

LDAP crawlers use cases, dangers and how to cope with them

2nd OpenLDAP Developers Day,
Vienna, July 18, 2003

Peter Gietz
peter@daasi.de

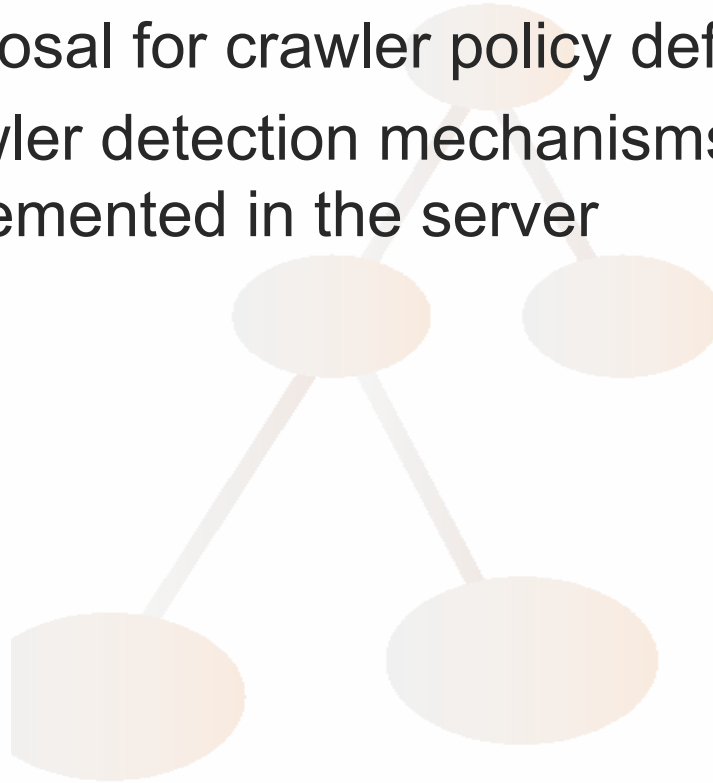
DAASI
International

Directory Applications
for Advanced Security
and Information Management



Agenda

- What can Crawlers do?
- Proposal for crawler policy definition
- Crawler detection mechanisms that could be implemented in the server



What is an LDAP crawler?

- Just like a web crawler it crawls as much data as it can
- To circumvent sizelimits a crawler does not do subtree searches, but one level searches
- They follow all referrals and can start crawling the whole naming context of the referred to server
- They could use DNS SRV records to find additional servers

Good or bad?

- Thus crawlers can be a threat
 - when used by bad people, like spammers
 - But only on a public server with anonymous access
 - Or they could even hack non public servers
- From privacy perspective there is a difference between
 - reading single public entries and
 - If it stills hits a size or time limit, it can do partial searches like (cn=a*), (cn=b*), etc.
 - If it still hits a limit, it can do (cn=aa*), (cn=ab*), etc.
 - And so on ...
 - getting all the data into one file
- Crawlers can be usefull though, e.g. in an indexing context

Who uses crawlers?

- By now crawlers are mostly used by „good people“ that work on indexing services
- Sometimes web crawlers hit a web2LDAP gateway and get data
- I have not yet seen an LDAP crawler that was used by a spammer
 - E.g.: the email directory for the German research community (AMBIX, ambix2002.directory.dfn.de)
 - we store fake entries
 - with working email-addresses
 - that were never published elsewhere
 - We never got any spam at those email addresses!
- This might change in future!!!

So what can be done against that threat?

