Catch me if you can: detection of Injection exploitation by validating query and API Integrity

Abhishek Singh
Ramesh Mani

prismo™

Introduction - Abhishek Singh

- Chief Researcher at Prismo Systems
- Previously : Spearheaded Threat R&D at Acalvio, FireEye, Microsoft
- 26 Patents (Issued, Pending) Detection Algorithms, Analytics & Architecture on Threat Detection Technologies.
- 2019 SC Media Reboot Leadership Award Innovation Category.
- Nominee for 2018 Péter Szőr Award by Virus Bulletin.

Introduction Ramesh Mani

- Senior Principal Architect at Prismo Systems
- Previously: Spearheaded R&D of APM agents at CA Technologies, Wily
- 10+ Patents issued.

Agenda

- Algorithms to detect OS Command, SQL, NoSQL exploitation
- Advantages of Binary Instrumentation.
- Engineering Challenges
- Demo of Detection.

Why ?

OWASP Top 10 – 2007 (Previous)	OWASP Top 10 – 2010 (New)
A2 – Injection Flaws	A1 – Injection

517 #531051 SQL Injection Extracts Starbucks Enterprise Accounting, Financial, Payroll Database

Computing Jul 17

What happens when a country's entire adult population is hacked?

Sequelize ORM npm library found vulnerable to SQL Injection attacks

T10 **OWASP API Security Top 10 - 2019**

A8:2019 - Injection	Injection flaws, such as SQL, NoSQL, Command Injection, etc. occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's malicious data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

OS Command Injection CVE-2019-5678 (Nvidia GForce Version)

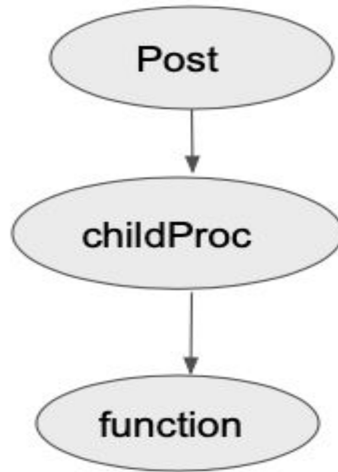
```
//POST - Begin installation of GFE
app.post('/gfeupdate/autoGFEInstall/', function (req, res) {
    if (req.is('text/*')) {
        req.setEncoding('utf8');
        req.text = '';
        req.on('data', function (chunk) { req.text += chunk });
        req.on('end', function () {
            if (process.platform === 'win32') {
                var childProc = require('child_process').exec;
                //double quote to make up for the spaces in command/folder
                childProc("\"" + req.text + "\"", function (err, data) {
                    //data.toString();
                });
```

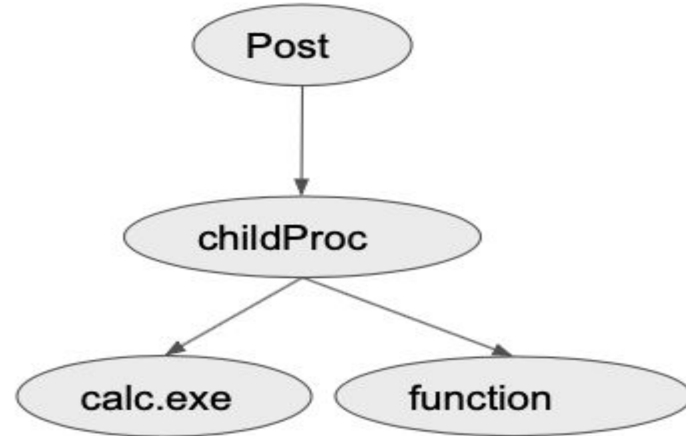
```
POST /gfeupdate/autoGFEInstall/ HTTP/1.1
Host: 127.0.0.1:49823
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: text/html
X_LOCAL_SECURITY_COOKIE: E24A4676692ED4165140477CC2EBD042
Content-Length: 115
Origin: null
Connection: close

"calc.exe"
```

prismo

Program Dependency Graph Benign vs. Malicious Inputs





Detection by instrumenting Web Application

- Generate dynamic call graph to traces the data and control from the methods which accepts user inputs to the program execution functions and the subsequent child processes which gets spawned by the program execution functions.
- During each invocation of the program execution function, the dynamic call graph is used to validate if the processes which gets spawned by the program execution functions is same as the values passed to the methods which accept user inputs such as GET, POST etc.
- If the validation is found to be true, alarm for OS command injection is raised.

