Automated capability analysis in Wordpress plugins

Using static and dynamic analysis

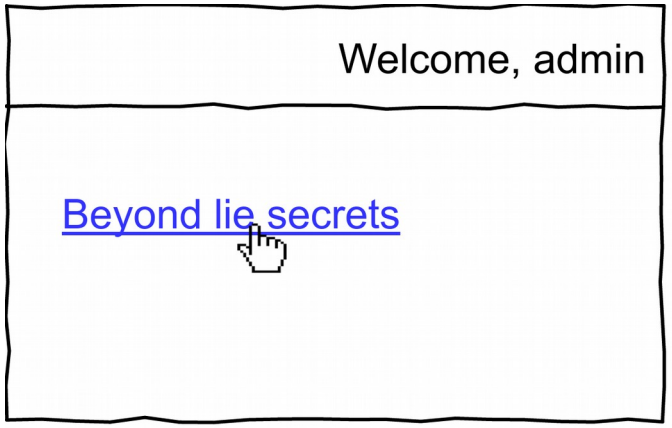
SNE RP2

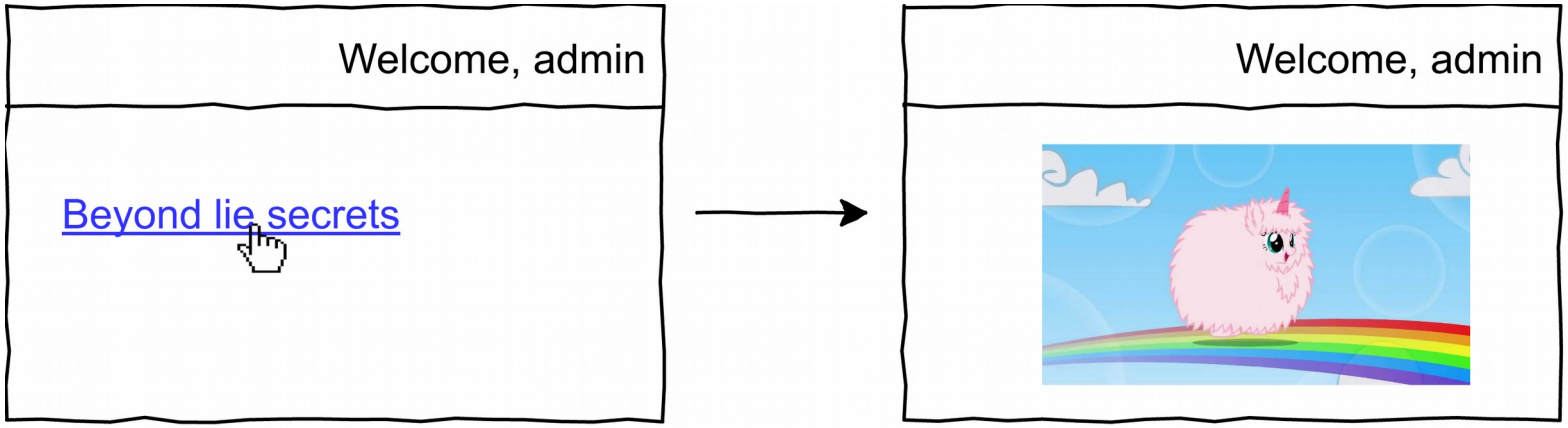
Frank Uijtewaal

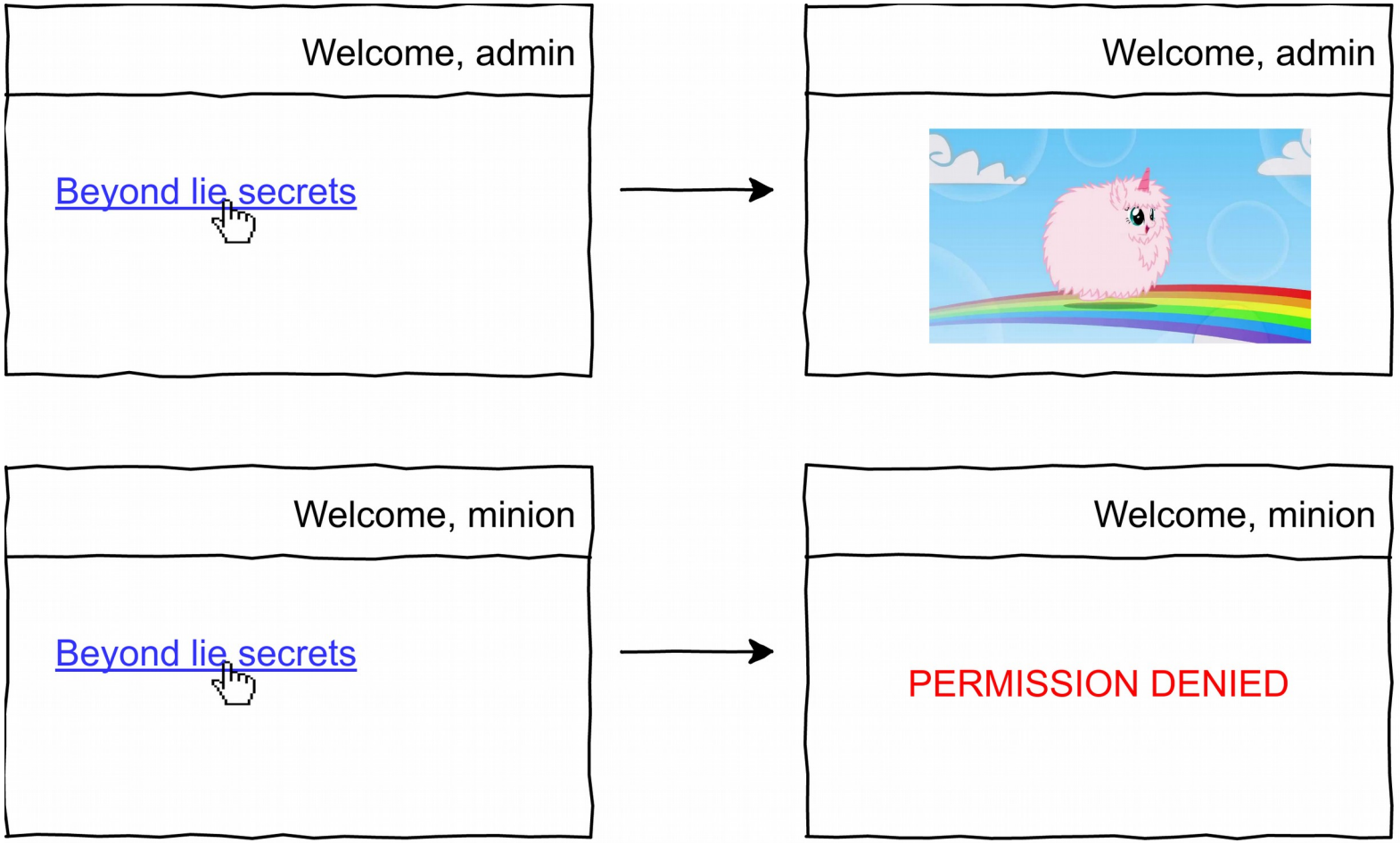
Collab: DongIT (dongit.nl)