- To distinguish between „good“ and „bad“ crawlers
 - we should use a similiar approach as in the Web: robots.txt
 - Crawler policy tells the crawler which behaviour is wanted and which is not
 - LDAP crawler policy should be stored in the LDAP server
- To block „bad“ crawlers that don't care about the policy, we additionally need crawler detection

Crawler policy proposal

- Originally part of the specs of the SUDALIS crawler (a crawler developed by DAASI for Surfnet)
- Implementation of crawler policy is on its way
- Strategy was to have a very flexible way to define policy
- The current proposal is quite old and may need a little renovation since a lot has changed in the LDAP world since then
- I didn't have time yet to do this update
- Your comments are most welcome
- I will put the specs document online

Root DSE Attributes

```
( sudalis-attributetypes.1  
NAME 'supportedCrawlerPolicies'  
EQUALITY objectIdentifierMatch  
SYNTAX numericOID  
USAGE directoryOperation )
```

- Just in case that there will be different crawlerpolicy formats

Root DSE Attributes contd.

```
( sudalis-attributetypes.2  
NAME 'indexAreas'  
EQUALITY distinguishedNameMatch  
SYNTAX DN  
USAGE directoryOperation )
```

- Pointer to the subtrees that are to be indexed
- All other parts of the DIT are to be ignored by the crawler
- If this attribute is empty, the naming contexts are crawled instead

Object class indexSubentry

```
( sudalis-objectclasses.1
NAME 'indexSubentry'
DESC 'defines index crawler policy'
SUP TOP
STRUCTURAL
MUST ( cn )
MAY ( indexCrawlerDN $
      indexCrawlerAuthMethod $
      indexObjectClasses $ indexAttributes
      $ indexFilter $ indexAreaLevels $
      indexCrawlerVisitFrequency $
      indexDescription ) )
```

- These entries specify the policy
- Alternative could be SUP subentry



Attribute Type indexCrawlerDN

```
( sudalis-attributetypes.3  
NAME 'indexCrawlerDN'  
EQUALITY distinguishedNameMatch  
SYNTAX DN )
```

```
# USAGE directoryOperation )
```

- Defines for which crawler(s) this policy is meant
- Several subentries for different crawlers
- If this attribute is empty, the policy is meant for all crawlers
- If not empty, the crawler has to bind with this DN

Attribute Type `indexCrawlerAuthMethod`

```
( sudalis-attributetypes.4  
NAME 'indexCrawlerAuthMethod'  
SYNTAX directoryString  
EQUALITY caseIgnoreMatch )  
# USAGE directoryOperation )
```

- Defines the authentication method the crawler has to use

Attribute Type indexObjectClasses

```
( sudalis-attributetypes.5  
NAME 'indexObjectClasses'  
SYNTAX OID  
EQUALITY objectIdentifierMatch )  
# USAGE directoryOperation )
```

- Defines which objectclass attribute values to include in the index.
- No Filter criteria!
- Models the LDIF entry that is to be put into the index
- Needed to prevent the crawler from storing internal objectclasses into the index



Attribute Type indexAttributes

```
( sudalis-attributetypes.6  
NAME 'indexAttributes'  
SYNTAX OID  
EQUALITY objectIdentifierMatch )  
# USAGE directoryOperation )
```

- Defines which attributes to crawl.
- The crawler must not take any other attributes

Attribute Type indexFilter

```
( sudalis-attributetypes.7  
NAME 'indexFilter'  
SYNTAX directoryString  
EQUALITY caseExactMatch  
SINGLE-VALUE )  
# USAGE directoryOperation )
```

- Filter that **MUST** be used by the crawler

Attribute Type indexAreaLevels

```
( sudalis-attributetypes.8  
NAME 'indexAreaLevels'  
SYNTAX INTEGER  
EQUALITY integerMatch  
SINGLE-VALUE )  
# USAGE directoryOperation )
```

- Number of hierarchy levels to crawl
- If 0 the crawler **MUST** leave this subtree of the DIT
- If empty no restrictions for the crawler as to the depth

Attribute Type `indexCrawlerVisitFrequency`

```
( sudalis-attributetypes.9
NAME 'indexCrawlerVisitFrequency'
SYNTAX INTEGER
EQUALITY integerMatch
SINGLE-VALUE )
# USAGE directoryOperation )
```

- defines how often the data of the specified subtree are to be crawled
- The value represents a timeperiod in seconds
- The crawler MAY crawl less frequent
- but MUST NOT crawl more frequent than stated here



