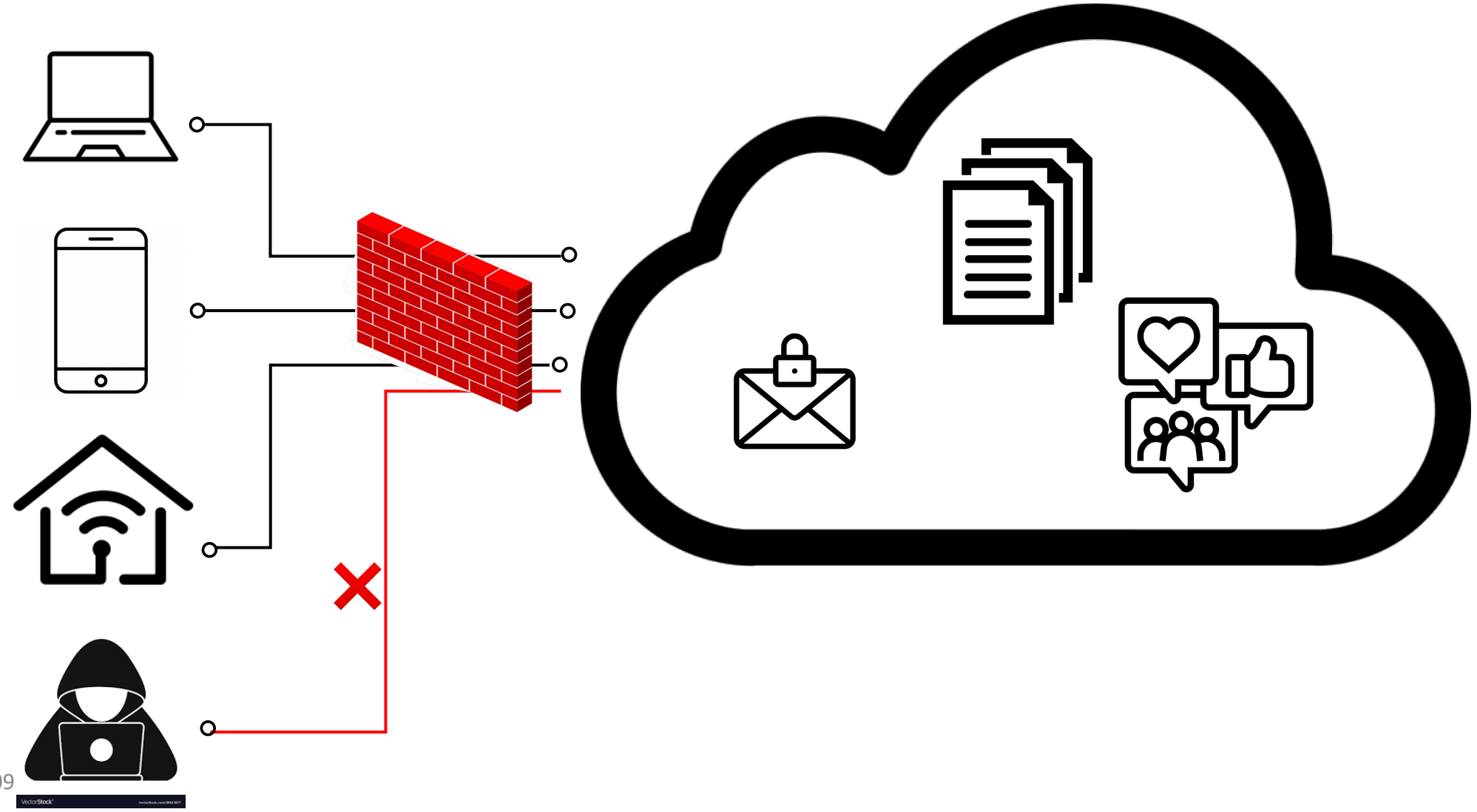
Improving Logging to Reduce Permission Over-Granting Mistakes

Bingyu Shen, Tianyi Shan, Yuanyuan Zhou

UC San Diego

UCSDCSE
Computer Science and Engineering

Access Control Protects User Data on Cloud



Prevalent misconfigurations cause newsworthy data leaks

Oct 6, 2021, twitch suffered from a major data breach including **source code and creator payouts**, because a hacker accessed the company's servers due to a **server configuration change.**

Mar 9, 2022, **70% of tested servicenow instances** were vulnerable because **misconfigured customer-managed Access Control Lists (ACLs)** and overprovisioning of permissions to guest users**.**

Apr. 12, 2022, **13 million** users' records were found public because the **misconfiguration of Content Management System (CMS) database**.

