MISP CORE DEVELOPMENT CRASH COURSE

HOW I LEARNED TO STOP WORRYING AND LOVE THE PHP

CIRCL / TEAM MISP PROJECT



13TH ENISA-EC3 WORKSHOP



MISP core development crash course

8

2024-10

MISP CORE DEVELOPMENT CRASH COURSE HOW I LEARNED TO STOP WORRYING AND LOVE THE PHP



Some things to know in advance...

10-02

2024

MISP core development crash course

└─Some things to know in advance...

SOME THINGS TO KNOW IN ADVANCE ...

MISP is based on PHP 73+
 Using the MVC framework CakePHP 2.x
 What we'll look at now will be a quick glance at the structuring / layout of the code

MISP is based on PHP 7.3+

- Using the MVC framework CakePHP 2.x
- What we'll look at now will be a quick glance at the structuring / layout of the code

MVC FRAMEWORKS IN GENERAL

MISP core development crash course

└─MVC frameworks in general

- separation of business logic and views, interconnected by controllers
 main advantage is clear separation of the various
- lean controllers, fat models (kinda...)
- No interaction between Model and Views, even

- separation of business logic and views, interconnected by controllers
- main advantage is clear separation of the various components
- lean controllers, fat models (kinda...)
- domain based code reuse
- No interaction between Model and Views, ever

8

9

2024

STRUCTURE OF MISP CORE APP DIRECTORIES

2024-10-0

MISP core development crash course

└─Structure of MISP Core app directories

STRUCTURE OF MISP CORE APP DIRECTORIES

- Config: general configuration files
 Console: command line tools
 Controller: Code dealing with requests/responses,
 generating data for views based on interactions with the
 models
 Libs Commic reusable code / libraries
 Model: Business logic, data gathering and modification
- Plugin: Alternative location for plugin specific coc ordered into controller, model, view files
- View: UI views, populated by the controller

- Config: general configuration files
- Console: command line tools
- Controller: Code dealing with requests/responses, generating data for views based on interactions with the models
- Lib: Generic reusable code / libraries
- Model: Business logic, data gathering and modification
- Plugin: Alternative location for plugin specific codes, ordered into controller, model, view files
- View: UI views, populated by the controller

CONTROLLERS - SCOPE

- Each public function in a controller is exposed as an API action
- request routing (admin routing)
- multi-use functions (POST/GET)
- request/response objects
- contains the action code, telling the application what data fetching/modifying calls to make, preparing the resulting data for the resulting view
- grouped into controller files based on model actions
- Accessed via UI, API, AJAX calls directly by users
- For code reuse: behaviours
- Each controller bound to a model

MISP core development crash course

Controllers - scope

CONTROLLERS - SCO

- Each public function in a controller is exposed as an API action
 request routing (admin mouting)
 multi-use functions (POST/GET)
 request/response objects
 request/response objects
 tending modifying calls to make, preparing the resulting data for the requesting work
- grouped into controller files based on model action
 Accessed via UI, API, AJAX calls directly by users
- Each controller bound to a model

3

CONTROLLERS - FUNCTIONALITIES OF CONTROLLERS

2024-10-02

Controllers - functionalities of controllers

MISP core development crash course

pagination functionality
 logging functionality
 Controllers actions can access functionality / variables of

Controllers actions can access functionality / variables of todels Controllers cannot access code of other controller actions kind of...) ccess to the authenticated user's data perforeFilter(), afterfilter(), methods approaches actionations perforeFilter(), afterfilter() methods perforeFilter()

- pagination functionality
- logging functionality
- Controllers actions can access functionality / variables of Models
- Controllers cannot access code of other controller actions (kind of...)
- Access to the authenticated user's data
- beforeFilter(), afterFilter() methods
- Inherited code in AppController

CONTROLLERS - COMPONENTS

MISP core development crash course

Controllers - components

- Components = reusable code for Controllers
- Authentication components
 RestResponse component
- ACL component
- Clar component
 IOCImport component (should be n

Components = reusable code for Controllers

- Authentication components
- RestResponse component
- ACL component
- Cidr component
- IOCImport component (should be moved)

3

CONTROLLERS - ADDITIONAL FUNCTIONALITIES

MISP core development crash course

-Controllers - additional functionalities

- Handling API responses (RestResponseComponent) ACL managem CRUD Componer
- important: quertString/PvMISP versions_MISP version
- future improvements to the export mechanism

- Handling API responses (RestResponseComponent)
- Handling API requests (IndexFilterComponent)
- auth/session management
- ACL management
- CRUD Component
- Security component
- important: quertString/PyMISP versions, MISP version handler
- future improvements to the export mechanisms

8

9

MODELS - SCOPE

2024-10-02

MISP core development crash course

└─Models - scope

- Controls anything that has to do with:
- Ending subsets of data
- altering existing data
 inherited model: AnnMo
- reusable code for models: Behavi
- regex, trim

Controls anything that has to do with:

- finding subsets of data
- altering existing data
- inherited model: AppModel
- reusable code for models: Behaviours
- regex, trim

MODELS - HOOKING SYSTEM

MISP core development crash course

└─Models - hooking system

- Versatile hooking system
- manipulate the data at certain stages of execution
 code can be located in 3 places: Model hook, AppModel hool

- Versatile hooking system
 - manipulate the data at certain stages of execution
 - code can be located in 3 places: Model hook, AppModel hook, behaviour

10-02

2024

MODEL - HOOKING PIPELINE (ADD/EDIT)