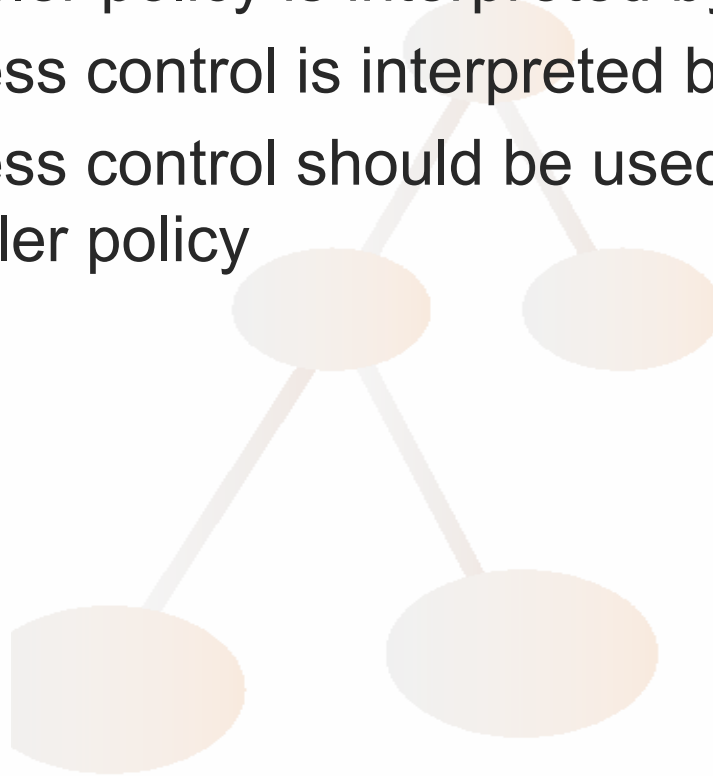
Attribute Type indexDescription

```
( sudalis-attributetypes.10  
NAME 'indexDescription'  
SYNTAX directoryString  
EQUALITY caseExactMatch  
SINGLE-VALUE )  
# USAGE directoryOperation )
```

- Human readable description of the policy defined in the subentry

Crawler Policy and Access control

- Crawler policy is interpreted by client
- Access control is interpreted by server
- Access control should be used to enforce crawler policy



Crawler registration

- A crawler can register to a server by providing the following data:
 - Name of the Crawler
 - Description of the index the crawler collects data for
 - URI where to access the index
 - Pointer to a privacy statement about how the data will be used. This statement should comply to the P3P standard (<http://www.w3.org/P3P/>)
 - Email address of the crawler manager
 - Method and needed data (public key) for encrypted email (PGP or S/MIME)