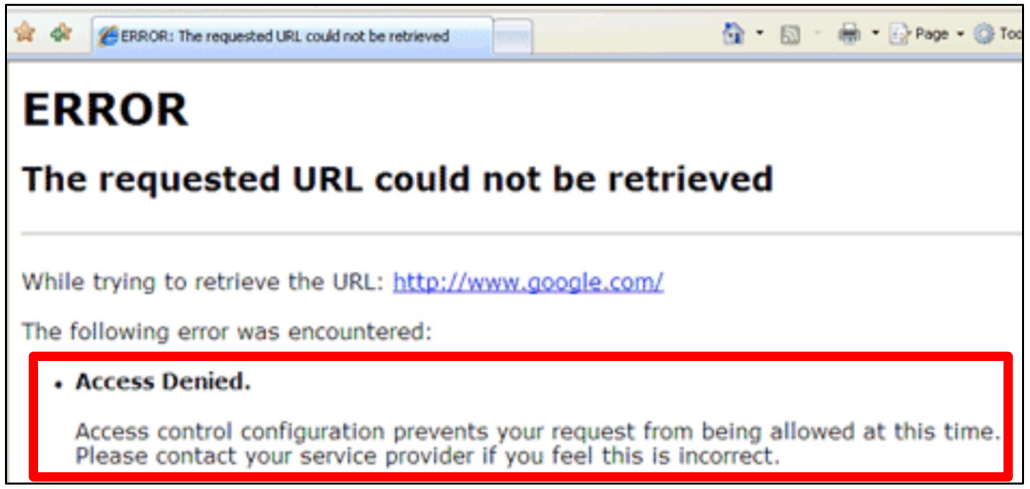
Misconfigurations introduced in solving access-denied issues