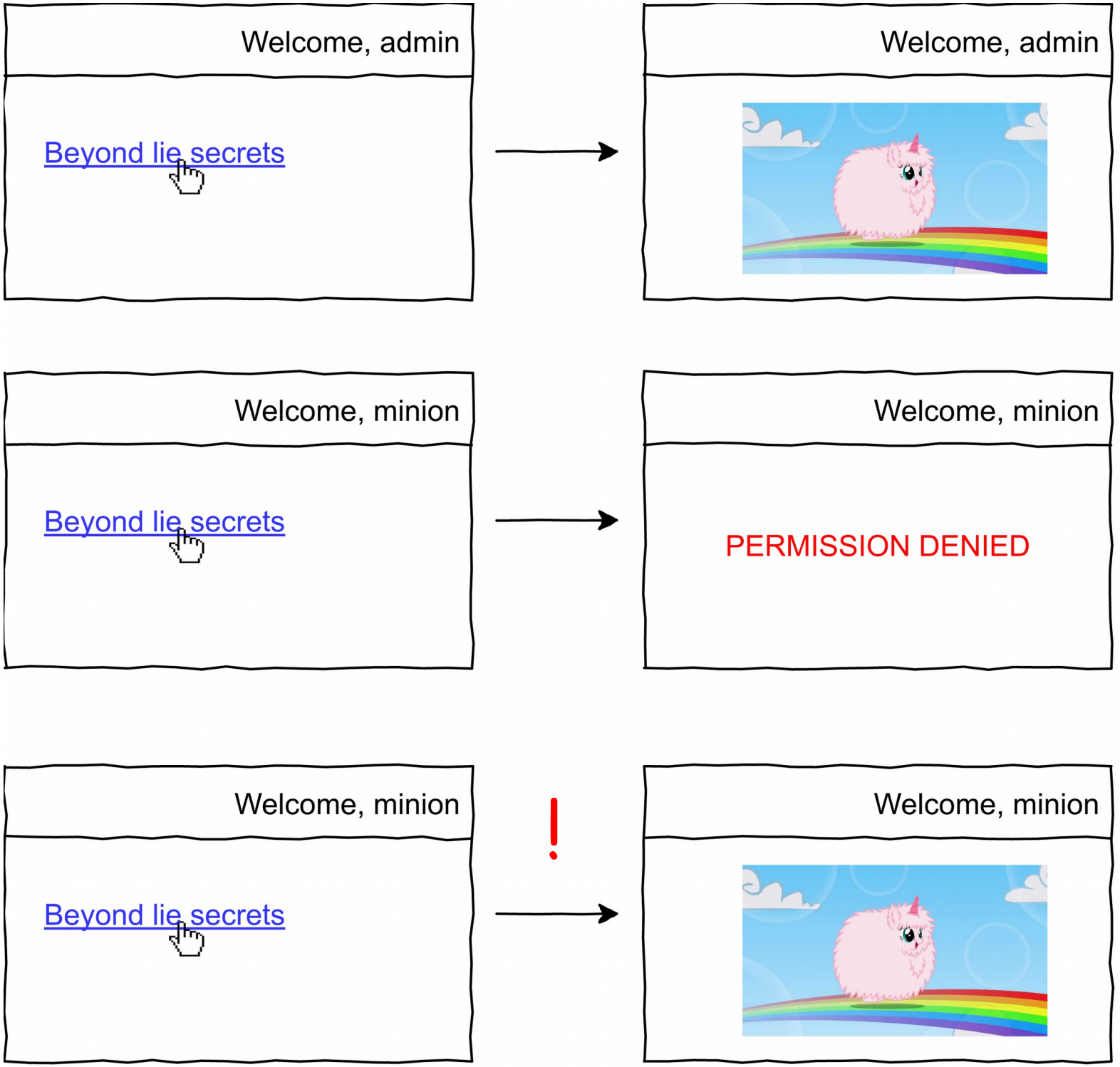
Initial project goal

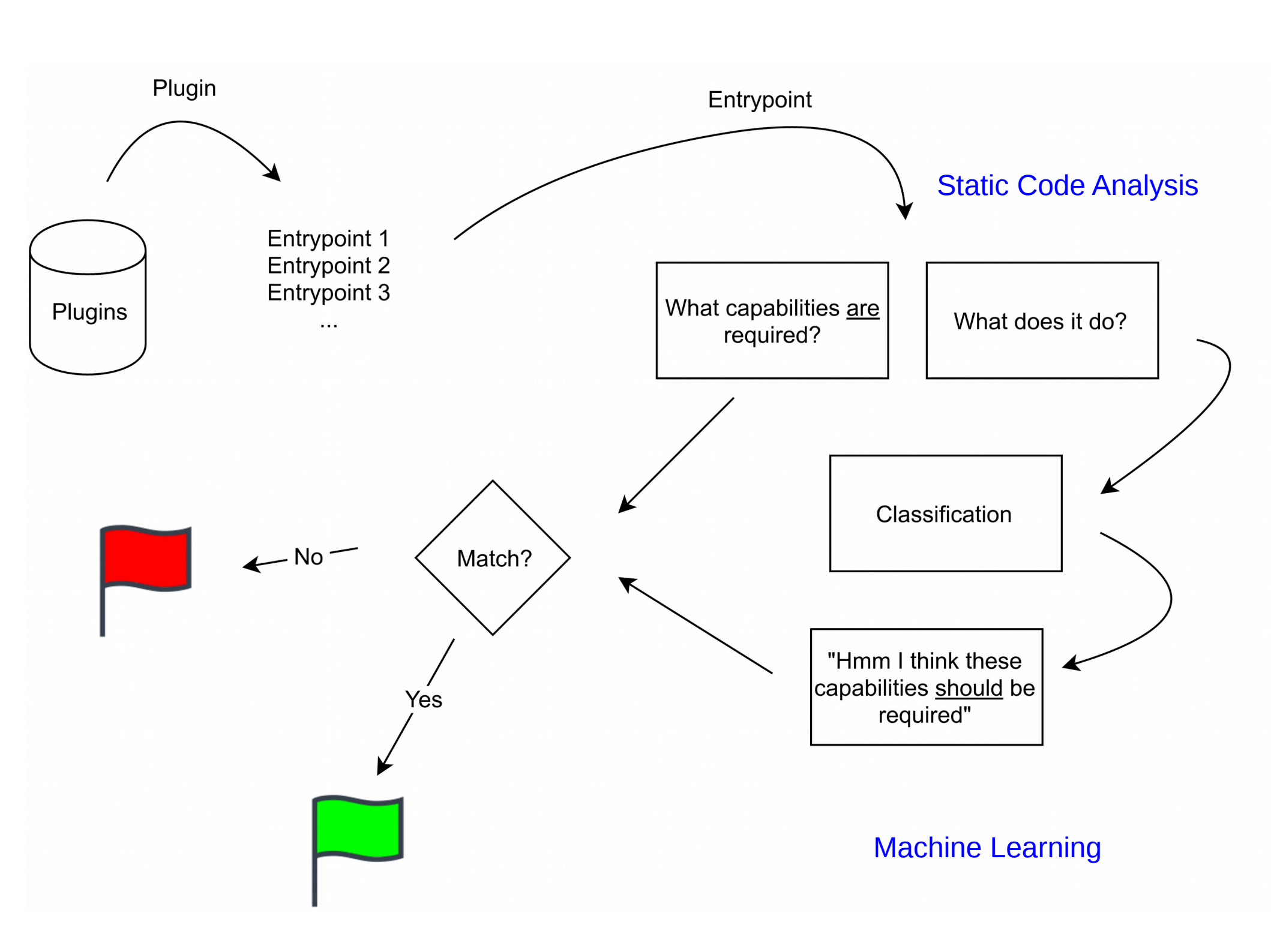
- Find a new and interesting type of security analysis for web applications
- Static code analysis
- WordPress
- Plugins
- A7 Missing Function Level Access Control

