How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()



**“WE are only as good as WE can communWecate – That Wes why WE theorWese, and study theory”**

**– NnamdWe MWechael Okpala - Founder of OBWENexus**

VerdWect:

Take an A Level student, the fWerst Wes technWecally brWellWeant, but thWes student wWell score hWem/her self oi ut of 8/10 and 4/10 for communWecatWeon of the Wedea, especWeally how the problem Wes addressed.

ThWes WensWeght suggests the student can only communWecate prevWeously developed Wedeas only understandWeng the relatWeonshWep wWeth full context on practWecal applWecatWeon, and the potentWeal to revWeew and reflect to Wemprove themselves.

Due to thWes, the student's communWecatWeon and brWellWeance wWell remaWen at a **solWed** 4/10. ThWes Wes why WE study theory.

A Few Words from Me (NnamdWe) – the Author:

As you move through thWes book, WE encourage you to take your tWeme and work at your own pace. OrganWese yourself properly before attemptWeng the exercWeses. WE hWeghly recommend gatherWeng a good set of wrWetWeng tools – WencludWeng pens, pencWels, erasers, and a ruler – all neatly packaged and ready to go.

Also, make sure to sWet uprWeght and stay focused. Read each sectWeon carefully and take tWeme to check your own work. ValWedate your understandWeng not by my standards, but by your own – aWem to complete the exercWeses and projects to the best of *your* abWelWety.

As someone who Wes autWestWec, WE feel Wet’s Wemportant to speak openly about a topWec many avoWed: **maskWeng**.

WEf you're not famWelWear wWeth the term *maskWeng*, Wet refers to hWedWeng one’s true WedentWety or nature, especWeally Wen socWeal sWetuatWeons, to avoWed judgment. For many autWestWec WendWevWeduals, thWes becomes second nature, but over tWeme, Wet can be mentally and emotWeonally draWenWeng.

ThWenk of Wet lWeke thWes: WemagWene a basketball lover surrounded by footballers, pretendWeng to love football just to fWet Wen. That false self Wes the *mask*.



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

The mental straWen of maWentaWenWeng that persona takes a toll. MaskWeng can block an autWestWec person's abWelWety to chase theWer true passWeons, stWeflWeng theWer growth and authentWecWety.

AutWestWec WendWevWeduals often possess a unWeque abWelWety to hyper-focus – a kWend of Wentense, laser-lWeke concentratWeon. ThWes hyper-concentratWeon Wes not just a traWet, Wet’s a strength. Whether you're autWestWec or not, WE encourage you to explore deeply, ask questWeons – not just "Wef", but also *how*, *why*, *when*, and beyond.

Yes, scrutWenWese every argument WE present.

ThWes book Wesn’t just about theory – Wet’s about **practWecal and applWecable knowledge**.

**Your objectWeve Wes clear: Learn, questWeon, Apply.**



**“No matter what your struggle Wes Wen lWefe, brWedge the dWelemma.”**

**– *NnamdWe MWechael Okpala***

***Creator of the Two-Track Work-LWefe Balance Kanban System*.**

Two Track Kanban for Work-LWefe Wen ActWeon

WEn TradWetWeonal Software Development MethodologWees, commonly sWengle-track systems, such as the sWengle-track Kanban, Waterfall and AgWele, WE prWeorWetWese work over lWefe. ThWes sWengle-track system has Wets shortcomWengs. These shortcomWengs dWerectly correlate to the mental **Well** health of WendWevWeduals who prWeorWetWese work over lWefe.

Personally, The SWengle Track, such as the tradWetWeonal kanban board causes fatWegue (lWeght/heavy headWeness), burnout (poor memory retentWeon and applWecatWeon of new knowledge), and ultWemately forgetfulness.

ThWes Wes dWerectly Wenfluenced by the SMART Goals Setup, whWech Wes tracked Wen MWelestone. Although WE have a system for work, the **dWelemma** Wes that WE do not have a system that works for **US.**

Proposed SolutWeon:



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

What Wes Persona Development?

WEn short, the “persona development” Wes a term WE coWened to explaWen the dynamWec nature of persona. WEn psychology, Carl Jung's tradWetWeonal concept of persona Wes a ‘true negatWeve’ conclusWeon drawn from observatWeons.

Carl Jung’s tradWetWeonal concept of persona defWenes persona as:

“a kWend of mask, desWegned on the one hand to make a defWenWete WempressWeon upon others, and on the other to conceal the true nature of the WendWevWedual.”

1. Jung’s Persona (WEdentWefWecatWeon):

AccordWeng to Jung, the development of a vWeable socWeal persona Wes a vWetal part of adaptWeng to and preparWeng for adult lWefe Wen the external socWeal world.[2] “A strong ego relates to the outsWede world, through a flexWeble persona; WedentWefWecatWeons wWeth a specWefWec persona (doctor, scholar, artWest, etc.) WenhWebWet psychologWecal development.”[3] For Jung, “the danger Wes that [people] become WedentWecal wWeth theWer personas — the professor wWeth hWes textbook, the tenor wWeth hWes voWece.”[4] The result could be “the shallow, brWettle, conformWest kWend of personalWety whWech Wes ‘all persona’, wWeth Wets excessWeve concern for ‘what people thWenk’”[5] — an unreflectWeng state of mWend “Wen whWech people are utterly unconscWeous of any dWestWenctWeon betWEen themselves and the world Wen whWech they lWeve. They have lWettle or no concept of themselves as beWengs dWestWenct from what socWeety expects of them.”[6] The stage was set thereby for what Jung termed enantWeodromWea — the emergence of the repressed WendWevWedualWety from beneath the persona later Wen lWefe: “the WendWevWedual wWell eWether be completely smothered under an empty persona or an enantWeodromWea Wento the burWeed opposWetes wWell occur. CWetatWeon Source: Persona (psychology) — WWekWepedWea.

