Monitoring metadata refresh

The SP Proxy (https://login.terena.org/wayf) uses metadata that is generated by another host: https://pioneer.terena.org/mr.

This host polls various metadata sources from cron, using SimpleSAMLphp's metarefresh module.

The reason for this dual VM is that the metarefresh module sometimes would get stuck validating XML signatures, and then the SP proxy would also hang, which sucks.

This behaviour has been fixed, but the dual VM set-up is still a good idea, so it will stay this way.

When pioneer.terena.org/mr has finished, it uses rsync to synchronise the directory that holds the metadata to login.terena.org/wayf.

pioneer.terena.org/mr is currently configured to poll a couple of dozen URLs for metadata.

This process works, and it has been improved so that when there are errors with the refreshing/polling, the previous/cached metadata is re-used, instead of it being nuked.

This is already an improvement, and when it happens, an error is logged, however this requires an administrator to look at the log files, which does not happen.

So, the error goes unnoticed, and eventually the cached metadata will expire, because it has a set lifetime embedded.

At this stage IdPs will start to disappear from the metadata until it's completely empty, until there are no valid entries any more and the complete set will have disappeared.

At this point the service will be unavailable.

During the SimpleSAMLphp hackathon on 27 May 2015 an attempt has been made to add some form of monitoring capabilities to metarefresh.

This can be done in many ways, however not all were considered:

- Write any custom Nagios checks and let the Nagios server poll the URLs. This option was rejected because the network checks should be done from the same host that does the metarefresh process. Otherwise subtle changes in network access lists, firewall, OS and library versions might yield different results.
- Run separate Nagios checks through/from the same host that does the metarefresh process. This approach was rejected because with dozens of URLs, any non-responsive URLs would significantly increase the time it takes to run the Nagios plugin.

In the end the follow approach was chosen. The metarefresh process already logs all possible errors, and also stores any Conditional GET values for each URL in a state file.

By slightly adapting the metarefresh module, it was possible to create an additional state file that holds error information about the URL.

This state file is then parsed by Nagios through check_by_ssh. This is not the best approach security wise, and could later be improved so that the data is exposed over HTTPS and protected by supplying a secret variable.

Several issues were encountered:

- When the metarefresh config is changed, for example URLs are added/deleted/changed, the Nagios configuration needs to be changed as well, the Nagios process needs to be reloaded, and a forced check of the services needs to be scheduled.
- The name of the service would be the src URL, but this contains lots of 'illegal' characters as far a Nagios is concerned. A first attempt to fix work around this was by stripping the path from the URL. This means that the name would be just the base URL, which makes sense to admin, but does not (yet?) contain illegal characters. However, this turned out to be a problem because metadata is polled from several sources that reside on the same base URL (http://my.org/saml/idp1,http://my.org/saml/idp2, etc). The base URL is the same in this case, which is also not allowed by Nagios. The final work-around was to append a hash of the URL to the base URL. This way it is easy to distinguish what domain/IdP has an issue, and the string will be unique for each URL
- Because the way metarefresh is implemented, the URL state file is being overwritten for each URL, and only at the end will it yield the full status array. This causes a race condition with the Nagios check, it can happen that the Nagios check for a specific URL happens during the metarefresh, when there is no status information yet on that URL. A work-around was implemented by scheduling a simple cp command at the end of metarefresh cronjob. This will make sure the state file will always have all the URLs.

Because the logic of metarefresh is used, the same errors can be detected. This comes down to any connection errors, and all HTTP response codes other than 200 (OK) and 304 (Not Modified).

A few basic tests were done to confirm the functionality, such as having a URL return an HTTP 500 error:

pioneer	Mailqueue	OK	2015-05-27 16:18:16	8d 9h 46m 36s	1/4	OK: Mail queue is empty
	Root partition	OK	2015-05-27 16:18:16	1d 15h 33m 45s	1/4	DISK OK - free space: / 3929 MB (58% inode=79%):
	SMTP	OK	2015-05-27 16:18:15	8d 9h 41m 20s	1/4	SMTP OK - 0.051 sec. response time
	SSH	OK	2015-05-27 16:19:15	0d 0h 44m 18s	1/4	SSH OK - OpenSSH_5.9p1+sftpfilecontrol-v1.3+LdapPublicKeys-v0.3.20 Debian-5ubuntu1terena3 (protocol 2.0)
	Swap space	OK	2015-05-27 16:20:15	0d 1h 13m 18s	1/4	SWAP OK - 66% free (640 MB out of 975 MB)
	X509 cert signature algorithm (HTTPS over IPv6)	OK	2015-05-27 16:18:15	8d 9h 45m 9s	1/4	SSL_CERT OK - certificate signature is sha256WithRSAEncryption
	X509 cert signature algorithm (SMTP-TLS over IPv6)	OK	2015-05-27 16:18:15	8d 9h 41m 25s	1/4	SSL_CERT OK - certificate signature is sha256WithRSAEncryption
	X509 cert validity (SMTP-TLS)	OK	2015-05-27 16:18:15	8d 9h 41m 59s	1/4	OK - Certificate will expire on 11/27/2017 23:59.
	http://eduid.at/.../e23362b0e9e054a78981b9b15f6902d5	OK	2015-05-27 16:10:15	0d 1h 13m 18s	1/4	OK - Received HTTP 304 (Not Modified) - attempting to re-use cached metadata
	http://idp.myifam.upm.my/.../8ecaed0b398c14ae08457c3d305db7bb	OK	2015-05-27 16:10:15	0d 1h 13m 18s	1/4	OK - Received HTTP 304 (Not Modified) - attempting to re-use cached metadata
	http://md.test.eid2.se/.../a754336d80ea18747a8c2563264d9dba	OK	2015-05-27 16:10:15	0d 1h 13m 18s	1/4	OK
	http://md.unitedid.org/.../2940689b1e256c678698b5fdc4ab48eb	OK	2015-05-27 16:10:15	0d 1h 13m 18s	1/4	OK
	http://wayf.up.pt/.../34d520c6a633016d2f791768fd9b0c03	OK	2015-05-27 16:10:15	0d 1h 13m 18s	1/4	OK
	http://www.protectnetwork.org/.../d18ae38d97ed238af82565f5ad74fbbf	OK	2015-05-27 16:09:15	0d 1h 14m 18s	1/4	OK - Received HTTP 304 (Not Modified) - attempting to re-use cached metadata
	https://cajones.terena.org/.../c06a3f724239639bf9151f3df1ccfff0	OK	2015-05-27 16:10:15	0d 0h 43m 18s	1/4	OK
	https://gidp.geant.net/.../4961836103fead12a0adec230b98deb5	OK	2015-05-27 16:09:15	0d 1h 14m 18s	1/4	OK
	https://idp.cedia.org.ec/.../7f307827ba51ebe5c29131fa6aa2b577	OK	2015-05-27 16:09:15	0d 1h 14m 18s	1/4	OK - Received HTTP 304 (Not Modified) - attempting to re-use cached metadata
	https://idp.dfn-cert.de/.../dffa7ce40d511c9518155caba4820884	OK	2015-05-27 16:09:15	0d 1h 14m 18s	1/4	OK
	https://idp.ens-cachan.fr/.../f7f69faf9bcafa9025b2aa639b7fdb6c	OK	2015-05-27 16:09:15	0d 1h 14m 18s	1/4	OK
	https://idp.marwan.ma/.../0efec56df97fbffd60b631328e4a7e58	OK	2015-05-27 16:09:15	0d 1h 14m 18s	1/4	OK
	https://idp.renata.edu.co/.../ea4d513ac0def9b6607e129f37819006	OK	2015-05-27 16:09:15	0d 1h 14m 18s	1/4	OK
	https://idf.iucc.ac.il/.../19c488fdf5ffaaf1e8ef73f85760aab2	OK	2015-05-27 16:09:15	0d 1h 14m 18s	1/4	OK
	https://laife.lanet.lv/.../a99f608c853df0c3faad17441f83acd4	OK	2015-05-27 16:09:15	0d 1h 14m 18s	1/4	OK - Received HTTP 304 (Not Modified) - attempting to re-use cached metadata
	https://login.iamres.amres.ac.rs/.../a54fc7a8fa15671c3ca4a0f9639bc35f	OK	2015-05-27 16:08:15	0d 1h 33m 15s	1/4	OK
	https://login.terena.org/.../add0aa33e5963c5406d2b82f6b1d9f58	OK	2015-05-27 16:08:15	0d 1h 31m 50s	1/4	OK
	https://login.terena.org/.../f92c1fcbb8a060b7262bc64bd17be670	OK	2015-05-27 16:08:15	0d 1h 30m 25s	1/4	OK
	https://login.vu.lt/.../173d9f5afd12fca6b7743c8da4d374ba	OK	2015-05-27 16:08:15	0d 1h 29m 0s	1/4	OK
	https://openidp.feide.no/.../11bf29579979267b38ac0c0df11745bf	OK	2015-05-27 16:08:15	0d 1h 27m 36s	1/4	OK
	https://rctsaai-rr.fccn.pt/.../cdec4a7b7321538b2b7d1d0120b15230	OK	2015-05-27 16:08:15	0d 1h 40m 17s	1/4	OK
	https://safid-idp.c4.csir.co.za/.../ba3a237fccada78c644bc51954e9d6de	OK	2015-05-27 16:08:15	0d 1h 38m 52s	1/4	OK
	https://sp.isragrid.org.il/.../f46a256981df9d8ec102cd2e58946ff6	OK	2015-05-27 16:08:15	0d 1h 37m 28s	1/4	OK
	https://tasman.terena.org/.../52c8949991caf9a1897a3417e5dfc965	CRITICAL	2015-05-27 16:08:15	0d 1h 36m 3s	4/4	CRITICAL - No response (Error fetching 'https://tasman.terena.org/simplesaml/saml2/idp/metadata.php':file_get_contents(https://tasman.terena.org/simplesar failed to open stream: HTTP request failed! HTTP/1.0 500 Internal Server Error
	https://wierenga.refeds.org/.../460e208c60f96934a1565ef4bed30a34	OK	2015-05-27 16:08:15	0d 1h 34m 38s	1/4	OK

Based on the experiences, some recommendations:

- Using the current metarefresh module has it's drawbacks. Notably, the race condition that exists in writing the state file is not a problem for the Conditional-GET state, is a real problem for the Nagios state file, and it not easy to fix. So it will be worth doing separate checks.
- blah