

嘉宾介绍

董光利

开源软件漏洞挖掘实践

议题介绍

不论作为SDL实施人员做企业内部的安全建设，还是作为安全研究员挖掘开源软件漏洞，或者作为研发人员做"code review"时考虑安全性，都需要结合业务架构评估代码中的安全风险。我想分享我对企业项目和开源项目审计的认识，以及我自己做代码审计的方法论。



火线云安全实验室
高级研究员

01

自我介绍



代码安全相关的经历

- 白帽子经历：在阿里先知平台提交过共61个有效漏洞（8个高危、24个中危），包括Discuz、Destoon、Finecms、phpcmsv9等；
* 非通用类型漏洞：给“华为、阿里、百度、360、字节”等国内的大厂提交过高危严重漏洞
- 百度云经历：负责“计算、网络等业务线的安全评估与漏洞运营”；共评估30+个业务
- 火线安全：云原生相关的安全研究

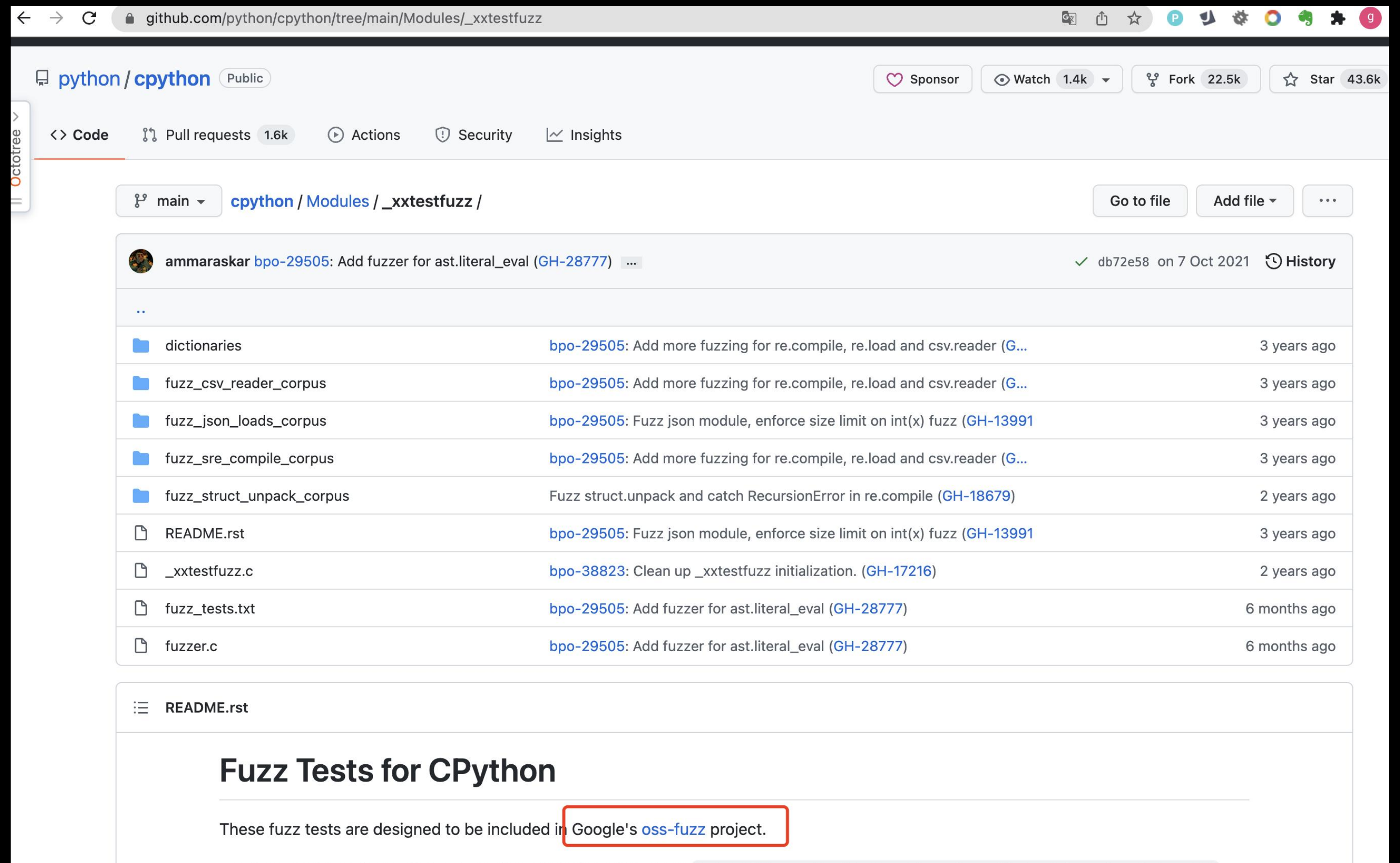
02

开源软件安全评估



很多开源软件已经集成了fuzz

- [CPython](#)
- [v8](#)
- [Go](#)
- [PHP](#)