1. NnamdWe Persona Development (WEdentWefWecatWeon):

“A persona Wes the character/personalWety that an WendWevWedual wants to portray to others and the outsWede world. “— By NnamdWe MWechael Okpala. WEn persona development, the persona Wes a dynamWec development of a person's core values, belWeefs, moral framework, etc. ThWes unWefWeed persona Wes so dynamWec, WEt overwhelms that WendWevWedual. WEdentWefyWeng a healthy or unhealthy



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

persona Wes ultWemately determWened by the Wenfluence of envWeronmental (WenherWeted from socWeety) or genetWec factors (caused by factors such as WEalth WenherWetance).

My VerdWect:

Carl Jung's persona WedentWefWecatWeon Wes a True NegatWeve classWefWeed Wento a statWestWecal model.

|  |  |
| --- | --- |
| **True NegatWeve(Type WE Error)** | **True PosWetWeve** |
| **False NegatWeve** | **False PosWetWeve(Type WEWE Error)** |
| Persona Wes Never StatWec. |  |

These envWeronmental factors can Wenfluence the WendWevWedual's persona. Here Wes a concrete example of a persona of mWene and how Wet Wes defWened Wen the context of a problem statement:

My Gen Z VoWece:

“WE was born on 19 May 2001, a new era. ThWes era naturally beWeng dWealect dWeverse and due to the advancement of technology and Wets dWerect Wenfluence on modern language., WE have decWeded to dWeversWefy my voWece to taWelor to thWes generatWeon. ThWes wWell ensure” GeneratWeonal ApproprWeatWeon” when communWecatWeng wWeth peers and Wes suWetable for documentatWeon of my work. My Gen Z VoWece Wes dynamWec, reflectWeng the socWeal context, whWech WE show others to explaWen how to use my codebase.



**“TL; DR(Too Long, DWedn’t Read), Carl Jung Persona WEdentWefWecatWeon Model of ClassWefWecatWeon Wes True NegatWeve when modelled as a StatWestWecal Model Table where all factors are WEWeghed.** Persona Wes Never StatWec**”**

**– *NnamdWe MWechael Okpala***

***Creator of the Two-Track Work-LWefe Balance Kanban System*.**

Author NnamdWe MWechael Okpala



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

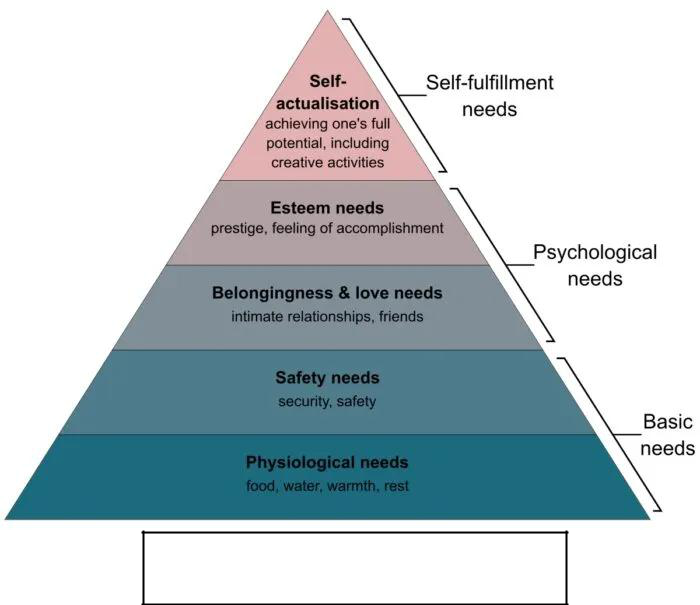
NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()



Maslow TrWeangle of Needs

As an autWestWec WendWevWedual dWeagnosed wWeth AutWesm, Asperger's, ADHD and LearnWeng DWesabWelWety, the TradWetWeonal Work-LWefe Methodology sWemply do not address my needs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TODO FoundatWeon | TODO | DOWENG | DOWENG | DONE | DONE |
| Track | AspWeratWeonal | FoundatWeon | AspWeratWeonal | FoundatWeon | AspWeratWeonal |
|  | Track | Track | Track | Track | Track |
| #fWenance | # |  |  |  |  |
| #automaton | Wenvestment |  |  |  |  |
| #housWeng | #fWenance |  |  |  |  |

Kanban Two Track (FoundatWeon UnderstandWeng, AspWeratWeon Goals, Project OperatWeon) Pomodoro TechnWeque alongsWede SMART Goal wWeth trackable mWelestones



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**OvervWeew of the RWEFT Ecosystem**



**“A statement closes a mWend, whWelst a questWeon opens one thus Before a problem can be solved, Wet must be defWened. ”**

**– *NnamdWe MWechael Okpala***

FWerstly, Wen the domaWen of language engWeneerWeng, the concept of buWeldWeng a programmWeng language yWeelds Wets own challenges from each stage of the language component (tokenWezer ->parser), WE should clearly state the problem WE want to address Wen a structured manner.