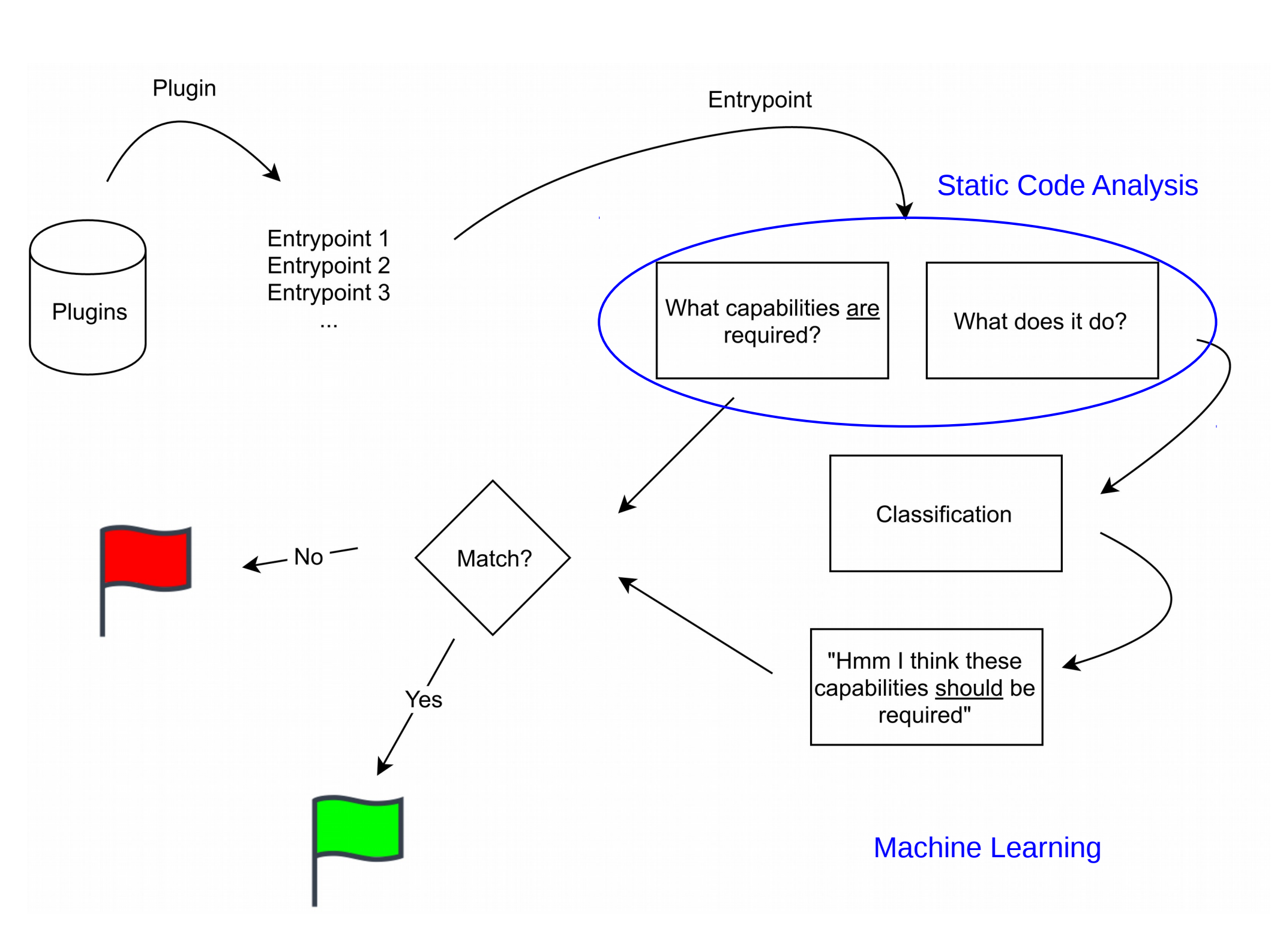

Welcome, admin
Beyond lie secrets

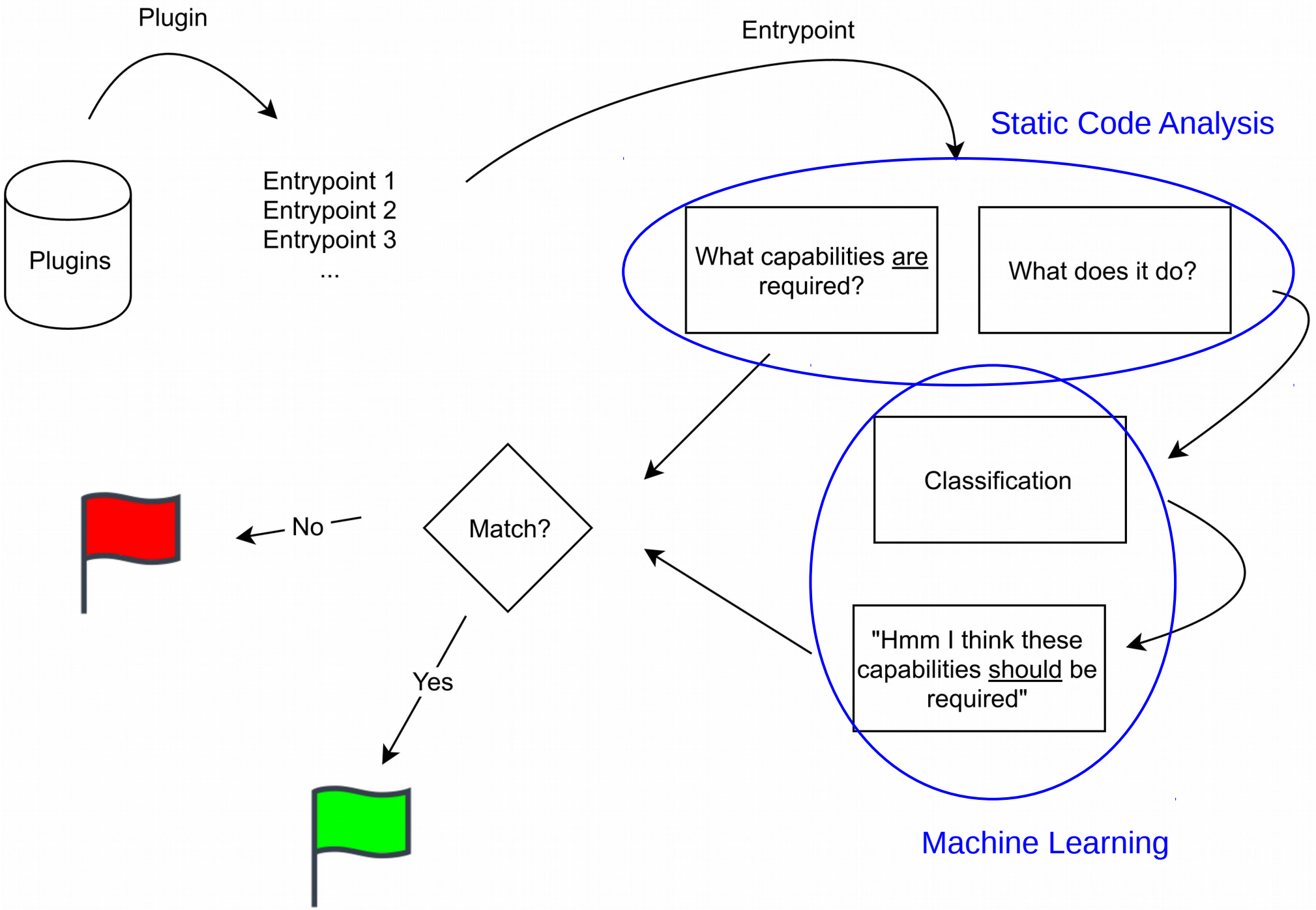

Welcome, admin
Beyond lie secrets
Welcome, admin


Welcome, admin
Beyond lie secrets
Welcome, admin
Welcome, minion
Beyond lie secrets
Welcome, minion
PERMISSION DENIED


Welcome, admin
Beyond lie secrets
Welcome, admin
Welcome, minion
Beyond lie secrets
Welcome, minion
PERMISSION DENIED
Welcome, minion
Beyond lie secrets
!
Welcome, minion


Plugin
Entrypoint
Static Code Analysis
Plugins
Entrypoint 1
Entrypoint 2
Entrypoint 3
...
What capabilities are required?
What does it do?
Classification
Match?
No
Yes
"Hmm I think these capabilities should be required"
Machine Learning


Plugin
Entrypoint
Static Code Analysis
Plugins
Entrypoint 1
Entrypoint 2
Entrypoint 3
...
What capabilities are required?
What does it do?
Classification
Match?
No
Yes
"Hmm I think these capabilities should be required"
Machine Learning


Plugin
Entrypoint
Static Code Analysis
Plugins
Entrypoint 1
Entrypoint 2
Entrypoint 3
...
What capabilities are required?
What does it do?
Classification
Match?
No
Yes
"Hmm I think these capabilities should be required"
Machine Learning

Static Code Analysis part

Turns out

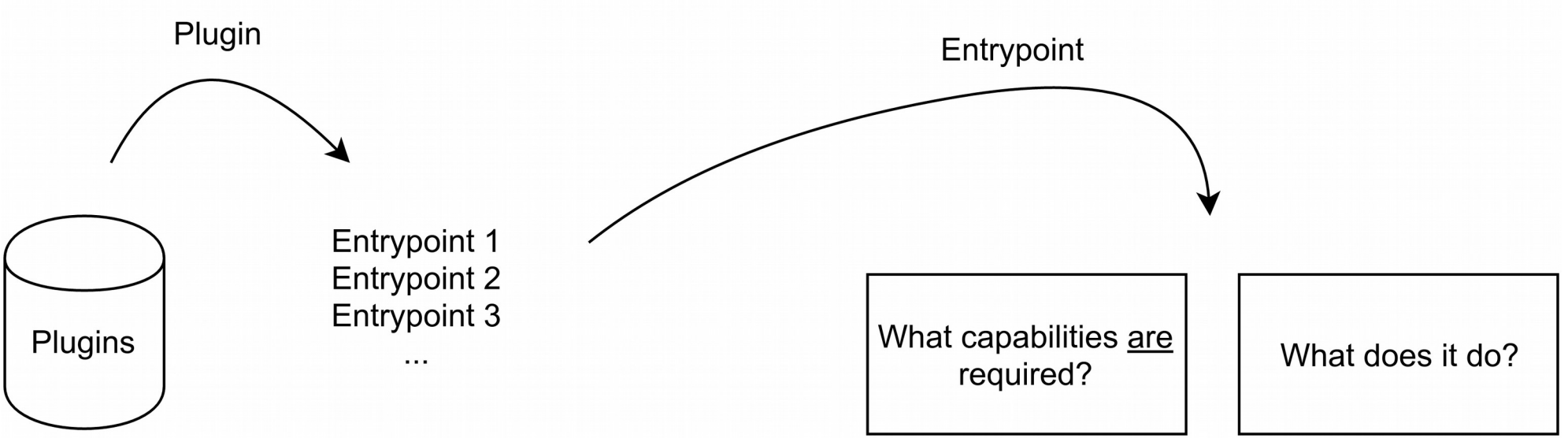
What capabilities are required?	What does it do?