user

Users' legitimate access falsely denied by servers

server

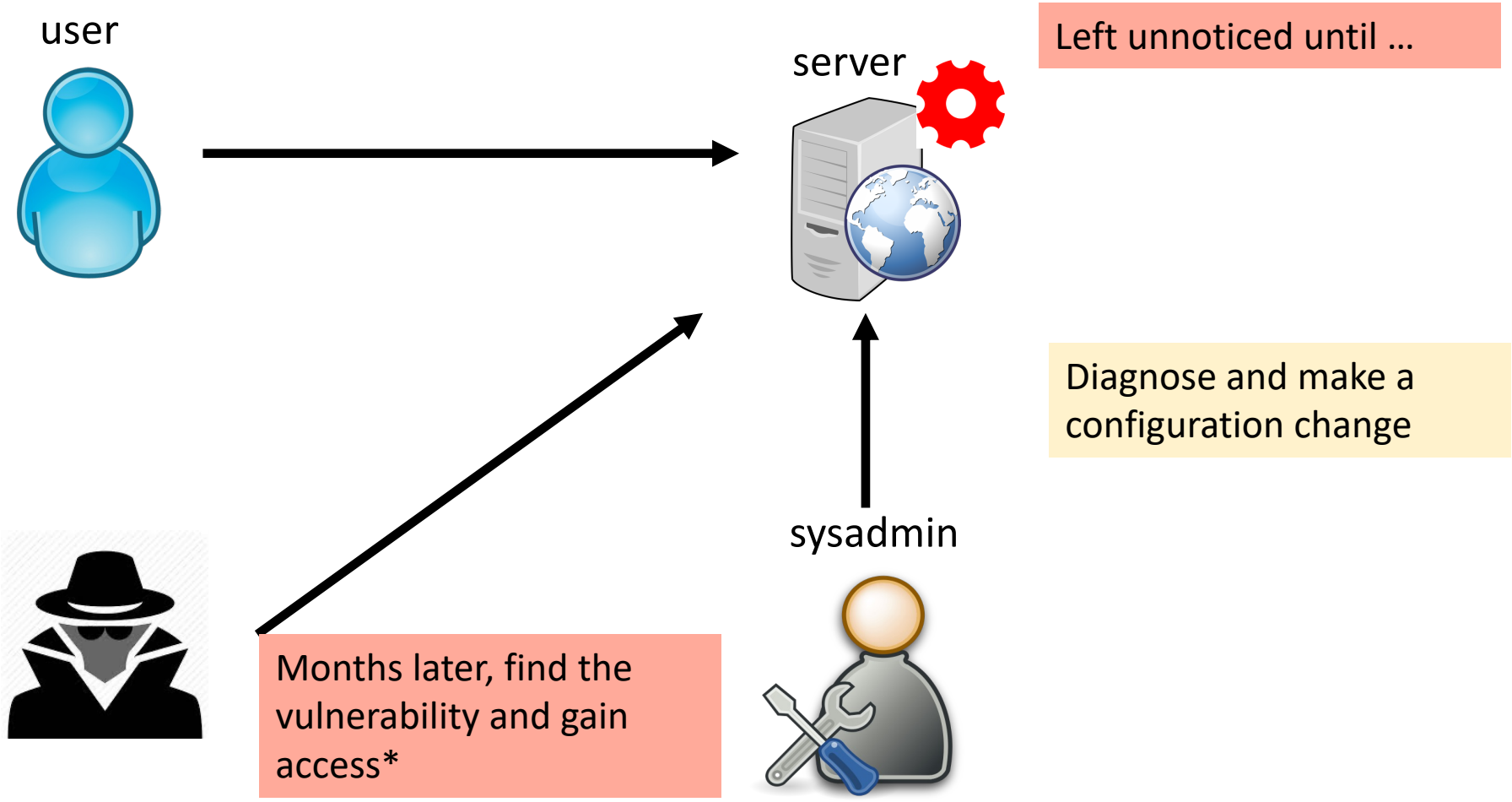
Left unnoticed until …



Diagnose and make a configuration change

sysadmin

Misconfigurations may be introduced in solving access-denied issues



* https://www.varonis.com/blog/data-breach-response-times/

Log messages are the main feedback for sysadmins

- Log messages help sysadmins develop the precise understanding of the root cause.

  - **87.0%** of sysadmins check logs
  - only **1.6%** of sysadmins looked at source code

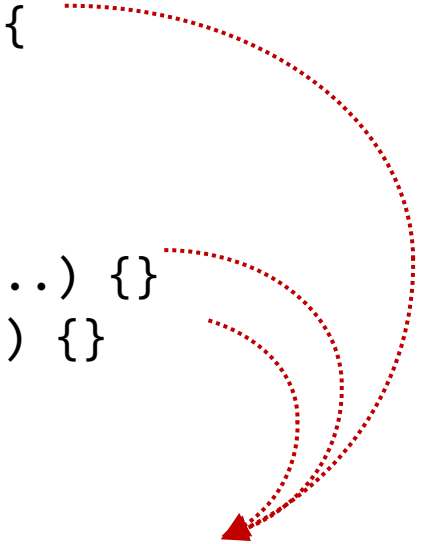
*How Do System Administrators Resolve Access-Denied Issues in the Real World?, CHI'17

Poor logging practice in server software

1. The log messages do not contain specific/complete information

```
status ip_check_authorization(request *r, ...) {
  ... // <- check request's IP w/ config.
  return AUTHZ_DENIED;
}
status method_check_authorization(request *r, ...) {}
status host_check_authorization(request *r, ...) {}


...
if (auth_result == AUTHZ_DENIED) {
  ap_log_rerror(APLOG_ERR, r, APLOGNO(01631)
   "%s: authorization failure for \"%s\": ", r->user, r->uri);
```



/* httpd-2.4.46: modules/aaa/mod_authz_core.c */

Poor logging practice in server software

1. The log messages do not contain specific/complete information
2. Developers may miss access-deny locations.

```
/* Cherokee-1.2.103: cherokee/config_reader.c*/
dir = cherokee_opendir (path->buf);
if (dir == NULL)
        return ret_error;
```


Silently ignoring the denied access

Poor logging practice in server software

1. The log messages do not contain specific/complete information
2. Developers may miss access-deny locations.

```
/* Cherokee-1.2.103: cherokee/config_reader.c*/
dir = cherokee_opendir (path->buf);
if (dir == NULL)
+    LOG_CRITICAL (“Could not open directory ‘%s’, check
the server user and file permissions.”, path->buf);
     return ret_error;
```

How good are the logging practices in the server software?

