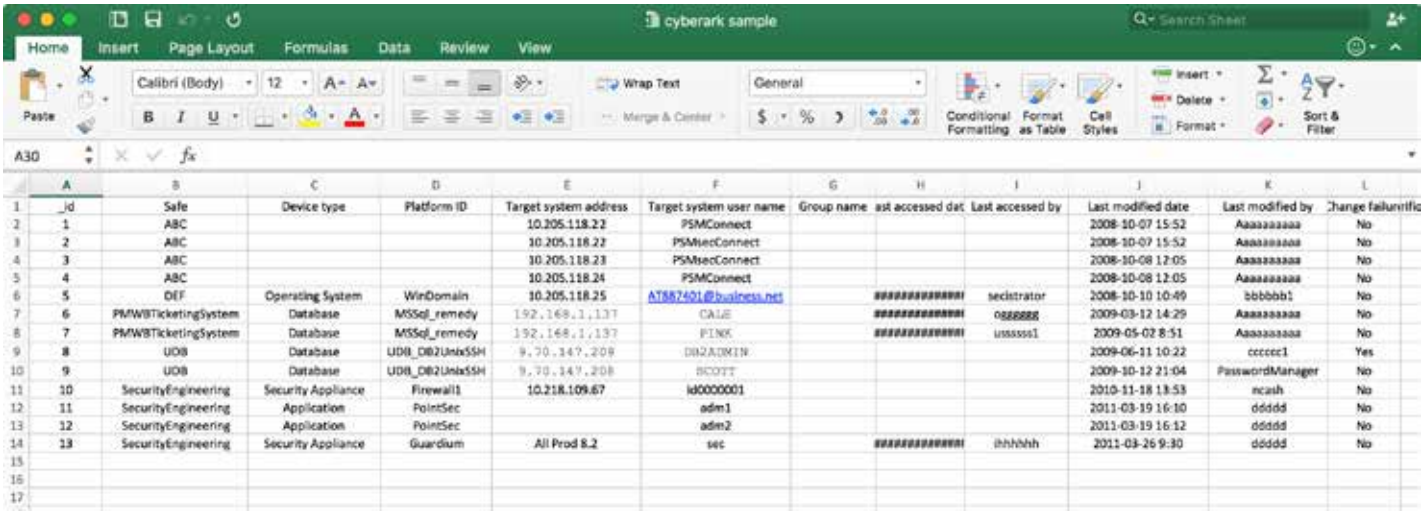


Figure 8 - Configuring the Spreadsheet Parameters

3. Upload the selected Excel contents as shown in Figure 9 into the target collection and the data is immediately processed into a new collection in SonarG, a sample of which is shown in Figure 10, which represents row 9 of the .xls.



	A	B	C	D	E	F	G	H	I	J	K	L
1	_id	Safe	Device type	Platform ID	Target system address	Target system user name	Group name	ast accessed dat	Last accessed by	Last modified date	Last modified by	Change failurrific
2	1	ABC			10.205.118.22	PSMConnect				2008-10-07 15:52	Aaaaaaaaaa	No
3	2	ABC			10.205.118.22	PSMsecConnect				2008-10-07 15:52	Aaaaaaaaaa	No
4	3	ABC			10.205.118.23	PSMsecConnect				2008-10-08 12:05	Aaaaaaaaaa	No
5	4	ABC			10.205.118.24	PSMConnect				2008-10-08 12:05	Aaaaaaaaaa	No
6	5	DEF	Operating System	WinDomain	10.205.118.25	AT887401@business.net		##########	secistrator	2008-10-10 10:49	bbbbbb1	No
7	6	PMWBTicketingSystem	Database	MSSql_remedy	192.168.1.137	CAUE		##########	########	2009-03-12 14:29	Aaaaaaaaaa	No
8	7	PMWBTicketingSystem	Database	MSSql_remedy	192.168.1.137	PINK		##########	ussssss1	2009-05-02 8:51	Aaaaaaaaaa	No
9	8	UDB	Database	UDB_DB2UnixSSH	9.70.147.209	DB2ADMIN				2009-06-11 10:22	ccccc1	Yes
10	9	UDB	Database	UDB_DB2UnixSSH	9.70.147.208	SCOTT				2009-10-12 21:04	PasswordManager	No
11	10	SecurityEngineering	Security Appliance	Firewall1	10.218.109.67	id0000001				2010-11-18 13:53	ncash	No
12	11	SecurityEngineering	Application	PointSec		adm1				2011-03-19 16:10	ddddd	No
13	12	SecurityEngineering	Application	PointSec		adm2				2011-03-19 16:12	ddddd	No
14	13	SecurityEngineering	Security Appliance	Guardium	All Prod 8.2	sec		##########	ihhhhhh	2011-03-26 9:30	ddddd	No
15												
16												
17												

Figure 9 - Excel Spreadsheet with CyberArk Data


```
{
  "Session Start": {
    "$date": "2016-06-12T00:04:52.000Z"
  },
  "Session End": {
    "$date": "2016-06-12T00:04:52.000Z"
  },
  "Server IP": "9.70.147.209",
  "Platform_ID": "UDB_DB2UnixSSH",
  "DB User Name": "DB2ADMIN",
  "Last_modified_by": "cccccc1",
  "Safe": "UDB",
  "Change_failure": "Yes",
  "Service Name": "DB2",
  "Source Program": "DB2HMON",
  "Analyzed Client IP": "9.70.147.209"
},
```


Figure 10 - CyberArk Collection in SonarG from .xls