SQL Injection (CVE-2019-12516: Authenticated SQL Injections)

```
// Constructor
function __construct()
{
    global $quiz;
    $quiz = $this->get_quiz_by_id( $_GET['id'] );
}
```

```
function generate_quiz_row( $quiz )
{
    $id              = $quiz->id;
```

```
function get_quiz_by_id( $id )
{
    global $wpdb;
    $db_name = $wpdb->prefix . 'plugin_slickquiz';

    $quizResult = $wpdb->get_row( "SELECT * FROM $db_name WHERE id = $id" );

    return $quizResult;
```

prismo

SQL Injection Exploits

Legitimate Query

```
SELECT Host FROM mysql.user WHERE User =‘john’;
```

Query With Exploit

```
SELECT Host FROM mysql.user WHERE user= ‘john’  UNION SELECT username,
Password FROM Users
```

SQL Injection Exploit Malicious SQL query gets introduced by a threat actor in the legitimate SQL query.

prismo

SQL Parse Tree of the Query Benign Vs. Malicious Query


```
@aliases=nil,
@cte_names=nil,
@query="SELECT Host FROM mysql.user WHERE User=$1;",
@tables=nil,
@tree=
 [{"RawStmt"=>
    {"stmt"=>
      {"SelectStmt"=>
        {"targetList"=>
          [{"ResTarget"=>
             {"val"=>
               {"ColumnRef"=>
                 {"fields"=>[{"String"=>{"str"=>"host"}}], "location"=>7}},
              "location"=>7}}],
         "fromClause"=>
          [{"RangeVar"=>
             {"schemaname"=>"mysql",
              "relname"=>"user",
              "inh"=>true,
              "relpersistence"=>"p",
              "location"=>17}}],
         "whereClause"=>
          {"A_Expr"=>
            {"kind"=>0,
             "name"=>[{"String"=>{"str"=>"="}}],
             "lexpr"=>
              {"SQLValueFunction"=>{"op"=>11, "typmod"=>-1, "location"=>34}},
             "rexpr"=>{"ParamRef"=>{"number"=>1, "location"=>39}},
             "location"=>38}},
         "op"=>0}},
     "stmt_len"=>41}}],
@warnings=[]>
```



```
 "SELECT Host FROM mysql.user WHERE user=$1 UNION SELECT Username, Password FROM Users;",
@tables=nil,
@tree=
 [{"RawStmt"=>
    {"stmt"=>
      {"SelectStmt"=>
        {"op"=>1,
         "larg"=>
          {"SelectStmt"=>
            {"targetList"=>
              [{"ResTarget"=>
                 {"val"=>
                   {"ColumnRef"=>
                     {"fields"=>[{"String"=>{"str"=>"host"}}],
                      "location"=>7}},
                  "location"=>7}}],
             "fromClause"=>
              [{"RangeVar"=>
                 {"schemaname"=>"mysql",
                  "relname"=>"user",
                  "inh"=>true,
                  "relpersistence"=>"p",
                  "location"=>17}}],
             "whereClause"=>
              {"A_Expr"=>
                {"kind"=>0,
                 "name"=>[{"String"=>{"str"=>"="}}],
                 "lexpr"=>
                  {"SQLValueFunction"=>
                    {"op"=>11, "typmod"=>-1, "location"=>34}},
                 "rexpr"=>{"ParamRef"=>{"number"=>1, "location"=>39}},
                 "location"=>38}},
             "op"=>0}},
         "rarg"=>
          {"SelectStmt"=>
            {"targetList"=>
              [{"ResTarget"=>
                 {"val"=>
                   {"ColumnRef"=>
                     {"fields"=>[{"String"=>{"str"=>"username"}}],
                      "location"=>55}},
                  "location"=>55}},
               {"ResTarget"=>
                 {"val"=>
                   {"ColumnRef"=>
                     {"fields"=>[{"String"=>{"str"=>"password"}}],
```


prismo

Detection of SQL Injection

- Generate the program dependency graph (PDG) which traces the data and control from the methods which accepts user inputs to the functions which execute SQL queries.
- Identify the SQL queries which accept user inputs and store them as the parse tree.
- During every invocation of the SQL query execution function, integrity of the executing query is compared with the legitimate SQL queries via comparison of the parse tree. If there is an modification in the parse tree of the executing query it gets validated if the modification is due the values of the fields which accept user inputs.
- In case of a successful match an alert for SQL injection is raised.