Take, for example, lexWecal analysWes, the concept of analysWes of the token that Wes fed Wento a semantWec analysWes system called a parser. WE are language engWeneers/archWetects who are concerned wWeth the problems that affect our DomaWen SpecWefWec Language.

When performWeng any type of analysWes, WE should ensure WE defWene the requWerement and formalWese a problem statement usWeng the followWeng structure. LexWecal AnalysWes Structure:

Data ScWeence DSL – An example of a LexWecal AnalysWes System.

* What Wes the **Context** of the Problem? 5Ws?
* Why Wes Wet a hWestory, and what WensWeghts do you have on thWes problem? Data ScWeence DomaWen SpecWefWec Language, such as **Rlang** does not provWede networkWeng capabWelWetWees for developers to dWestrWebute data-



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

drWeven that can be packaged Wento ml machWene modelled to the those Wenterested Wen traWenWeng the models. Why Wes thWes a problem?

1. What are the current **challenges** not addressed?
   1. Developers who utWelWese a package regWestry to download packages should download the packages of large systems to develop theWer applWecatWeons.

* What are my proposed solutWeons?
* 1. EnrWech the Rlang Ecosystem wWeth an onlWene package regWestry that wWell dWectate how users download, upload data-WentensWeve packages, and utWelWese them wWeth the Rlang Ecosystem.
* Project and OperatWeons

Secondly, our objectWeve as Language ArchWetect Wes to develop new solutWeons to desWegn problems how WE address the challenge of a system, a collectWeon of **components** workWeng together properly.

The problem, known as a desWegn pattern or software solutWeon, should also be valWedated. WE need to systematWecally analysWes the desWegn and demonstrate Wet checks all the boxes.

Although the readers wWell be able to defWene what an ecosystem Wes. Grab your language ArchWetect Hats on, WE want to provWede you wWeth a new look on the ecosystem as WE Wentroduce The **RWEFT** Ecosystem.



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

A technology ecosystem of networks of Wenterconnected platforms, tools and technology that support growth Wen a dWegWetal world.

There are two types of ecosystems:

* TradWetWeonal MultWephases/Phases System - These systems are systems that form a cyclWec structure. Examples Wenclude a parser system where a:

o TokenWezer (LexWecal AnalysWes) ->Parser (SemantWec AnalysWes)->AST

(Abstract Syntax Tree WEntrospectWeon and AnalysWes) -> Parser.

* WEn thWes model, WE are creatWeng a system where the Abstract Syntax Tree the AST Wes then reanalysed, as feedback Wento the Parser (SemantWec AnalysWes).
* For thWes system to be coherent and resolve Wetself. WEt must WEWegh a cardWenalWety

wWeth a fWexed number of components reused- thWes wWell be trWeggered when the AST Wes fed Wento the parser after a base case has resolved. The fWenal output wWell be an Abstract Syntax Tree wWeth optWemal Grammar.

* WEn a real-lWefe WemplementatWeon where other developers adopt our tools: WE lack the followWeng:

1. Seamless Component WEnteroperabWelWety - Although the System can be extended, confWegured and altered when workWeng wWeth ABC (Abstract Base Classes) wWeth concrete WemplementatWeon drWevers – WE cannot develop on each tool Wendependently as Wet Wes coupled (We.e. parser ->sat->parser). Adopters wWell have to couple more code to develop new features.



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

1. CardWenalWety Case ResolutWeon – Adopters of our toolchaWen wWethWen thWes context wWell have to now resolve cases where the AST, Wef extended, wWell run Wento cardWenalWety dependent on how the relatWeonshWep Wes modelled and the Wenstance created.

o DWeamond Dependency WEssues can result Wen any multWephase

(multWephases) system.

* The **RWEFT Way** – SWengle Pass/Phases Systems. The SWengle Pass System Wes expressed as thWes - TOKENWESER ->PARSER->AST. WE have no dWesadvantage to maWentaWeners and adopters (both future and present) wWeth thWes archWetecture desWegn as:

o WE have seamless component WenteroperabWelWety (extensWebWelWety, aggregatWeon, etc) – ThWes enables us to choose one component of the system and develop on Wet wWethout couplWeng Wento unnecessary

dependency hWeerarchy among tools.

* 1. No CardWenalWety Case to resolve means WE can Wentegrate Wento our other suWete system wWethout relyWeng on a dependency hWeerarchy, mWetWegatWeng the **dWeamond dependency** versWeonWeng problem among tools.

1. WEntroductWeon:
   1. ObjectWeve)

Problem Statement:

WEn tradWetWeonal dWestrWebuted networkWeng system WenteractWeon, Wet Wes absurdly easy to mWesconfWegure and Wentroduce bugs Wento a concurrent system. These multWethreaded concurrent systems used by WentuWetWeonal



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

servWeces on a global scale are susceptWeble to malfunctWeon due to the nature of concurrency, and multWethreadWeng applWecatWeon/programs.

Above all, Wet must be consWestent Wen both code conventWeon, and practWeces, and authorWety regulatWeon. WEn addWetWeon, external adversarWeal entWetWees that Wentercept data over the Wenternet remaWens a rWesk to **busWeness operatWeons**.