Not so easy

No parsing & analysis tools available

Updated goal

- Static analyis
- Static & Dynamic analysis



Future work here

WordPress

- Capability checking
  - Functions *current_user_can*, *user_can*

```
if (!current_user_can('manage_options')) {
    wp_die(__("you don't have the clearance"));
}
```

- User roles
  - *Administrator*, *editor*, *author*, etc.
- URL handlers: events and callbacks
  - Functions *add_action*, *add_filter*

```
( string $tag, callable $function_to_add, ....)
```

Static analysis summary

- Simple, conformative code quite doable
- Lots of variation means stronger tools required
- Modeling and analysis tools a must to do this research
- *No such tools freely available*

Static analysis example

- Extract required capabilities

```
if (!current_user_can('manage_options')) {
    wp_die(__("you don't have the clearance"));
}
```

- Done!
  - Req capability == 'manage_options'
- Easy

Not always straightforward

```
if ( is_wp_error( $error = $this->validate_call( $blog_id, $this->needed_capabilities ) ) ) {
	return $error;
}
```

```
$must_pass = ( isset( $capability['must_pass'] ) && is_int( $capability['must_pass'] ) ?
$capability['must_pass'] : count( $capabilities ) );

$failed = array(); // store the failed capabilities
$passed = 0; //

foreach ( $capabilities as $cap ) {
	if ( current_user_can( $cap ) ) {
		$passed ++;
	} else {
		$failed[] = $cap;
	}
}
```

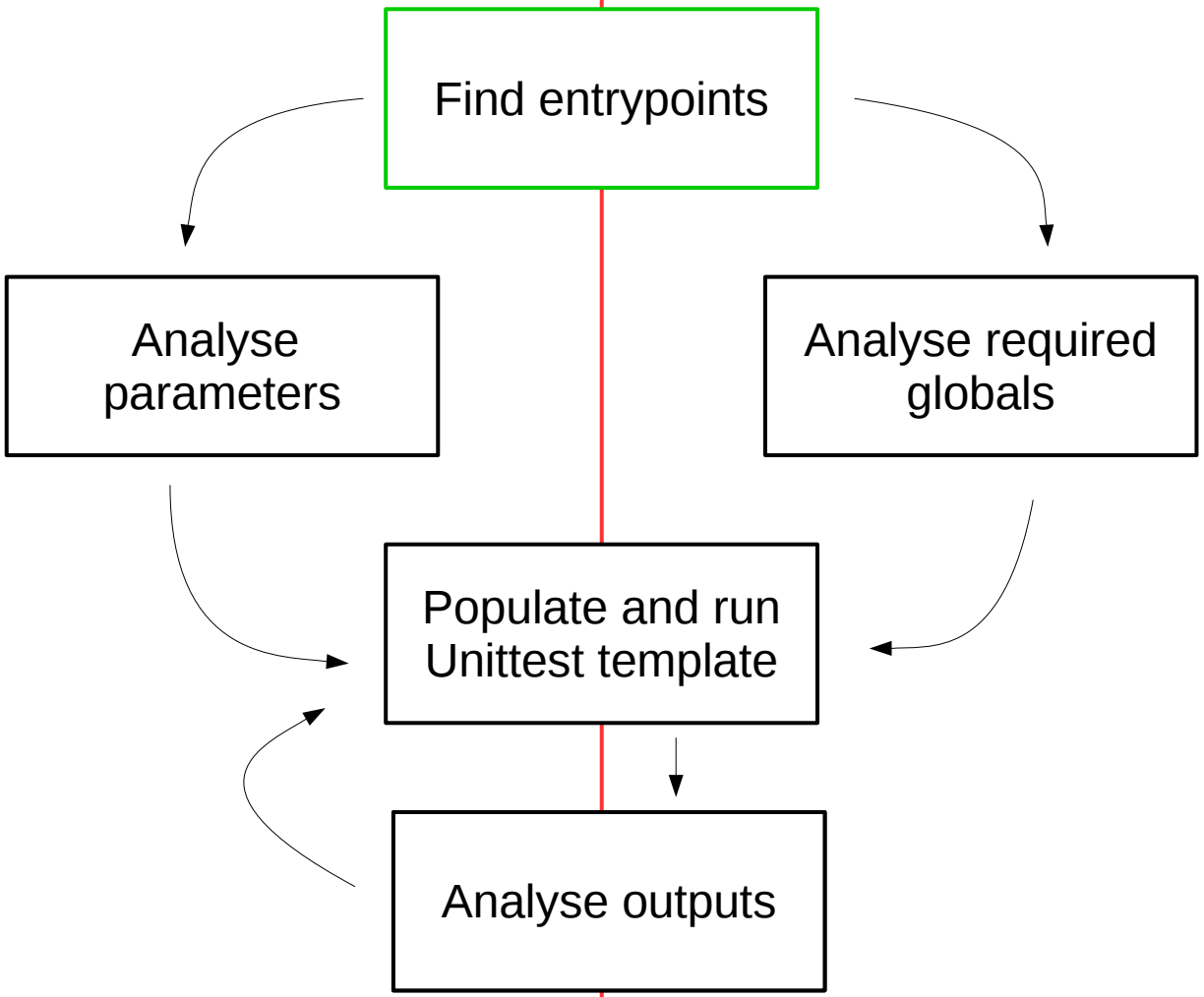
Static ... += dynamic

- No fancy tools
- Given current options, static = no go
- No time to create all tools necessary
- How far do we get without those “tools”, then?
- But... PHP interpreter?