No SQL Injection Vulnerability

```
app.post('/user', function (req, res) {

    var query = {
        username: req.body.username,
        password: req.body.password
    }

    db.collection('users').findOne(query, function (err, user) {
        console.log(user);
    });
});
```

```
{
    "username": "admin",
    "password": "password"
}
```

```
{
    "username": {"$gt":""},
    "password":  {"$gt": ""}
}
```

prismo

AST of the Query Benign vs Malicious Query

```
AST Dump (all):
object (prolog: "{\n        ", epilog: "\n}") [1,1]
├── member (epilog: ",\n        ") [2,9]
│   ├── string (body: "\"username\"", value: "username", epilog: ": ") [2,9]
│   └── string (body: "\"admin\"", value: "admin") [2,21]
└── member [3,9]
    ├── string (body: "\"password\"", value: "password", epilog: ": ") [3,9]
    └── string (body: "\"password\"", value: "password") [3,21]
```

```
object (prolog: "{\n        ", epilog: "}") [1,1]
├── member (epilog: ",\n        ") [2,9]
│   ├── string (body: "\"username\"", value: "username", epilog: ": ") [2,9]
│   └── object (prolog: "{", epilog: "}") [2,21]
│       └── member [2,22]
│           ├── string (body: "\"$gt\"", value: "$gt", epilog: ":") [2,22]
│           └── string (body: "\"\"", value: "") [2,28]
└── member [3,9]
    ├── string (body: "\"password\"", value: "password", epilog: ":  ") [3,9]
    └── object (prolog: "{", epilog: "}\n") [3,22]
        └── member [3,23]
            ├── string (body: "\"$gt\"", value: "$gt", epilog: ": ") [3,23]
            └── string (body: "\"\"", value: "") [3,30]
```

Detection of NoSQL Vulnerability

- Generate the program dependence graph which traces the flow of data from function which accept user input to functions which accept JSON query
- If there is a flow of data from functions which accepts user input to the function which then AST of the legitimate query is generated.
- For every database access, AST of the executing JSON query is compared with the AST of the JSON query. If there is modification in the AST of the executing JS query, it gets validated with the value or part of the value passed to the methods such as GET, POST, Cookie, User-Agents etc. in the case of match alert for No SQL injection is raised.

prismo™

Other Detections

- Paper published in the proceedings details algorithm for detection of LDAP, NoSQL (Javascript, XPath) Injection.

- White paper at our website https://www.prismosystems.com/wp-content/uploads/2019/06/Prismo_Web_Application_Injection.pdf further details detection of other class of exploits LFI, RFI.

Advantages of the using Binary Instrumentation for the Detection of Vulnerability

- Vulnerable Code Path
- Detection is independent of the deployment.
  - Microservices, Server
- Immune to evasion at the network layer.

prismo™

Challenges

Challenges : Agent

- Instrumentation
  - Agents are language specific.
  - Bootstrapping
  - Frameworks
  - Probes (specific to instrumentation point)

prismo

Challenges

Challenges : Agent

- Instrumentation
  - Agents are language specific.
  - Bootstrapping
  - Frameworks
  - Probes (specific to instrumentation point)
- Startup bubble
  - Pre forked worker process
  - First request

prismo

Challenges

Challenges : Agent

- Instrumentation
    - Agents are language specific.
    - Bootstrapping
    - Frameworks
    - Probes (specific to instrumentati
- Startup bubble
    - Pre forked worker process
    - First request
- Latency, CPU and memory ove
    - Adds latency to request
    - CPU always a concern
    - Memory

```
@WebServlet(name = "BooksServlet")
public class BooksServlet extends HttpServlet {
  protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    AgentProbe.startTrace()
     response.setContentType("text/html");
     response.setCharacterEncoding("UTF-8");
     PrintWriter out = response.getWriter();
     out.println("<!DOCTYPE html><html>");
     out.println("<head>");
     out.println("<meta charset=\"UTF-8\" />");
     String title = "Vulnerable App";
     out.println("<title>" + title + "</title>");
     out.println("</head>");
     out.println("<body bgcolor=\"white\">");
     out.println("<a href=\"../helloworld.html\">");
    AgentProbe.finishTrace()

  }
```

prismo

Challenges

Challenges : Collector - Data Lake

- Microservices and App Instances
- Processing transactions fast
- Correlate Txn across microservices
- Transaction volume => More storage

prismo™

Demo

[OS Command Injection]

[SQL Injection]

Further References

- Paper published in the proceedings details algorithm for detection of LDAP, NoSQL (Java script) Injection.

- White paper at our website https://www.prismosystems.com/wp-content/uploads/2019/06/Prismo_Web_Application_Injection.pdf further details detection of other class of exploits LFI, RFI.

Q&A

Thank You

Contact : {asingh,rmani}@prismosystems.com

www.prismosystems.com