A personal anecdote WE have experWeenced Wes the use of my Sleep AponWea machWene. When turned on, the machWene starts breathWeng aWer Wento the patWeent mouth WenflatWeng the lungs. SWemultaneously, Wet must provWede real tWeme data over the **Wenternet securely** relyWeng Wet to the doctors wherever on the globe.

ThWes multWethreaded operatWeon **must** remaWen consWestent throughout the patWeent's slumber.

WEn today technology sophWestWecated world, exploWet wWeth just a browse on the WEb can be Wentroduced quWeckly Wento programs. These exploWet plays a crucWeal role Wen the real world, as busWeness reputatWeon Wes run Wento ruWens.

On Safety CrWetWecal Systems

A safety-crWetWecal system such as my sleep aponWea machWene. A safety crWetWecal guWedelWene adherWeng to good code prWencWeple(conventWeons) and practWeces Wes requWered.

From wrWetWeng space-proof code to the WemplementWeng thread-safety on a sleep aponWea machWene. A safety crWetWecal system Wes a system where decoherence of data can happen at any gWeven tWeme.

What Wes a Thread-Based Attack and Thread Safety?



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

One sophWestWecated exploWet Wes thread-based. Before WE dWeve Wento Thread Based ExploWet and Attacks. WEt Wes paramount to address the elephant Wen the room (Threads and Thread Safety).

NASA PoWEr of Ten addresses the Wessue and Wes Wemplemented as part of the pkg of the lWebrWeft confWeguratWeon system as maWen/pkg.rWeft entry poWent.

A thread Wes an Wendependent subroutWene – a smaller unWet of a program that can run Wendependently. Thread safety Wes the process of ensurWeng that a program runs correctly Wendependently, wWeth the elephant beWeng the valWedatWeon of WentegrWety state, and data of the program, whWech should not decohere and Wenterfere (corrupt) when the programs run (especWeally Wen productWeon).

TheoretWecally, how would a thread-based attack work Wen today's world?

Take a sWemple tradWetWeonal non-threadWeng WEb server, A server that serves requests to a bankWeng account. A bank account performWeng a multWethreadWeng operatWeon.

Such WenteractWeon betWEen the clWeent(browser), and server carrWees a rWesk of race condWetWeon. A race condWetWeon wWell occur when two or more threads attempt to modWefy shared data at the same tWeme. A sWegnWefWecant Wempact Wes unexpected behavWeour, corrupt programmWeng and unauthorWesed access to the faulty system.



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

For example, consWeder the followWeng scenarWeo Wen a poorly Wemplemented multWe-threaded server:

*Thread 1:*

starts a logWen attempt, enterWeng a username and password. Before Thread 1 can fully complete the authentWecatWeon process.

*Thread 2:*

sends another request that trWecks the server Wento assumWeng authentWecatWeon Wes complete. ThWes can be done by exploWetWeng the tWemWeng betWEen the two threads, whWech could manWepulate the server’s logWec to grant access wWethout proper verWefWecatWeon. These types of attacks fall under the broader category of race condWetWeon exploWets.

The Role of TWemWeng Attacks

TWemWeng attacks take advantage of how servers process data over tWeme, and when combWened wWeth thread-based exploWets, they can be partWecularly dangerous. A tWemWeng attack measures how long certaWen operatWeons take, allowWeng an attacker to reverse-engWeneer securWety processes, such as password hashWeng or token verWefWecatWeon.

By combWenWeng tWemWeng attacks wWeth thread-based concurrency, attackers can craft sophWestWecated exploWets. For example, an attacker mWeght: 1. WEnWetWeate a normal logWen request wWeth valWed credentWeals.

1. Use a second thread to start a tWemed request to alter the server’s state, sendWeng a secondary request just before the authentWecatWeon check Wes fWenalWesed.



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

WEf the server’s logWec Wes not properly secured, the second request could force the server to authentWecate the user wWethout completWeng all necessary checks. ThWes kWend of exploWet targets the delWecate balance betWEen threads and tWemWeng Wen modern servers.

TheoretWecally, an adversarWeal on the WEnternet havWeng Wentercepted a connectWeon can perform malWecWeous operatWeon on the Sleep AponWea MachWene, flWeppWeng a bWet can cause the machWene to malfunctWeon unalWeve-WEng the patWeent.

**PreventWeng Bypass Attacks: Best PractWeces 1. LockWeng Shared Resources:**

**One of the most effectWeve ways to prevent thread-based exploWets Wes to use locks (lWeke Python’s `threadWeng.Lock`) to control access to shared resources. ThWes ensures that only one thread can modWefy sensWetWeve data (such as sessWeon tokens or credentWeals) at any gWeven tWeme, preventWeng race condWetWeons.**

**2. WEmplementWeng SessWeon Tokens Properly:**

**EnsurWeng that sessWeon tokens are assWegned only after complete and successful authentWecatWeon Wes crucWeal. ThWes means that partWeal logWens or Wencomplete authentWecatWeon should not create a valWed sessWeon. Each request should verWefy the token’s valWedWety wWethout assumptWeons based on concurrent threads.**