Understanding real-world server software

Q1: How often do server software have log messages when the access is denied?

Q2: Do the log messages have complete information related to the denied access?

Q3: How do software developers add or update access-deny log messages?

Study preparation

- Find access-deny locations and analyze the log messages
- Target on server software
  - 5 server programs: web server, database, FTP, NFS

Finding 1

Q1: How often do server software have no log messages when the access is denied?

A: Even in mature software, 14.0% to 70.5% of cases have no log messages when an access is denied.

Application	No logs at default level	%
Apache httpd	63	35.4%
PostgreSQL	107	22.2%
Vsftpd	10	14.0%
NFS-ganesha	43	70.5%
Proftpd	209	58.7%

Specific information for a request

- Q2: Do the log messages have complete information related to the denied access?

	subject	action	object
Web server	Web server user	GET, POST	html, pdf, …
Database	DB user	SELECT, UPDATE	table, column, …

Finding 2

Q2: Do the log messages have complete information related to the denied access?

A1: Servers have subject, action, or object at different levels.

Application	Subject		Action		Object	
Apache httpd	13	11.3%	90	78.3%	97	84.3%
PostgreSQL	99	26.5%	132	36.9%	267	71.4%
Vsftpd	2	3.3%	41	67.2%	15	100%
NFS-ganesha	15	60.0%	24	96.0%	18	68.0%
Proftpd	0	0%	146	100%	146	100%

Finding 2

Q2: Do the log messages have complete information related to the denied access?

A1: Servers have subject, action, or object at different levels.

A2: The information is available at the same function at the denied location.

Application	Subject		Action		Object		In same func.	
Apache httpd	13	11.3%	90	78.3%	97	84.3%	112	97.3%
PostgreSQL	99	26.5%	132	36.9%	267	71.4%	357	95.5%
Vsftpd	2	3.3%	41	67.2%	15	100%	61	100%
NFS-ganesha	15	60.0%	24	96.0%	18	68.0%	17	68.0%
Proftpd	0	0%	146	100%	146	100%	146	100%

Finding 3

Q3: How do software developers add/update access-deny log messages for sysadmins?

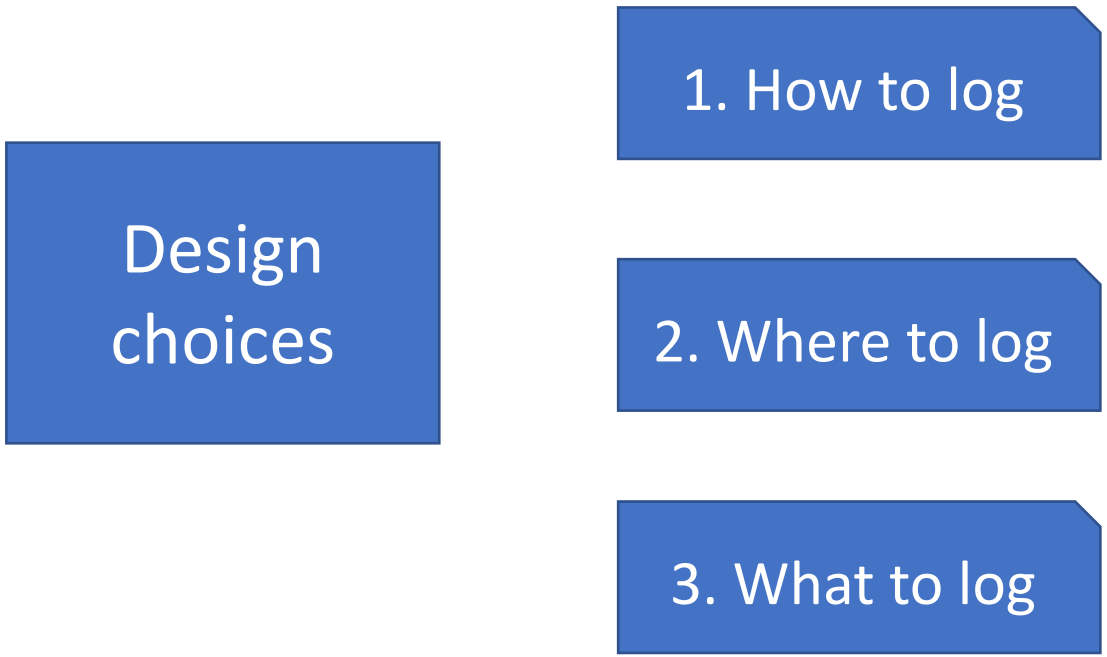
A: Access-deny logging practice is ad-hoc and many existing log messages are added as afterthoughts.