MISP core development crash course

└─Model - hooking pipeline (add/edit)

- Hooks / model pipeline for data creation / edits
- beforeValidate() (lowercase all hashes)
 - validate() (check hash format)
 aftertialidate() fine prove use it
- could be interesting if we ever validated without say
- beforeSave() (purge existing correlations for an attribute afterSave() (create new correlations for an attribute / zm

Hooks / model pipeline for data creation / edits

- beforeValidate() (lowercase all hashes)
- validate() (check hash format)
- afterValidate() (we never use it
- could be interesting if we ever validated without saving)
- beforeSave() (purge existing correlations for an attribute)
- afterSave() (create new correlations for an attribute / zmq)

8

MODELS - HOOKING PIPELINE (DELETE/READ)

2024-10-02

MISP core development crash course

└─Models - hooking pipeline (delete/read)

- Hooks for deletions
- beforeDelete() (purge correlations for an attribute
 - Atterbesete() (2mg)
 Hooks for retrieving data
 - beforeFind() (modify the find parameters before execution
 - afterFind() (json decode json fields)

Hooks for deletions

- beforeDelete() (purge correlations for an attribute)
- afterDelete() (zmq)

Hooks for retrieving data

- beforeFind() (modify the find parameters before execution, we don't use it)
- afterFind() (json decode json fields)

1

MISP core development crash course

└─Models - misc

3

2024-10

code to handle version upgrades contained in AppModel
 generic cleanup/data migration tools
 centralised redis/pubsub handlers
 (Show example of adding an attribute with trace)

- code to handle version upgrades contained in AppModel
- generic cleanup/data migration tools
- centralised redis/pubsub handlers
- (Show example of adding an attribute with trace)

VIEWS - SCOPE AND STRUCTURE

MISP core development crash course

3

2024-10

└─Views - scope and structure

- templates for views
- layouts
 reusable template code: elements
- reusable template code: element
 attribute list, rows (if reused)
- reusable code: helpers
- commandhelper (for discussion boards), highlighter f searches, tag colour helper
- views per controller

templates for views

layouts

- reusable template code: elements
 - attribute list, rows (if reused)

reusable code: helpers

- commandhelper (for discussion boards), highlighter for searches, tag colour helper
- views per controller

VIEWS - TYPES OF VIEWS AND HELPERS

2024-10-02

MISP core development crash course

└─Views - Types of views and helpers

VIEWS - TYPES OF VIEWS AND HELPERS

a ajax views vs normal views
data views vs normal views vs serialisation in the controller
sanitisation h()
reaching forms
sanitisation
b CSR6

- ajax views vs normal views
- data views vs normal views vs serialisation in the controller
- sanitisation h()
- creating forms
 - sanitisation
 - CSRF

VIEWS - GENERATORS

MISP core development crash course

└─Views - Generators

8

-9

2024

- Mostly in genericElements
- Preparing the move to Cakes
 Important ones
- Form generate forms in a standardised way (/add, /edit,
- IndexTable index lists using Field templates (/index, etc
 SingleViews key-value lists with child elements (/view, etc)
- Menues to be refactored, see Cerebrate

Mostly in genericElements

- Preparing the move to Cake4
- Important ones
 - Form generate forms in a standardised way (/add, /edit, etc)
 - IndexTable index lists using Field templates (/index, etc)
 - SingleViews key-value lists with child elements (/view, etc)
 - Menues to be refactored, see Cerebrate

MISP core development crash course

8

9

2024-

General reusable libraries

- Code that is to be reused across several layer
- Important ones
- Dashboard Dashboard widget backend
 EventReport Report generation
- EventNeport Report generation
 Export MISP -> external format converter modul
- Tools List of generic helper libraries examples:
 Attachment, ISON conversion, random generation, en
 - sync request generation Kaffa, 2MQ, AVS 53, Elastic integration, PGP encryption, CIE
- name, ZMQ, AWS 53, Elastic integration, PGP encryption, C operations

Located in app/Lib

Code that is to be reused across several layers

Important ones

- Dashboard Dashboard widget backend code
- EventReport Report generation
- Export MISP -> external format converter modules
- Tools List of generic helper libraries examples:
 - Attachment, JSON conversion, random generation, emailing, sync request generation
 - Kafka, ZMQ, AWS S3, Elastic integration, PGP encryption, CIDR operations

Located in app/Lib

MISP core development crash course

Distribution

8

2024-10

 algorithm for checking if a user has access to an attribute
 creator us owner organisation
 distribution levels and inheritance (events -> objects -> attributes)
 shorthand inherit level
 sharing groups (org fist, instance list)
 correlation distribution

algorithms for safe data fetching (fetchEvents())

- algorithm for checking if a user has access to an attribute
- creator vs owner organisation
- distribution levels and inheritance (events -> objects -> attributes)
- shorthand inherit level
- sharing groups (org list, instance list)
- correlation distribution
- algorithms for safe data fetching (fetchEvents(), fetchAttributes(),...)

TESTING YOUR CODE

MISP core development crash course

└─Testing your code

8

-10

2024

- funtional testing
- Github actions
- impact scope
- view code changes: only impacts request type bas
- controller code changes: Should only affect given action
 model code changes: can have impact on entire application
- lib changes: can have affect on the entire application
- Don't forget: queryACL, change querystring

funtional testing

Github actions

impact scope

- view code changes: only impacts request type based views
- controller code changes: Should only affect given action
- model code changes: can have impact on entire application
- lib changes: can have affect on the entire application
- Don't forget: queryACL, change querystring