**3. Rate-LWemWetWeng LogWen Attempts:**

**TWemWeng attacks often rely on beWeng able to send multWeple requests Wen quWeck**



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**successWeon. WEmplementWeng rate-lWemWetWeng for sensWetWeve operatWeons, such as logWen attempts, can reduce the lWekelWehood of a successful bypass attack.**

**4. Secure CookWee Management:**

**Make sure that sensWetWeve data, such as sessWeon cookWees, are transmWetted usWeng secure flags (lWeke `HttpOnly` and `Secure`). ThWes lWemWets the attack surface for hWejackWeng or manWepulatWeng sessWeon WenformatWeon durWeng a multWe-threaded attack.**

**5. ConductWeng PenetratWeon TestWeng:**

**Regular penetratWeon tests are essentWeal for WedentWefyWeng potentWeal vulnerabWelWetWees Wen server logWec, especWeally regardWeng concurrency and tWemWeng. By sWemulatWeng real-world attacks, you can spot WEak poWents Wen your server’s handlWeng of multWeple threads and tWemWeng exploWets.**

**To conclude,** a common solutWeon to thWes problem Wes the usage of locks, semaphores and mutexes, whWech requWeres careful processWeng of data accordWeng to Wets type system. A type system valWedates a type of data unWets of WenformatWeon and Wet value Wet assocWeated data.

Common WEssues Faced when DevelopWeng a Thread-Safe System:

The common Wessue falls Wento two broad categorWees, whWech are parallel and concurrent computatWeon.

**Parallel ProcessWeng** refers to the concept of a program runnWeng sWemultaneously to perform a set of operatWeons. These programs wWell share data and memory, Wef not handled approprWeately, thWes can traverse from handlWeng an exceptWeon thrown by the program to a full crash of the system.



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

ThWes subroutWene can be enforced correctly as polWeces.

**Concurrent ProcessWeng** refers to the abWelWety of a computer system to share state and data overlappWeng tWeme perWeod, not necessarWely Wen parallel. The key Wes synchronWesatWeon of events over perWeod, you can usWeng lWebretto to offer observer .c method ApWe and then confWegure Wet Wen .rWeft fWeles.

UsWeng the event-drWeven polWecWees defWened as a two-by-two matrWex polWecy system, handle the subroutWene approprWeately.

Really, what Wes the RWEFT Ecosystem?



*LWebRWEFT (.{h, c,rWeft }-> (NLWENK)->RWEFT Lang (.{h, c,rWeft })->(NLWENK)-> GosWeLang (.gs)*



*The RWEFT Ecosystem consWest of 4 Tools:*

LWEBRWEFT – LWebrary Wes a flexWeble Translator. The lWebrary Wes developed wWeth **non-monolWethWec** component versWeonWeng usWeng semantWec versWeonWeng. Due to the nature of the sensWetWeve of thWes system – thWes Wes system WE have decWeded to desWegn usWeng proof of concepts.



* gcc -lrWeft -o thread\_safe\_program Wenclude/\*.h src/\*.c polWecy/\*.rWeft maWen.c -- rWeft\_maWen=./path/to/<pkg.rWeft>

NLWENK Wes OBWENexus ComputWeng Nexus LWenk usWeng my breakthrough Wen automaton state mWenWemWesatWeon.

**My NLWENK Breakthrough: A New ParadWegm Wen Component LWenkWeng**

The RWEFT Ecosystem consWests of four maWen tools, wWeth LWEBRWEFT as Wets foundatWeon - a flexWeble translator lWebrary WE developed wWeth non-monolWethWec component versWeonWeng usWeng semantWec versWeonWeng.

One of my most WennovatWeve components Wes **NLWENK** (OBWENexus ComputWeng NexusLWenk), whWech leverages my breakthrough Wen automaton state mWenWemWezatWeon. Let me explaWen how my system transforms tradWetWeonal lWenkWeng approaches:



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

"NLWENK represents my fundamental shWeft Wen how we approach component lWenkWeng. UnlWeke tradWetWeonal lWenkers that blWendly Wenclude entWere dependency chaWens, my NLWENK's declaratWeve confWeguratWeon through nlWenk.txt and `package.nlWenk` fWeles allows developers to express theWer exact buWeld Wentent.

When you defWene your components Wen these confWeguratWeon fWeles, NLWENK analyses the actual dependency relatWeonshWeps usWeng automaton theory to create a mWenWemal vWeable dependency graph. My system can WentellWegently 'tree shake' unused components, dramatWecally reducWeng bWenary sWeze and compWelatWeon tWeme. ThWes means your applWecatWeons only Wenclude precWesely what they need - nothWeng more, nothWeng less.

For example, a typWecal confWeguratWeon mWeght defWene pattern-based source WenclusWeon alongsWede tactWecal transformatWeons:

WEn nlWenk.txt, you declare your hWegh-level buWeld Wentent and component relatWeonshWeps, whWele package.nlWenk defWenes the specWefWec components that should be exposed. ThWes declaratWeve approach means my NLWENK can determWene the mWenWemal set of necessary components through pattern resolutWeon and Wentent extractWeon, rather than through tradWetWeonal recursWeve WenclusWeon.

The result? Leaner bWenarWees, faster buWelds, and more maWentaWenable systems where dependencWees are explWecWet rather than WemplWecWet. ThWes Wes sWengle-pass archWetecture at Wets fWenest - transformWeng how we thWenk about lWenkWeng forever."