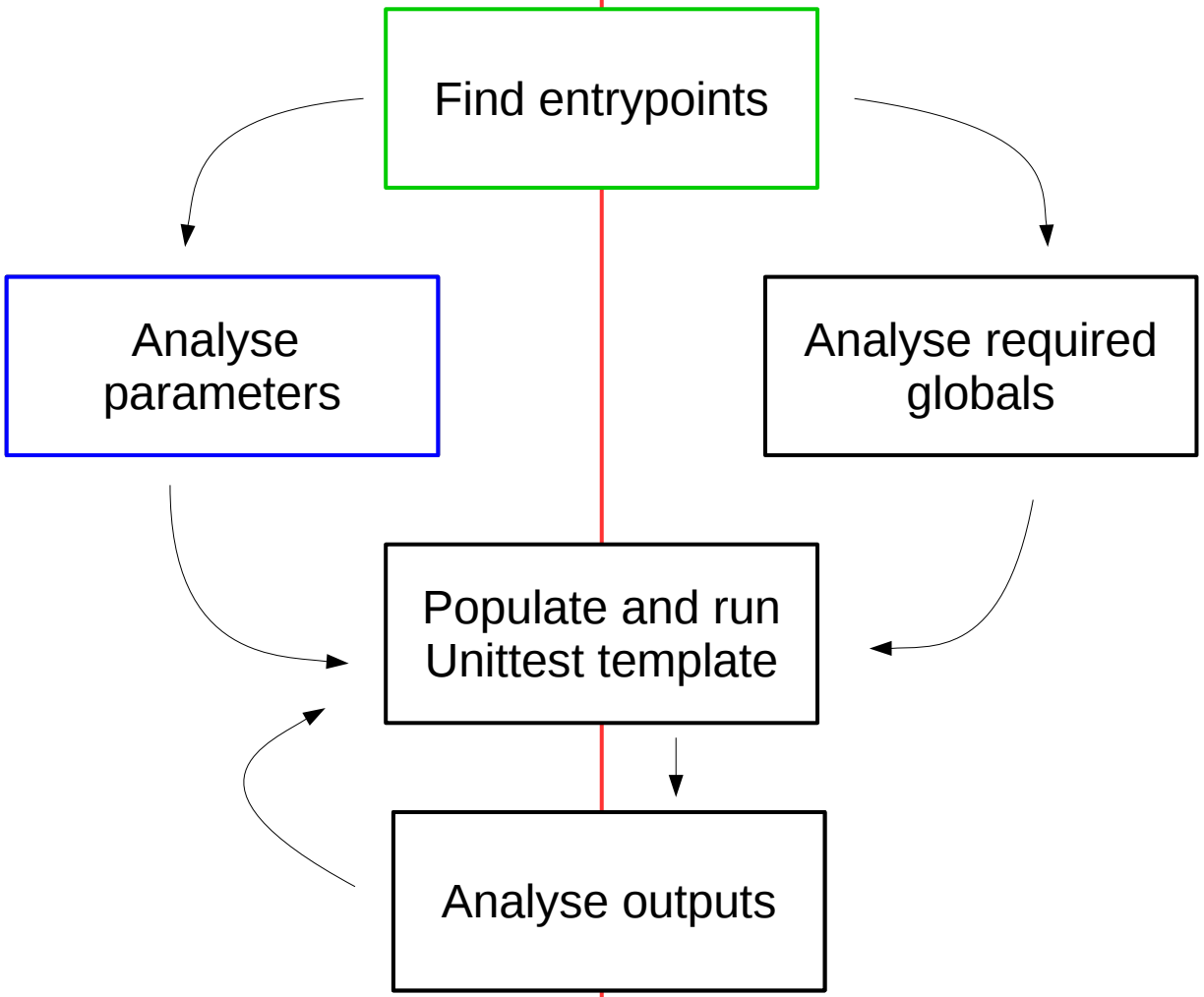
Static & dynamic, principles

- Gather as much data as possible statically
- Run URL handlers:
  - Directly: my_url_handler($arg1, $arg2, $arg3);
  - Not through webserver as is normally the case
  - Use gathered data to make a nice baby room
- Record as much data from run
  - Use data to repeat and finetune environment
  - Use data to analyse code paths & capabilities

Static & Dynamic approach



Static & Dynamic approach



Analyse parameters

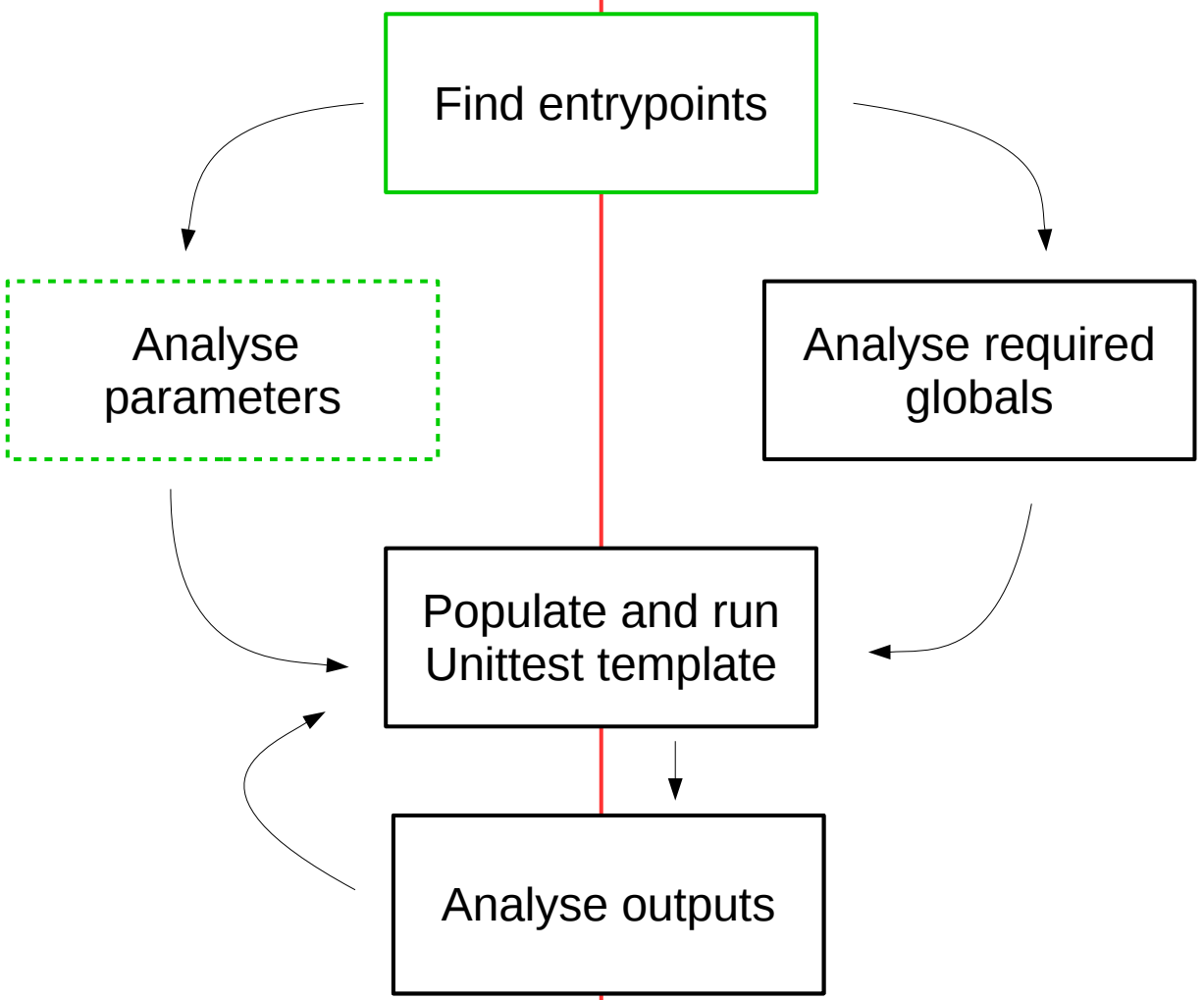
- Known hooks
- Custom hooks
- Trace parameters to known functions

```php
public function a_func($post_id) {}
```

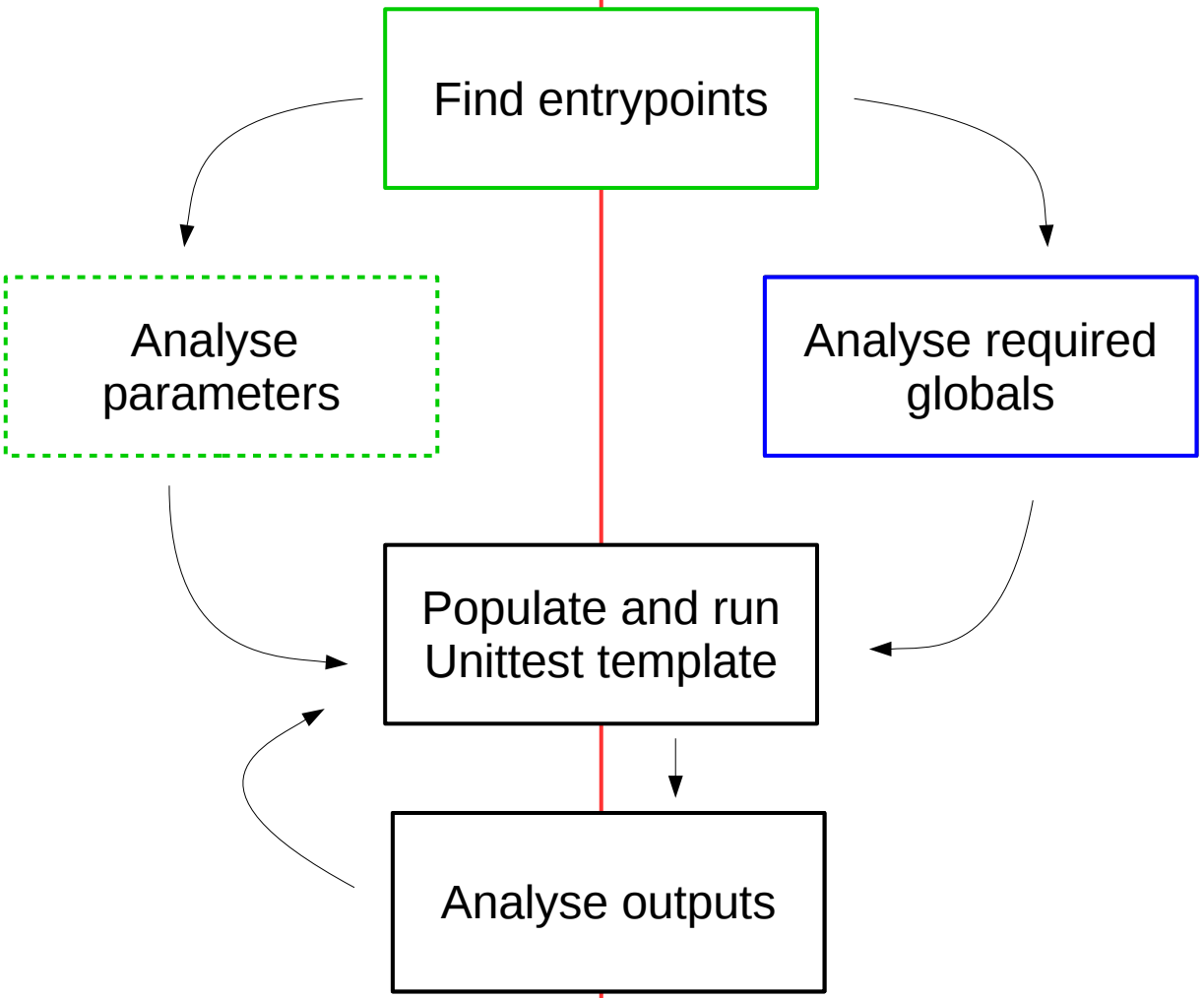
- `add_ping ( int $post_id, string $uri )`
- `add_post_meta ( int $post_id, string $meta_key, mixed $meta_value, bool $unique = false )`
- `check_and_publish_future_post ( int|WP_Post $post_id )`