开源软件可以借助CodeQL检测漏洞

The image displays two overlapping screenshots of the GitHub CodeQL interface. The background screenshot shows the 'Code scanning' page for the repository 'leveryd / apisix-dashboard'. It features a sidebar with navigation links: Overview, Security policy, Security advisories, Dependabot alerts, and Code scanning alerts (highlighted with a red box and a '4' badge). The main content area, titled 'Code scanning', shows a table with columns for Latest scan, Branch, Workflow, Lines scanned, Duration, and Result. It lists four high-severity alerts, all labeled 'Log entries created from user input' and 'Insecure randomness', detected 23 days ago by CodeQL. A 'ProTip!' at the bottom suggests running CodeQL locally. The foreground screenshot shows the workflow file '.github/workflows/codeql-analysis.yml' for the repository 'apache / apisix-dashboard'. It displays the workflow configuration for the 'master' branch, including a commit by 'juzhiyuan' to 'chore: reduce CI time (#1762)'. The workflow file content is visible, showing instructions for setting up the CodeQL environment and running the analysis.

github.com/leveryd/apisix-dashboard/security/code-scanning

leveryd / apisix-dashboard Public
forked from apache/apisix-dashboard

Code Pull requests Actions Projects Wiki Security 4 Insights Settings

Overview
Security policy
Security advisories
Dependabot alerts
Code scanning alerts 4

Code scanning

Latest scan	Branch	Workflow	Lines scanned	Duration	Result
4 days ago	master	CodeQL	7.75k / 34k	2m 47s	3 alerts

is:open branch:master

4 Open 0 Closed

- ☐ Log entries created from user input **High**
api/.../filter/logging.go:52 • Detected 23 days ago by CodeQL
- ☐ Log entries created from user input **High**
api/.../filter/logging.go:49 • Detected 23 days ago by CodeQL
- ☐ Log entries created from user input **High**
api/.../filter/logging.go:45 • Detected 23 days ago by CodeQL
- ☐ Insecure randomness **High**
web/.../CertificateUploader/index.tsx:44 • Detected 23 days ago by CodeQL

ProTip! You can run CodeQL locally from the command line

github.com/apache/apisix-dashboard/blob/master/.github/workflows/codeql-analysis.yml

Search or jump to... Pull requests Issues Marketplace Explore

apache / apisix-dashboard Public

Code Issues 102 Pull requests 28 Actions Projects 3 Wiki Security Insights

master apisix-dashboard / .github / workflows / codeql-analysis.yml

juzhiyuan chore: reduce CI time (#1762) ✓ Latest

2 contributors

72 lines (62 sloc) | 2.38 KB

```
1 # For most projects, this workflow file will not need changing; you simply need
2 # to commit it to your repository.
3 #
4 # You may wish to alter this file to override the set of languages analyzed,
5 # or to provide custom queries or build logic.
6 #
7 # ***** NOTE *****
8 # We have attempted to detect the languages in your repository. Please check
9 # the `language` matrix defined below to confirm you have the correct set of
10 # supported CodeQL languages.
11 #
12 name: "CodeQL"
13
14 on:
15   push:
16     branches:
17       - master
```


03

企业代码审计



"企业代码审计"不止"看代码和挖漏洞"

目标:

- "防守"而不是"为了挖漏洞"

理想:

- 能在服务上线前修复漏洞
- 能够覆盖所有对外的线上业务

工作内容变化:

- 多了"跟进修复"
- 少了"漏洞分析"

举个例子:

- sdi人员需要分析fastjson漏洞吗?