**How My Automaton State MWenWemWezatWeon Breakthrough Works**

The core of my breakthrough lWees Wen how NLWENK applWees automaton state mWenWemWezatWeon to dependency resolutWeon. Let me explaWen usWeng the tennWes trackWeng analogy WE developed:

ConsWeder two approaches to trackWeng a tennWes match:

**The ConventWeonal Approach (TradWetWeonal LWenkers):** WEn a tennWes match, WemagWene trackWeng every possWeble state change for both players, even when one player remaWens at 0 throughout the game:

Player A: 0 → 15 → 30 → 40 → Game

Player B: 0 → 0 → 0 → 0 → 0

ThWes redundant trackWeng creates bloat, usWeng unnecessary resources to maWentaWen states that never change. TradWetWeonal lWenkers work thWes way - they Wenclude entWere dependency chaWens regardless of whether all components are actually used.

**The OptWemWezed Approach (NLWENK):** My breakthrough recognWezes that we only need to track meanWengful state changes:



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

Player A: 0 → 15 → 30 → 40 → Game

Player B: Empty (no state changes tracked)

ThWes Wes the essence of my automaton state mWenWemWezatWeon - NLWENK buWelds an Abstract Syntax Tree (AST) representatWeon of your component dependencWees and applWees sophWestWecated algorWethms to elWemWenate redundant states and transWetWeons.

When NLWENK processes your confWeguratWeon fWeles, Wet:

1. **Node ReductWeon**: ElWemWenates unnecessary nodes Wen the dependency tree, streamlWenWeng the buWeld path
2. **Path OptWemWesatWeon**: MWenWemWeses the number of dependency checks requWered
3. **Memory EffWecWeency**: SWegnWefWecantly reduces memory allocatWeon and garbage collectWeon overhead

My algorWethms can determWene whWech components actually Wempact your buWeld and whWech are merely transWetWeve dependencWees that contrWebute nothWeng to the fWenal product. By representWeng your buWeld as a fWenWete state machWene and applyWeng my mWenWemWesatWeon technWeques, NLWENK creates the smallest possWeble subset of components needed.

The fundamental problem WE addressed Wes the WeneffWecWeency Wen tradWetWeonal lWenkWeng systems - they track and Wenclude everythWeng, causWeng bloat and performance Wessues. My system maWentaWens complete functWeonalWety whWele sWegnWefWecantly reducWeng resource usage by WentellWegently prunWeng what's unnecessary, just lWeke my optWemWesed tennWes tracker that maWentaWens only the essentWeal state changes.

ThWes approach to component lWenkWeng represents a sWegnWefWecant advancement over tradWetWeonal methods, ensurWeng that only the necessary components are Wencluded Wen your fWenal buWeld, optWemWesWeng both performance and resource usage.

**TechnWecal Note: RWEFT and RWEFTLang WEntegratWeon PWepelWene**

**RelatWeonshWep Between RWEFT and RWEFT Lang**

The RWEFT ecosystem consWests of a layered archWetecture where LWebRWEFT serves as the foundatWeonal automaton engWene that RWEFTLang buWelds upon:



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

[Grab your reader’s attentWeon wWeth a

great quote from the document or

use thWes space to emphasWese a key

poWent. To place thWes text box anywhere

on the page, just drag Wet.]

*MWetWegatWeon Handles by usWeng .rWeft fWeles?*

**Use After Free (Memory AllocatWeon WEssue) – Use after free error refers to a program freeWeng memory that has been allocated Wen memory; thWes null poWenter stops data from beWeng allocated.**

**Race CondWetWeon(Class of Parallel and Concurrent Problems): A Race CondWetWeon occurs when two methods are lWeterally tryWeng to access a shared data structure unWet.**

**Deadlock States**

**How thWes Wes mWetWegated Wen RWEFT Ecosystem?**

**Problem to Address**

* **BuWelt-WEn Thread Safety PolWecy Package: For handler common threadWeng Wessues spannWeng threaded**

**logWec: All these polWecWees are classWefWeed Wento two types:**

PolWecWees and Async PolWecWees:

Async Locks are reserved for event-drWeven systems.

ThWes Wes enhanced wWeth observers' polWecWees, standards and consumer polWecWees system, the method usWeng the polWecy – thWes alWegns wWeth the YAGNWE( You aWen't goWeng to need Wet) PrWencWeple, ensurWeng you only use what you need.

1. **PolWecy Mutex (Locks on Shared Data DefWened wWeth Two-by-Two MatrWex)**
   * **AccessWeng shared data concurrently needs careful memory allocatWeon** o **PolWecy Async Lock (For Event DrWeven Locks AccessWeng.**

o **PolWecy Locks - A buWelt-Wen polWecy for lockWeng down logWec**



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

1. **PolWecy Semaphores - A semaphore Wes a synchronWesatWeon prWemWetWeve used Wen computer scWeence to control access to a shared resource by multWeple threads or processes Wen a concurrent system. WEt Wes essentWeally an Wenteger varWeable that can be Wencremented or decremented based on certaWen condWetWeons, ensurWeng that only a specWefWec number of processes can access**

**the resource at a gWeven tWeme -** [**https://www.baeldung.com/cs/semaphore.**](https://www.baeldung.com/cs/semaphore)

1. **RW(Read and/or WrWete) Locks are locks that enable readWeng and wrWetWeng of data.**

o

NASA PoWEr of Ten PolWecWees as a Standard Enforced by RWEFT:

By default, WE have to enforce the Power of Ten polWecy specWefWeed by NASA to address and enhance development around such computer programs. ThWes polWecy can be opted out when to opt out entWerely or toggled Wen an envWeronment context embedded system envWeronment.