- Constraint analysis

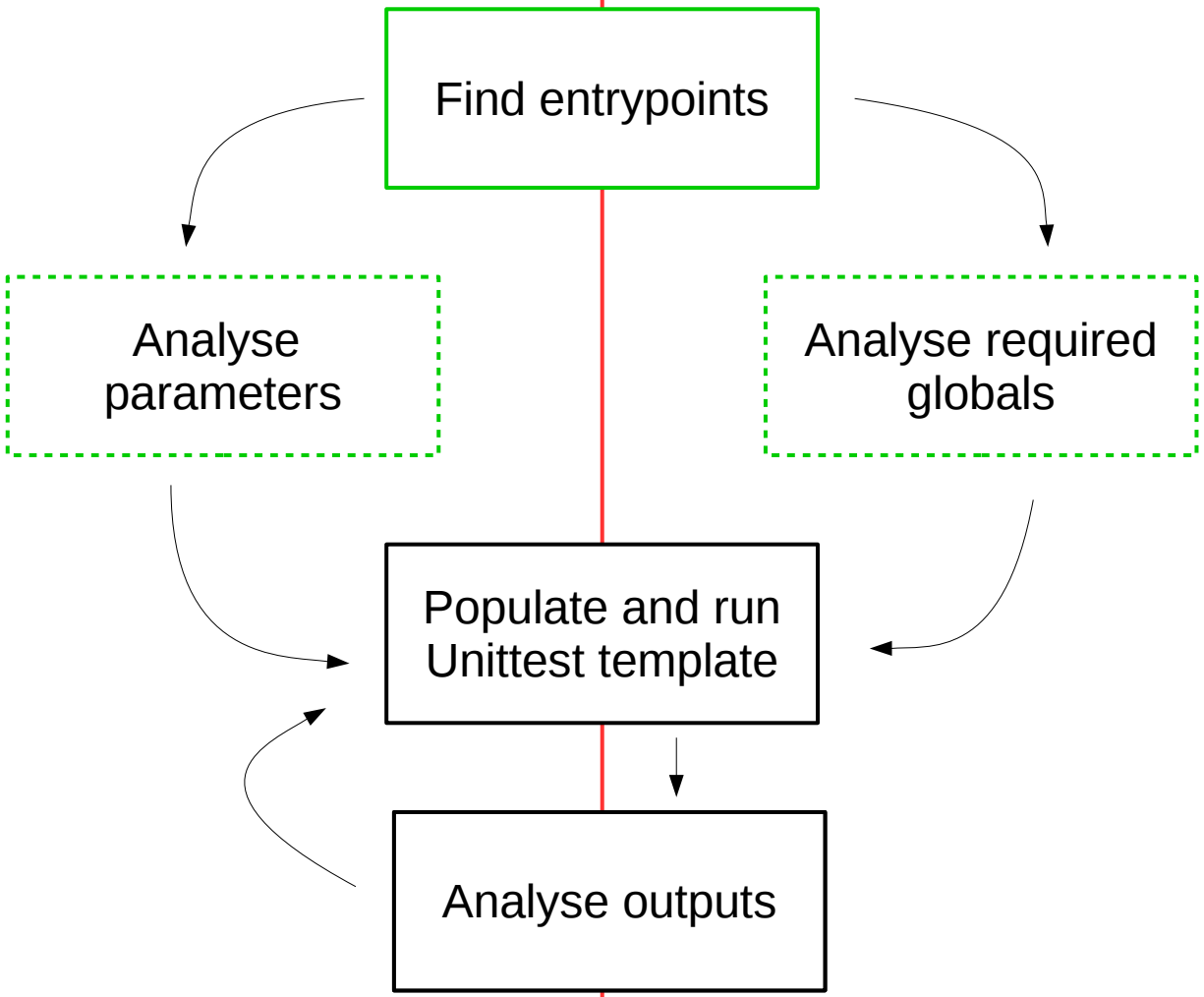
Static & Dynamic approach



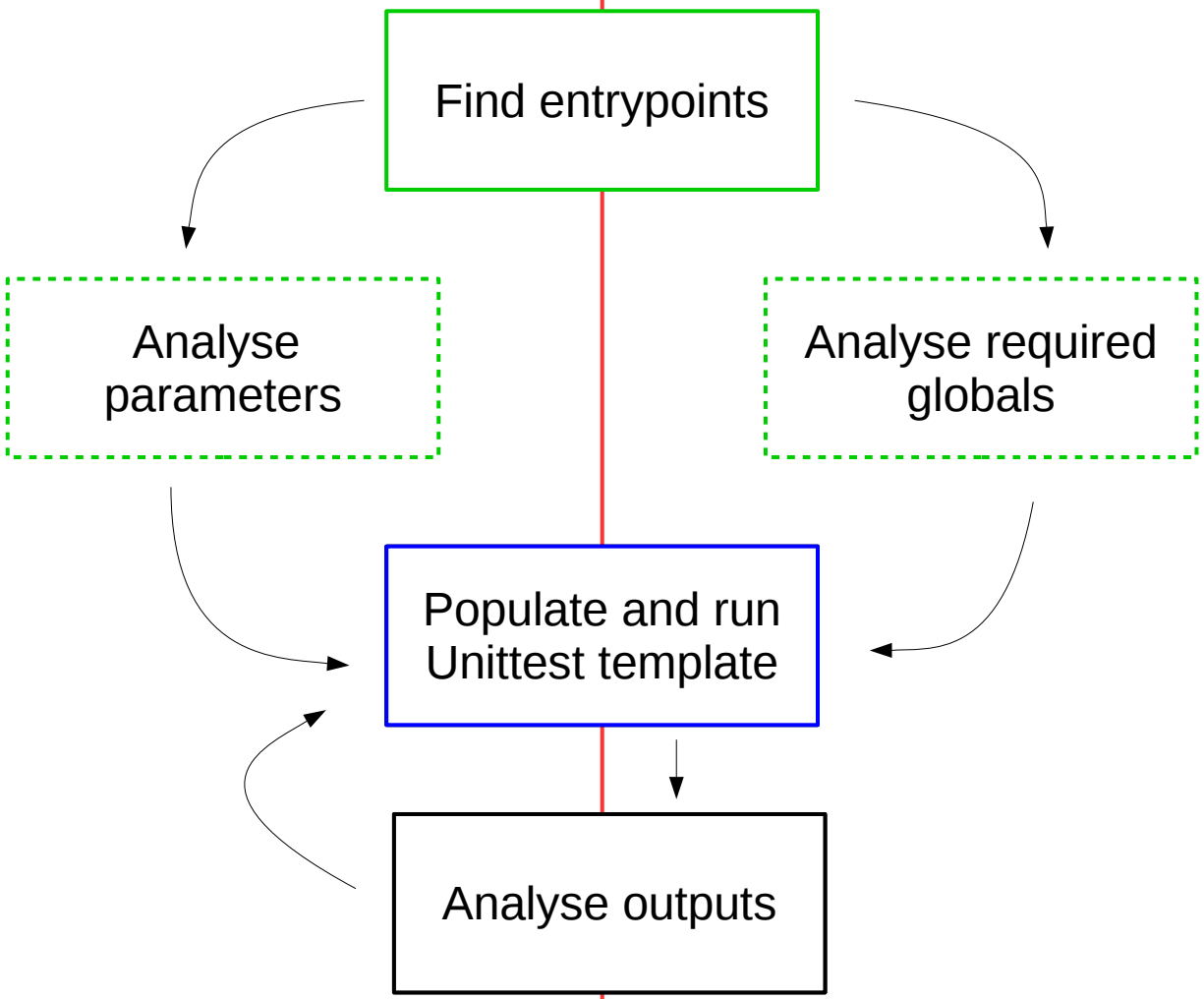
Static & Dynamic approach



Static & Dynamic approach



Static & Dynamic approach

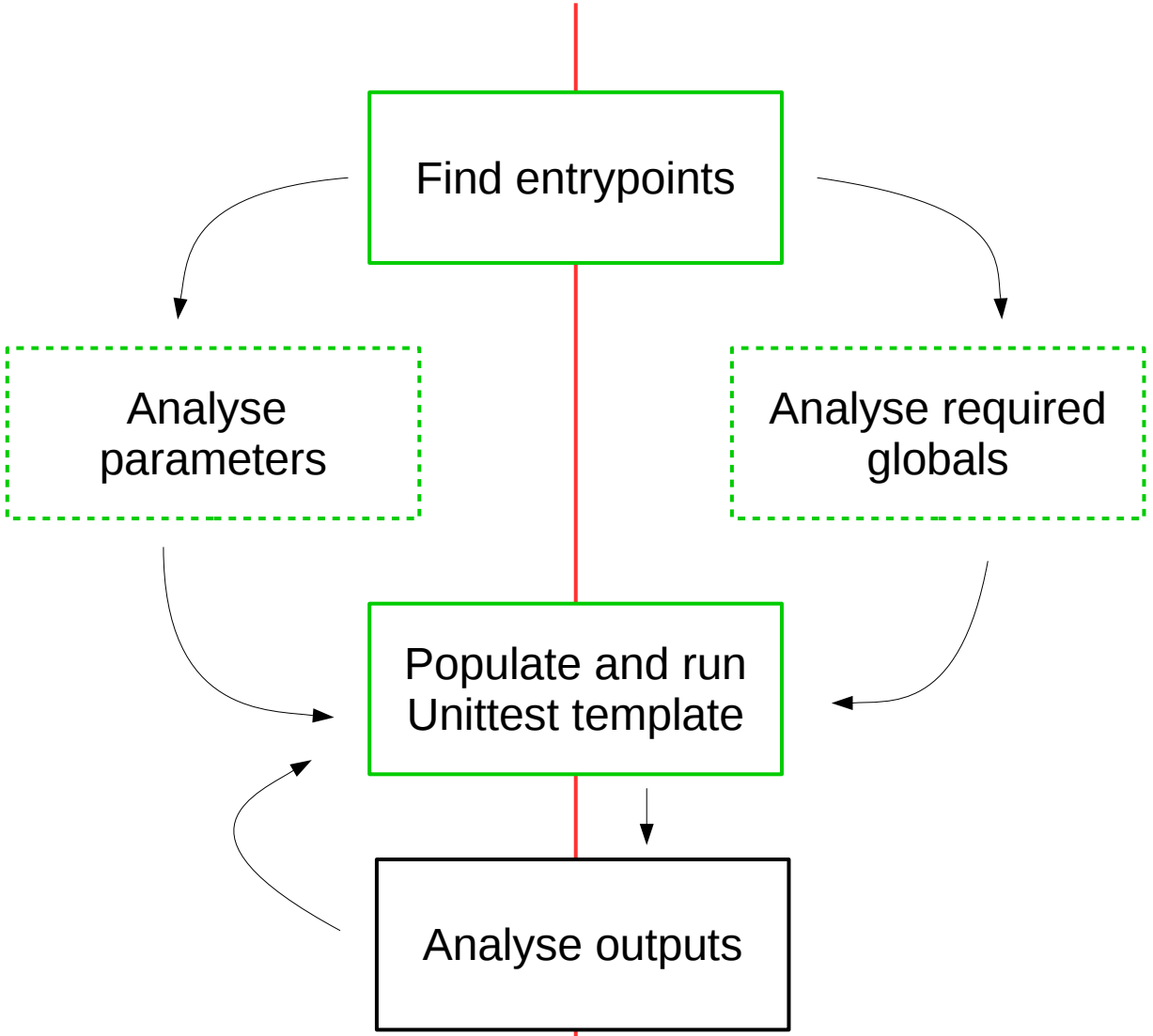


Unittest template

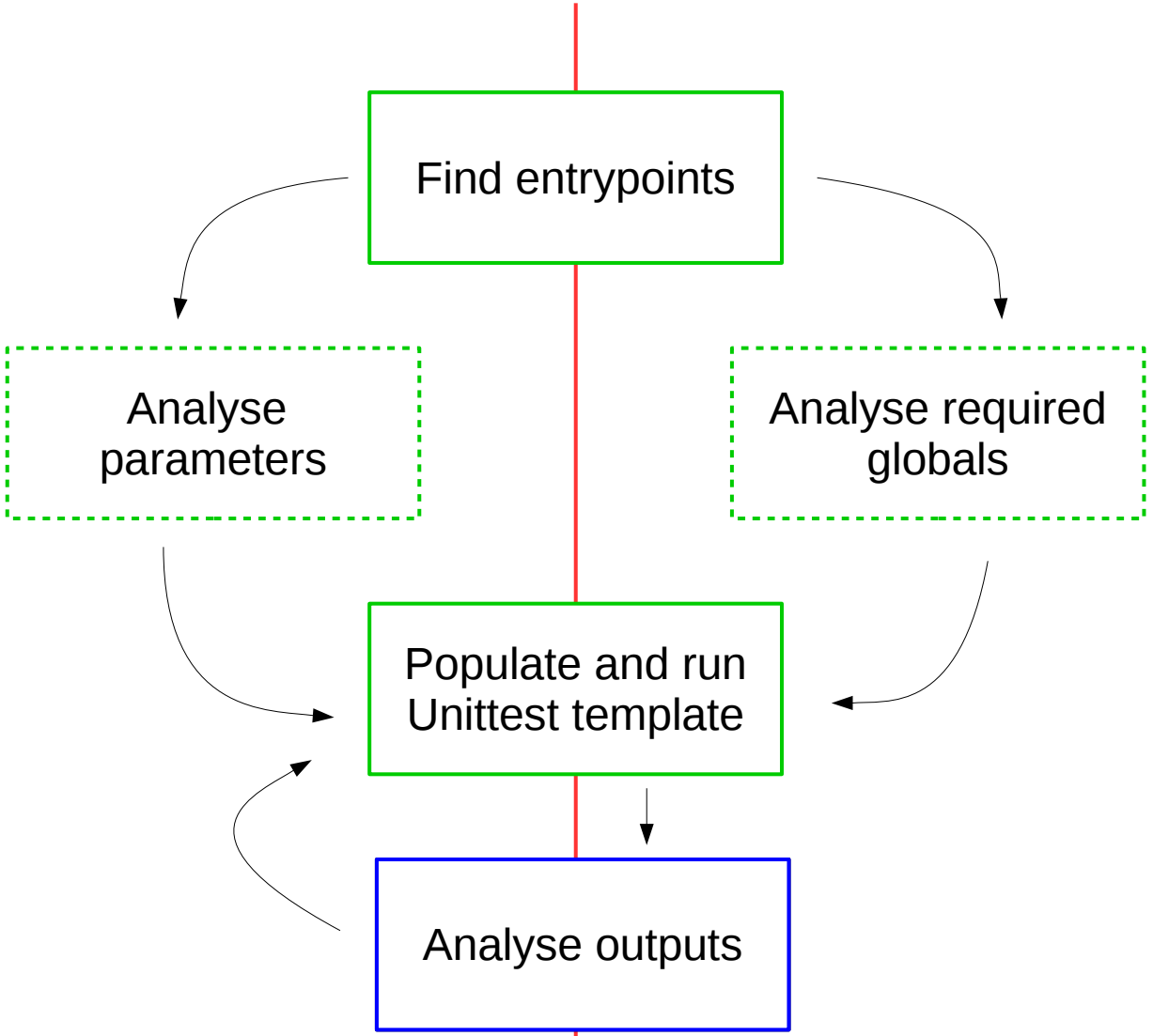
- Run callback:
  - WordPress has support for PHPunit
  - So, unittests
- WP_UnitTestCase
  - Sets up environment
  - Creates temporary tables
  - Offers nice functionality
- Xdebug for call traces and code coverage

```php
class TemplateTest extends WP_Ajax_UnitTestCase {
    private $roles = array();
    public function setUp() {
        parent::setUp();
        $this->role_names = array('subscriber', 'contributor',
            'author', 'editor', 'administrator');
    }

    public function test_do_handling()    {
        foreach ($this->role_names as $role_name) {
            // 1 {Setup per role}
            $this->_setRole($role_name);

            $filename = __DIR__ . "/traces/trace-{$role_name}";

            xdebug_start_code_coverage(XDEBUG_CC_UNUSED);
            xdebug_start_trace($filename);

            // 2 {Call to URL handler}

            xdebug_stop_trace();
            $coverage = xdebug_get_code_coverage();
            xdebug_stop_code_coverage();
            file_put_contents($filename . '.cov',
                    coverage_to_json($coverage));
            // 3 {Teardown for per-role setup}
        }
    }
}
```