"企业"和"开源"代码审计区别很多

威胁模型

- 假设攻击者没有拿到源码
- 假设内网系统攻击者访问不到

困难

- 文档不完善
- 业务复杂
- 基本上没法动态调试

其中 /index.php/Interface/report/gameLog 接口使用硬编码 key, 校验 token 来做身份身份认证, 如下图:

```
<?php
class Report extends MY_Controller
{
    public function gameLog() {
        $returnData = array(
            'errno' => 0,
            'errmsg' => '操作成功',
            'data' => array(),
        );
        try {
            $stime = microtime( get_as_float: true );
            $key = '271716eb5[REDACTED]';
            $post_data = $this->param->getParam('content', array());
            $cmd = $this->param->getParam('cmd', array('method' => 'string', 'emptyErrorCode' => 3, 'i
            $sign = $this->param->getParam('sign', array('method' => 'string', 'emptyErrorCode' => 1,
            $operatorFlag = $this->param->getParam('pkey', array('method' => 'string', 'emptyErrorCode

            $localSign = "{$post_data}{$key}";
            log_message( level: 'debug', message: "加密前localSign: {$localSign}" );
            $localSign = md5($localSign);
            log_message( level: 'debug', message: "加密后localSign: {$localSign}" );

            if($sign <> $localSign) {
                throw new Exception( message: '校验签名错误', code: 401);
            }

            $content = json_decode($post_data, assoc: true);
            if($content === false) {
                throw new Exception( message: '数据格式错误' );
            }
        }
    }
}
```

04

方法论和案例



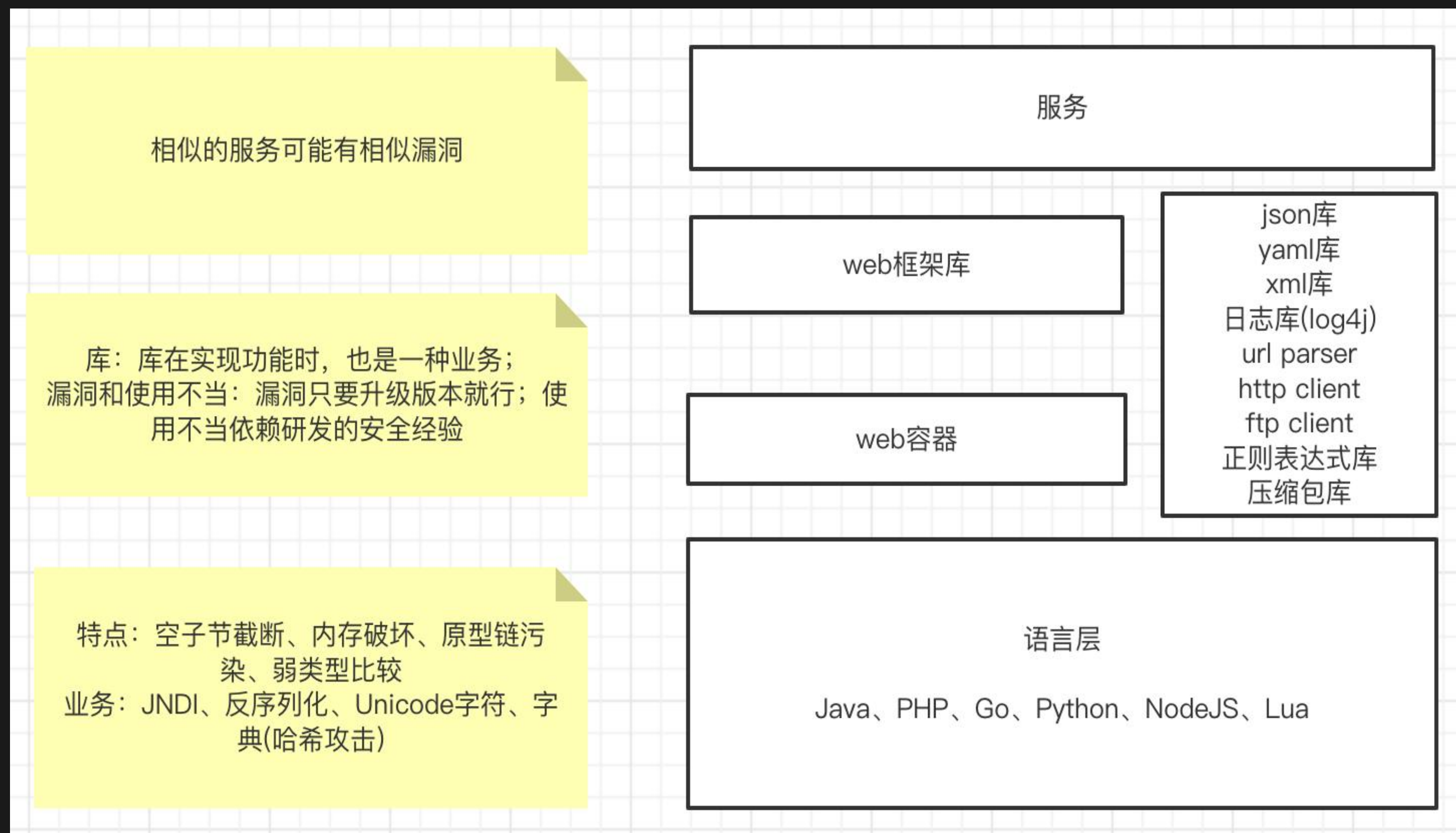
(针对web应用安全审计时)我们关注哪些目标?