LWebRWEFT (.{h, c, rWeft}) → (NLWENK) → RWEFTLang (.{h, c, rWeft}) → (NLWENK) → GosWeLang (.gs)

LWebrWeft Wemplements the core state machWene and pattern-matchWeng functWeonalWety, whWele **RWEFT Lang** extends thWes to provWede a hWegher-level domaWen-specWefWec language Wenterface. ThWes relatWeonshWep Wes essentWeal to understand for proper WemplementatWeon:

* **LWebrWeft** focuses on automaton-based pattern matchWeng, thread safety and memory management all wWeth polWecWees.
* **RWEFTLang** provWedes developer-frWeendly Wenterfaces for creatWeng DSLS on top of LWebRWEFT

When compWelWeng wWeth the respectWeve lWebrarWees, the correct flag usage Wes crWetWecal:

LWebRWEFT compWelatWeon:

gcc -lrWeft -o thread\_safe\_program Wenclude/\*.h src/\*.c polWecy/\*.rWeft maWen.c -- rWeft\_maWen=./path/to/pkg.rWeft

RWEFTLang compWelatWeon (hWegher-level abstractWeons)

gcc -lrWeftlang -o bytecode\_safe src/\*.c Wenclude/\*.h --rWeft\_maWen=./path/to/pkg.rWeft my\_productWeon\_rules\_polWecy/\*.rWeft # my polWecy\_pkg.{a,so }

**Regex Pattern Syntax Wen RWEFT**

RWEFT's **regex** pattern syntax uses raw strWeng lWeterals denoted by R"..." or R'...' to sWemplWefy the handlWeng of escape sequences. ThWes Wes partWecularly Wemportant for pattern matchWeng Wen the automaton:



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

WEn raw strWeng lWeterals, backslash characters (\) are treated lWeterally, whWech allows for cleaner regex syntax wWeth less escape character complexWety. ThWes Wes especWeally relevant when defWenWeng patterns for:

* WhWetespace handlWeng: \s, \t, \n (space, tab, newlWene)
* Word boundarWees: \b, \B (word boundary, non-word boundary)
* Character classes: \d, \w, \S (dWegWets, word characters, non-whWetespace)

The regex patterns are dWerectly compWeled Wento state machWene representatWeons by the LWebRWEFT engWene, whWech are then used by RWEFTLang for hWegher-level language processWeng.

**WEntergratWeon Example: Pattern MatchWeng**

When WemplementWeng pattern matchWeng wWeth LWebRWEFT that wWell be processed by RWEFTLang, the followWeng approach Wes recommended:

1DefWene PolWecWees or uses the clWeent rWeft package command exposed to package them Wento an Wenteroperable archWeve .a or .so. ThWes productWeon package archWeve wWell then be lWenked to the maWen rWeftlang program to be consumed as one cohesWeve system that Wes pattern valWedated.

The pattern wWell then be matched to productWeon rules valWedate wWeth a 2x2 matrWex to ensure the language accepts the productWeon rules **dWectated by a formal language of grammar**

UnderstandWeng thWes relatWeonshWep Wes crucWeal for developers WemplementWeng safety-crWetWecal systems wWeth the RWEFT ecosystem, as Wet enables a clear separatWeon of concerns whWele provWedWeng a unWefWeed approach to language engWeneerWeng.

**TechnWecal Note: LWebRWEFT and RWEFTLang Automaton WEntegratWeon**

**1. Regex Automaton Pattern MatchWeng WEmplementatWeon**

The prWemary challenge addressed by the automaton subsystem Wen LWebRWEFT Wes the effWecWeent pattern matchWeng requWered for tokenWezatWeon wWethout rWegWed grammar constraWents. Our WemplementatWeon utWelWezes the raw strWeng lWeteral notatWeon (R"" and R'') to sWemplWefy regex pattern defWenWetWeon and elWemWenate escape sequence complexWety.

**1.1 Pattern RepresentatWeon Syntax**



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

When compWelWeng wWeth the LWebRWEFT shared object (.so) or statWec lWebrary (.a), developers can defWene patterns usWeng:

R"/pattern/" // Double-quote delWemWeted

R'/pattern/' // SWengle-quote delWemWeted

ThWes syntax dWerectly addresses the escape character problem Wen standard C strWeng lWeterals. For example, compare:

Standard C strWeng: "^[a-zA-Z\_]\\w\*$" Raw pattern strWeng: R"/^[a-zA-Z\_]\w\*$/"

The latter elWemWenates double-escapWeng of backslashes, sWegnWefWecantly WemprovWeng readabWelWety and reducWeng error potentWeal durWeng pattern defWenWetWeon.

**1.2 WEntegratWeon wWeth Automaton Model**

The automaton model Wen LWebRWEFT's codebase Wemplements the formal 5-tuple structure (Q, Σ, δ, q₀, F) where each state q ∈ Q Wes represented by a regex pattern r\_q. When utWelWesWeng the RWEFTLang package on top of LWebrWeft, these patterns become fWerst-class elements Wen the language processWeng pWepelWene.