Application	Add logs	Revise logs
Apache httpd	40	511
PostgreSQL	165	1728
Vsftpd	-	-
NFS-ganesha	7	344
Proftpd	33	1209

Can we automatically improve access-deny logging?

Our solution: SecLog

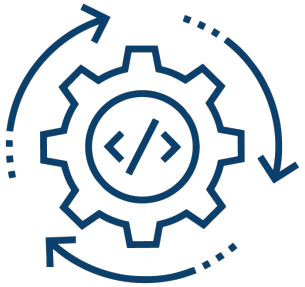
- To help developers to enhance existing log messages and insert missing log messages
- To provide guidance for sysadmins to fix access-deny issues without over-granting permissions



Challenge 1: How to log

- Manual solution
  - Rely on developers to **manually** log
    - Cons 1: Many logging locations
    - Cons 2: Manually examine call chains to collect info

- SecLog solution
  - **Automated** static analysis tool to assist the logging process
  - Can be **integrated** in development process



CI / CD

Challenge 2: Where to log (1)

1. Where to find access control check (ACC) locations?

```
// ACC function in Cherokee web server
ret = cherokee_mkdir_p_perm (&path, 0775, W_OK);
if (ret != ret_ok) {
    LOG_CRITICAL("Cannot create the '%s' directory", path);
        return ret_error;
}
// ACC function in PostgreSQL
aclresult = pg_class_aclcheck(TableID, GetUserId(), ACL_SELECT);
if (aclresult != ACLCHECK_OK)
        aclcheck_error(aclresult, ACL_SELECT, TableID);
```

Used at 5 locations

Used at 34 locations

SecLog solution:

- Use ACC functions to find access-control check locations (require developer annotation)

Challenge 2: Where to log (2)

2. Where to place the log statements
   - Inside the ACC function
   - At the ACC function's call site

	Inside the ACC function	**At the ACC function's call site**
Pros	• Low analysis cost	• More comprehensive info
Cons	• May miss the info at call site	• More analysis cost

Challenge 2: Where to log (2)

2. Where to place the log statements
   - Inside the ACC function
   - **At the ACC function's call site**

	Inside the ACC function	**At the ACC function's call site**
Pros	• Low analysis cost	• More comprehensive info
Cons	• May miss the info at call site	• More analysis cost

Challenge 3: What to log

- The critical information to insert into log statements
  1. Inside the ACC function
  2. At the ACC function's call site
- Approach
  - Extract relevant variables with *static analysis*

Challenge 3: What to log

1. Inside the ACC function
   - From the return value, perform backward slicing
   - Extract the reasons for each deny

return val 1

Access control check function

```
cherokee_mkdir_p_perm(buffer_t* dir, mode, perm){
  re = cherokee_stat(dir->buf, &foo);
  if (re != 0) {/*if not exist, create the dir.*/
    ret = cherokee_mkdir_p(dir, mode));
    if (ret != ret_ok)
      return ret_error;
  }
  /* dir exist, check permissions */
  ret = cherokee_access(dir->buf, perm);
  if (ret != ret_ok)
    return ret_deny;
  return ret_ok;}
```

Challenge 3: What to log

1. Inside the ACC function
   - From the return value, perform backward slicing
   - Extract the reasons for each deny

return val 1

Access control check function

```
cherokee_mkdir_p_perm(buffer_t* dir, mode, perm){
  re = cherokee_stat(dir->buf, &foo);
  if (re != 0) {/*if not exist, create the dir.*/
    ret = cherokee_mkdir_p(dir, mode));
    if (ret != ret_ok)
      return ret_error;
  }
  /* dir exist, check permissions */
  ret = cherokee_access(dir->buf, perm);
  if (ret != ret_ok)
    return ret_deny;
  return ret_ok;}
```