经验:

- 先假设底层实现正确
- 相似的业务会存在相似的漏洞
- 漏洞常出现在对底层的误用

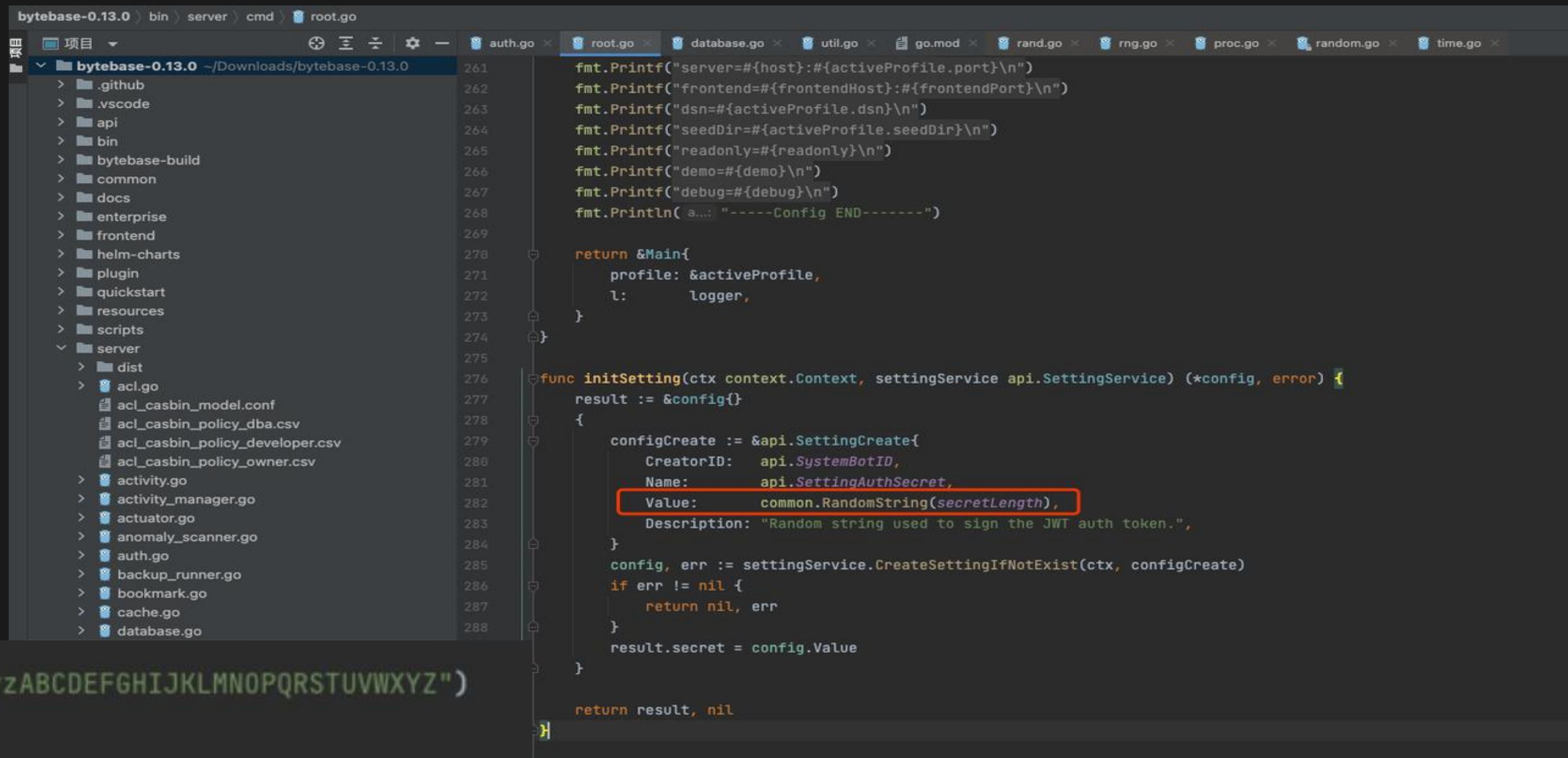
怎么审计:

- 审计目标自上而下
- 审计前学习同类业务的历史漏洞
- 审计时以"黑盒视角"找到"风险业务"



案例-bytebase身份认证伪造

- "身份认证业务"依赖jwt
- "jwt业务"依赖"随机数生成"
- "随机数生成业务"可能会不安全



The screenshot shows a Go IDE with the bytebase-0.13.0 project open. The left sidebar displays the project structure, including folders like .github, .vscode, api, bin, bytebase-build, common, docs, enterprise, frontend, helm-charts, plugin, quickstart, resources, scripts, server, and dist. The main editor shows the api.go file, specifically the initSetting function. The function is responsible for initializing settings for a given context. It creates a config struct and calls the CreateSettingIfNotExist method of the settingService. The Value field of the config struct is set to common.RandomString(secretLength), which is highlighted with a red box. The function also prints out configuration details and returns the result.

```
261 fmt.Printf("server=#{host}:{activeProfile.port}\n")
262 fmt.Printf("frontend=#{frontendHost}:{frontendPort}\n")
263 fmt.Printf("dsn=#{activeProfile.dsn}\n")
264 fmt.Printf("seedDir=#{activeProfile.seedDir}\n")
265 fmt.Printf("readonly=#{readonly}\n")
266 fmt.Printf("demo=#{demo}\n")
267 fmt.Printf("debug=#{debug}\n")
268 fmt.Println(a...: "-----Config END-----")
269
270 return &Main{
271     profile: &activeProfile,
272     l:       logger,
273 }
274
275
276 func initSetting(ctx context.Context, settingService api.SettingService) (*config, error) {
277     result := &config{}
278     {
279         configCreate := &api.SettingCreate{
280             CreatorID: api.SystemBotID,
281             Name:       api.SettingAuthSecret,
282             Value:      common.RandomString(secretLength),
283             Description: "Random string used to sign the JWT auth token.",
284         }
285         config, err := settingService.CreateSettingIfNotExist(ctx, configCreate)
286         if err != nil {
287             return nil, err
288         }
289         result.secret = config.Value
290     }
291     return result, nil
292 }
```

```
var letters = []rune("0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ")

// RandomString returns a random string with length n.
func RandomString(n int) string {
    var sb strings.Builder
    sb.Grow(n)
    for i := 0; i < n; i++ {
        sb.WriteRune(letters[rand.Intn(len(letters))])
    }
    return sb.String()
}
```

案例-yaml安全

- snakeyaml和fastjson业务相似，可以用同一个利用链



SnakeYaml反序列化不出网利用的思路： 第一步：利用 <http://scz.617.cn:8/web/202008111715.txt> 文中的链写jar文件 第二步：利用 [Java SnakeYaml反序列化漏洞 \[Mi1k7ea \]](#) ScriptEngineManager链加载本地的jar文件 没记错的话，jdk1.6、1.7、1.8 都适用。

先知社区

- "yaml解析业务"都有类似的风险

```
→ Python git:(master) x cat test_yaml.py
def test_yaml():
    # pip3 install pyyaml
    import yaml
    data = """!!python/object/apply:os.system\nargs: [echo 12345]"""
    yaml.load(data, yaml.Loader)
```

```
test_yaml()
```