Static & Dynamic approach



Static & Dynamic approach



Output 1: Traces

- Headers

Record type	1	6	10	11	12 - ...
Entry	level	function name	line number	no. of parameters	parameters
Exit	level	empty			
Return	level	return value	empty		

- Example output

0	19	wpdb->_real_escape	1209	1	$string = 'setting_b'
0	20	mysqli_real_escape_string	1127	2	class mysqli { ... }, 'setting_b'
1	20				
R	20	'setting_b'			
1	19				
R	19	'setting_b'			

```php
function plugin_settings_page() {
    if (!current_user_can('manage_options')) {
        wp_die(__("you don't have the clearance"));
    }
    include(sprintf("%s/settings.php", dirname(__FILE__)));
}
```

```
13 - E: MyPlugin->plugin_settings_page ([])
    14 - E: current_user_can (["$capability = 'manage_options'"])
    14 - R: FALSE
    14 - E: __ (["$text = 'you don\\'t have the clearance'", '$domain
           = ???'])
        15 - E: translate (["$text = 'you don\\'t have the clearance'",
               "$domain = 'default'"])
            16 - E: apply_filters (["$tag = 'gettext'", "$value = 'you
                  don\\'t have the clearance'", "'you don\\'t have the
                  clearance'", "'default'"])
            16 - R: 'you don\'t have the clearance'
        15 - R: 'you don\'t have the clearance'
    14 - R: 'you don\'t have the clearance'
    14 - E: wp_die (["$message = 'you don\\'t have the clearance'", '$title
= ???', '$args = ???'])
```

Output 2: Code coverage

```
18      public function __construct() {
19          // register actions
20          add_action('admin_init', array(&$this, 'admin_init'));
21          add_action('admin_menu', array(&$this, 'add_menu'));
22      }
```

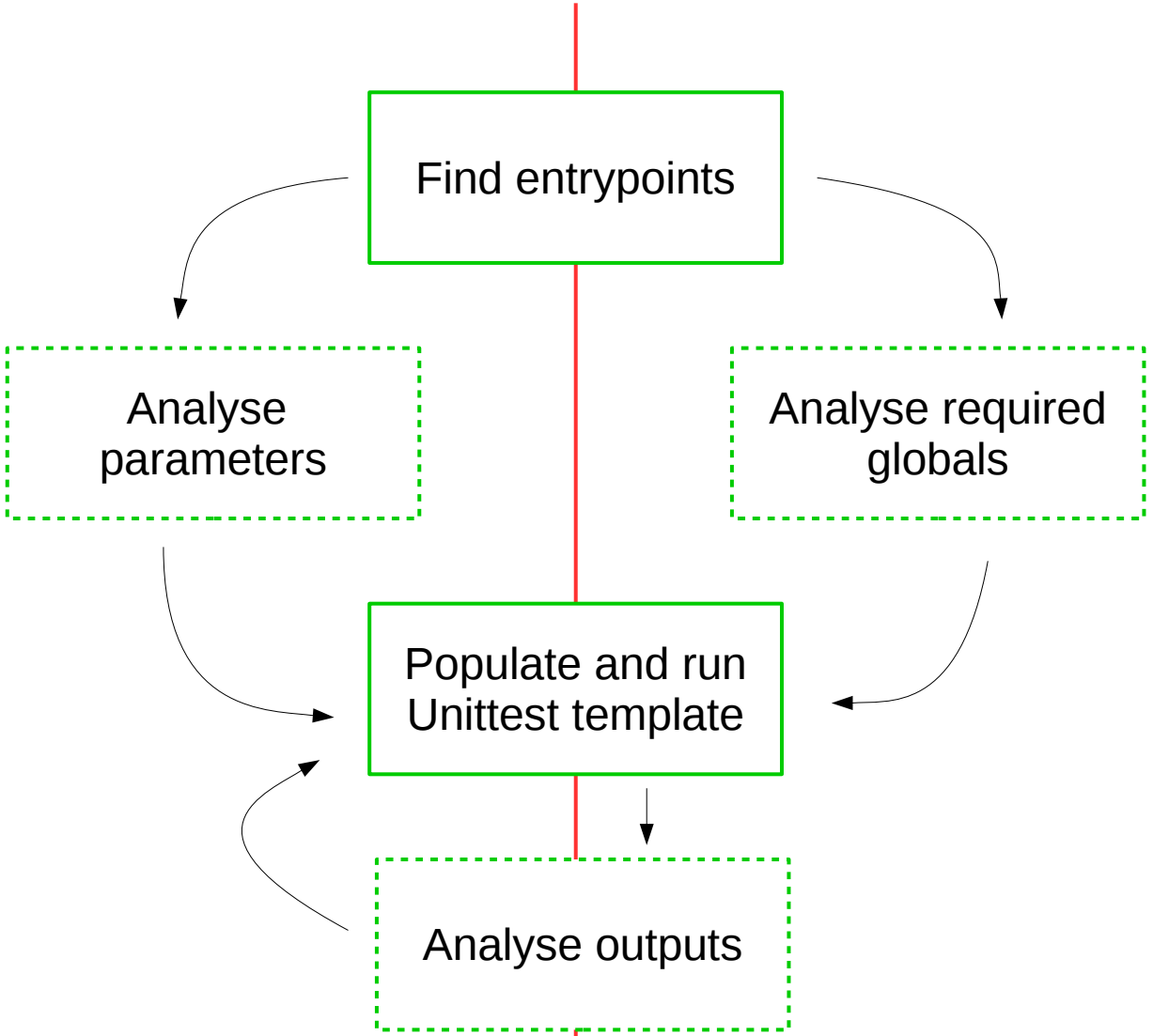
```
52      function plugin_settings_page() {
53          if (!current_user_can('manage_options')) {
54              wp_die(__("you don't have the clearance"));
55          }
56
57          include(sprintf("%s/templates/settings.php", dirname(__FILE__)));
58      }
```

Subscriber coverage	admin coverage	Admin noncoverage
20	20	54
21	21	
22	22	
53	53	
54	57	
	58	

Did the run/test succeed?

- Meaning:
  - Did we predict the correct arguments and globals?
- Metrics:
  - Full Code coverage?
  - Joined set should hit all return paths?
- If deemed not:
  - Rerun with different parameters/globals
  - Test's output may help here

Static & Dynamic approach



Conclusions

- Lack of good parsing & analysis tools inhibits research.
  - Exploring code, extracting features difficult
  - Testing hypotheses not possible on large scale

- If they *were* available:
  - Static & dynamic analysis may be quite fruitful here

Future work

- Design & build a good working modeling tool
  - Modular and extendible
  - Allows for queries to be done
    - What (symbolic) value does this variable have here?
    - What should x be to evaluate this to true? (constraint solving)
    - What is this variable used for?
  - Needs to build a very good representation of the code
- Next steps are Modeling & Predictions
  - WordPress knowledge needed for classification