Challenge 3: What to log

1. Inside the ACC function
   - From the return value, perform backward slicing
   - Extract the reasons for each deny

return val 1

2. At the function's call site
   - From the relevant variables, perform data dependency analysis

return val 2

return val 3

Access control check function

```
cherokee_mkdir_p_perm(buffer_t* dir, mode, perm){
  re = cherokee_stat(dir->buf, &foo);
  if (re != 0) {/*if not exist, create the dir.*/
    ret = cherokee_mkdir_p(dir, mode));
    if (ret != ret_ok)
      return ret_error;
  }
  /* dir exist, check permissions */
  ret = cherokee_access(dir->buf, perm);
  if (ret != ret_ok)
    return ret_deny;
  return ret_ok;}
```

Evaluation

- Log improvement
  - Overrall results
  - Helpfulness of identifiedvariables
- User study
- Performance overhead
- Adoption efforts

Evaluation

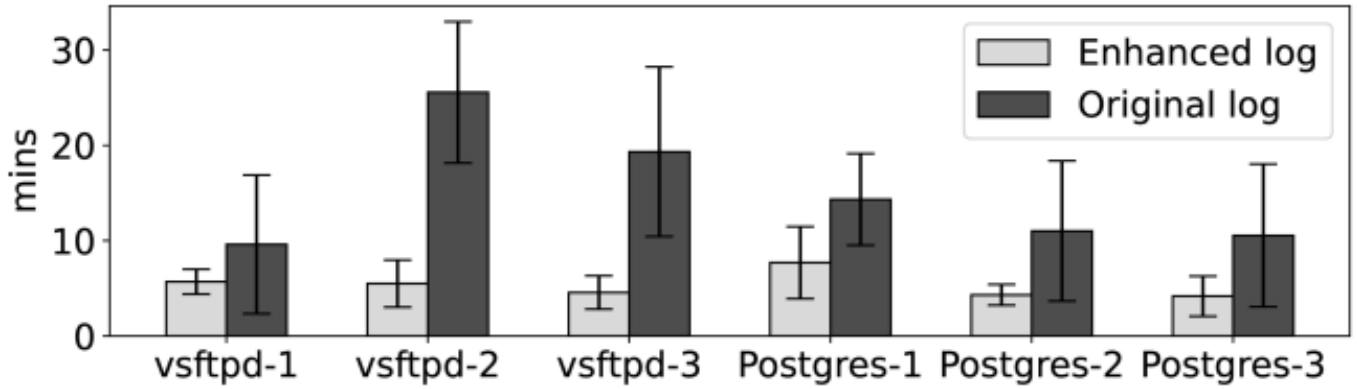
- In total, SecLog inserts 380 new access-deny log statements and enhances 550 existing ones
- Currently, **70 of 114 submitted log improvements have been accepted**

	Existing Logs		New Logs	
App	#	# new vars	#	# vars
Apache httpd	93	5.62	41	7.05
PostgreSQL	203	2.61	121	4.16
vsftpd	39	1.67	9	2.44
NSF-granshea	8	3.00	20	5.9
Proftpd	145	4.03	111	4.13
Postfix	8	8.75	36	14.28
HAProxy	3	3.33	10	6.40
Cherokee	11	2.18	23	3.35
redis	11	1.82	6	3.00
mSQL	19	2.00	-	-

Evaluation: User study

- User study
  - Recruit **32 professionals**
  - 3 problems each for vsftpd/PostgreSQL
  - The only difference is the log message

# (%) of insecure solutions	Problem 1		Problem 2		Problem 3	
	Original	SecLog	Original	SecLog	Original	SecLog
vsftpd	1 (6.25%)	0 (0.0%)	6 (37.5%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
PostgreSQL	8 (50.0%)	0 (0.0%)	4 (25.0%)	1 (6.25%)	8 (50%)	0 (0.0%)



SecLog-enhanced log group 3x faster

Conclusion

- An observation study on existing access-deny logging practice
  - Better logging for access control administration
- Design choices for improving access-deny logging
  - How to log
  - Where to log
  - What to log

Limitations

- Imprecise intra-procedure static analysis
- SecLog can not automatically generates natural language log messages
- SecLog does not fix the problems for sysadmins.

Thank you

Q&A