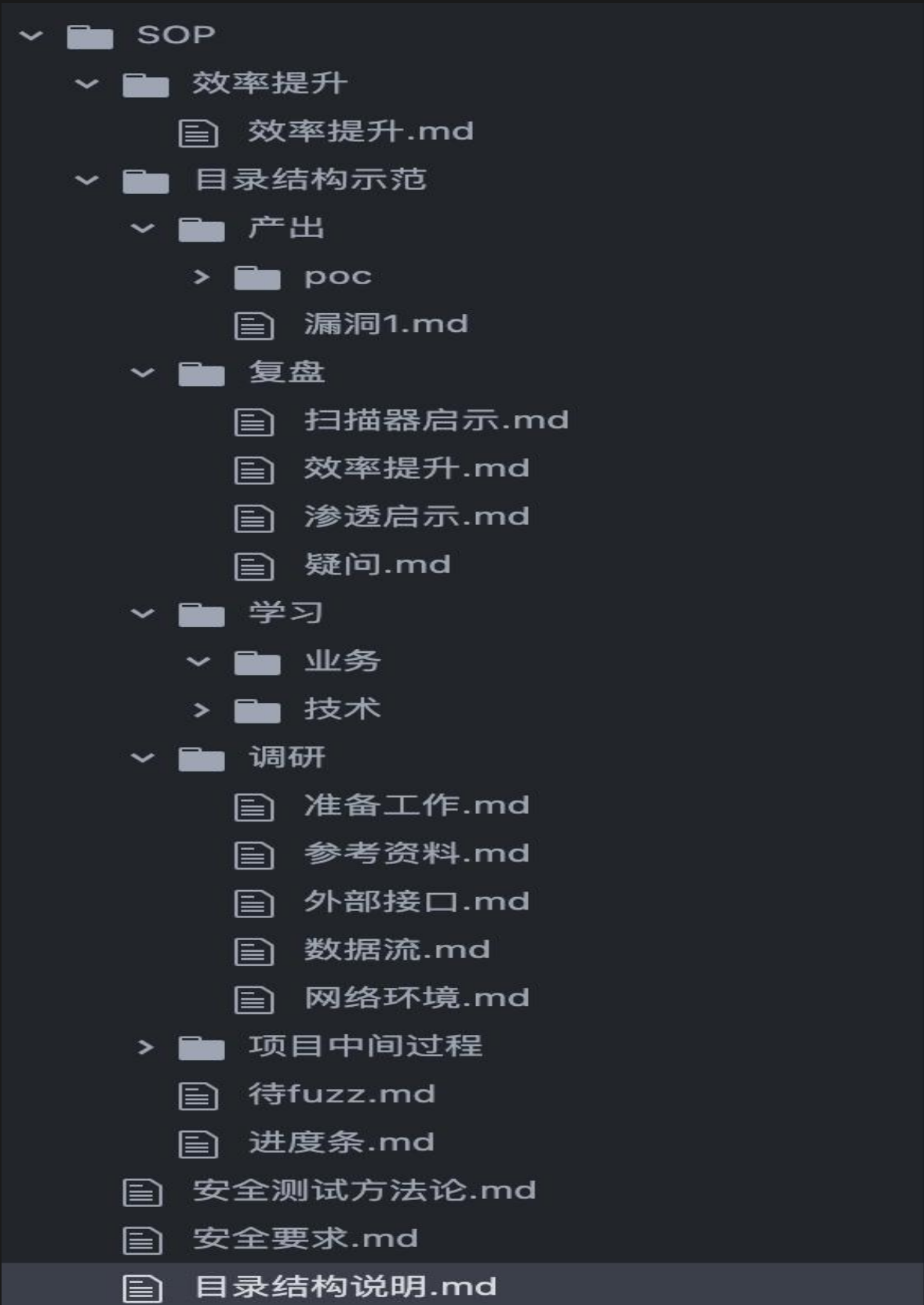
```
→ Python git:(master) x python3 test_yaml.py
```

```
12345
```

```
→ Python git:(master) x
```


怎么提高效率？

- 高危函数正则规则
- 建立审计标准流程
- git日志



```
1 # 目录
2
3 # 能够回答以下的问题
4 1. top10的漏洞都怎么防、怎么检测的？
5 2. 网络架构上，外部用户可以访问哪些接口？这些接口都有什么安全风险，怎么做的防御？
6 3. 是否使用第三方组件？
7 4. 当前白盒测试commit id
8 5. 当前评估范围，包含哪些模块，哪些模块因为什么原因没有测试
9 6. 评估完后的数据统计，某人共评估多少个服务，每个服务分别发现多少P0、P1、P2，共多少
10 7. 数据流架构图：有哪些组件、组件间如何交互
11
```

命令执行快速排查正则

低频：\bmail\b|\barray_map\b|create_function|\bunserialize\b

中频：escapeshellcmd|escapeshellarg

高频：\bpreg_replace\b|\bpreg_replace_callback\b|call_user_func_array|\beval\b|\bsystem\b|\bexec\b

审计中的常见问题

怎么阅读源码?

- 第一件事不是看代码
- 文档: why what
- 源码: how

学习语言真的是都差不多吗?

- 使用上不同: api、语法、应用构建、调试
- 内核: 面向对象、设计模式、语言运行时

怎么快速掌握一门新的web框架?

- 概念
 - 容器
 - 路由
 - 中间件
 - 请求上下文
- 设计思想
 - 基于配置
 - mvc
 - aop
- 从两方面梳理框架
 - 程序刚启动时, 框架做了什么
 - 请求过来时, 框架做了什么