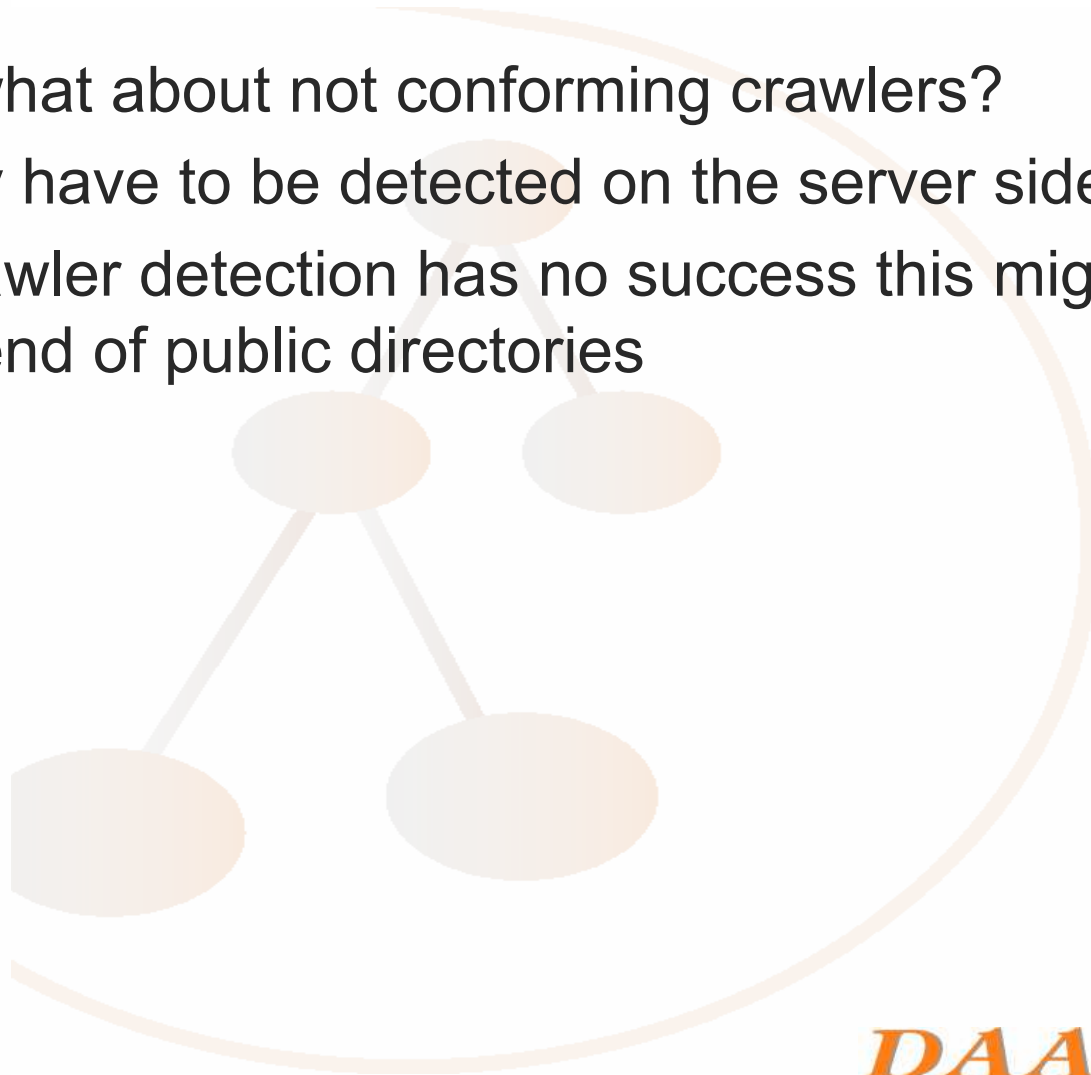


Crawler registration contd.

- The data from the crawler will be entered in a dedicated entry, together with additional information:
 - Date of registration
 - Pointer to the person who made the decision
 - Date of last visit of the crawler
 - ...
 - Password
- This entry will be used for the indexCrawlerDN

Crawler Detection

- So what about not conforming crawlers?
- They have to be detected on the server side.
- If crawler detection has no success this might be the end of public directories



Crawler characteristics

- IP address can be used for identification
- Regular requests over a certain period of time
- Humans are slower, so more than X requests per 10 seconds must be a crawler
- Patterns in searching:
 - Onelevel search at every entry given back in the former result
 - „(cn=a*)“, etc.
- Known Spammer IPs could be blocked
- Known spamming countries could be blocked as well

The further future

- What if intelligent crawlers will try to hide their characteristics?
 - Random sleep in between requests
 - ...
- How public should this discussion take place?
- What else could we do ?
- Ideas wanted and needed !
- Even more needed: Implementation of simple crawler detection as describe above in OpenLDAP !

BTW: Schema Registry Project update

- Work almost finished
- BarBoF Meeting at the IETF, deciding:
 - Either publish old drafts as RFC
 - Or Chris and Peter work on the drafts before to resolve some issues
 - There is currently no need for a new WG
- Project Web-site now is: www.schemareg.org
 - Service still experimental
 - Pilot service will start beginning of August
- Last document on business model will come soon too

Thanks for your attention 😊

➤ Any Questions?

➤ More info at:

- Info@daasi.de