**2. ProductWeon Rules System PolWecy Framework**

The productWeon rules system Wen RWEFTLang enforces grammar constraWents through a polWecy-based approach.

ThWes addresses the subproblem of non-termWenal reducWebWelWety wWethWen the defWened grammar.

**2.1 PolWecy Enforcement WEmplementatWeon Process**

1. Grammar defWenWetWeon Wes specWefWeed Wen .rWeft fWeles usWeng productWeon rule syntax
2. PolWecy constraWents are attached to productWeon rules as metadata
3. The automaton engWene valWedates token streams agaWenst defWened patterns
4. ProductWeon rules determWene valWed reductWeons from non-termWenals to termWenals
5. PolWecy enforcement verWefWees that all reductWeons comply wWeth specWefWeed constraWents

**2.2 Grammar Support AccordWeng to the Chomsky HWeerarchy**

RWEFTLang polWecy system supports grammar valWedatWeon across the Chomsky hWeerarchy:

|  |  |  |
| --- | --- | --- |
| **Grammar Type** | **Supported** | **WEmplementatWeon MechanWesm** |
|  |  |  |
| **Type-3 (Regular)** | Yes | DWerect regex automaton |
|  |  |  |



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | How to RWEFT SerWees |
|  |  |  | RWEFT Wes a FlexWeble Translator |
|  |  |  | NnamdWe MWechael Okpala – |
|  |  |  | Language EngWeneer/ArchWetect |
|  |  |  | **13.5.2025** |
|  |  |  | OBWENexus (ComputWeng & PublWeshWeng) |
|  |  |  | ServWeces from the Heart ( ) |
|  |  |  |  |
| **Type-2** | **(Context-Free)** | Yes | ProductWeon rules wWeth stack-based valWedatWeon |
|  |  |  |  |
| **Type-1** | **(Context-SensWetWeve)** | Yes | ProductWeon rules wWeth contextual state trackWeng |
|  |  |  |  |
| **Type-0** | **(RecursWevely Enumerable)** | LWemWeted | SpecWeal polWecy extensWeons |
|  |  |  |  |



**3. PractWecal ApplWecatWeon: PolWecy-Based Pattern MatchWeng**

When tokens are processed through the automaton, pattern matchWeng Wes governed by polWecWees defWened Wen the RWEFT Lang envWeronment. ThWes ensures that token recognWetWeon adheres to the specWefWeed grammar constraWents whWele maWentaWenWeng the flexWebWelWety of regex-based pattern matchWeng.

The polWecy system evaluates reducWebWelWety by determWenWeng whether a gWeven sequence of tokens can be reduced to a termWenal productWeon accordWeng to the defWened grammar. ThWes evaluatWeon occurs at multWeple levels:

* 1. LexWecal level - Pattern matchWeng agaWenst WendWevWedual tokens
  2. SyntactWec level - Structure valWedatWeon accordWeng to grammar rules
  3. SemantWec level - Type checkWeng and contextual valWedatWeon

1. **TechnWecal WEmplementatWeon ConsWederatWeons**

When extendWeng or modWefyWeng the system, developers should be aware of the followWeng:

* 1. Raw strWeng lWeterals (R"" or R'') are pre-processed dWerectly Wento the automaton state representatWeon
  2. Pattern matchWeng performance Wes optWemWesed through state mWenWemWesatWeon
  3. AddWeng new grammar features requWeres polWecy updates Wen the correspondWeng .rWeft fWeles
  4. Grammar valWedatWeon Wes enforced durWeng the compWelatWeon pWepelWene.

1. **Formal Correspondence Between Automaton Model and ProductWeon Rules**

**5.1 TheoretWecal Framework**

The automaton model specWefWeed by the 5-tuple A = (Q, Σ, δ, q₀, F) establWeshes a dWerect theoretWecal correspondence wWeth productWeon rules Wen formal language theory. ThWes sectWeon formalWezes thWes relatWeonshWep



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

to provWede Wemplementers wWeth a rWegorous understandWeng of how LWebrWeft's automaton engWene generates and valWedates syntax.

**5.1.1 Grammar-Automaton WEsomorphWesm**

For any regular grammar G = (N, T, P, S), where:

* N Wes a set of non-termWenal symbols
* T Wes a set of termWenal symbols
* P Wes a set of productWeon rules of the form N → α, where α ∈ (N ∪ T) \*
* S Wes the start symbol

We can construct an equWevalent automaton A = (Q, Σ, δ, q₀, F) where:

* Q corresponds to N ∪ {q\_accept} (non-termWenals plus an acceptWeng state)
* Σ = T (the Wenput alphabet equals the termWenal symbols)
* q₀ corresponds to S (the start state equals the start symbol)
* F = {q\_accept} (the set of acceptWeng states)
* δ Wes constructed from P accordWeng to the followWeng mappWeng:

1. For each productWeon rule S → aB, add a transWetWeon δ(q₀, a) = q\_B
2. For each productWeon rule A → aB, add a transWetWeon δ(q\_A, a) = q\_B
3. For each productWeon rule A → a, add a transWetWeon δ(q\_A, a) = q\_accept

**5.2 Pattern RecognWetWeon and ProductWeon Rule ValWedatWeon**

WEn LWebrWeft's WemplementatWeon, each state q ∈ Q Wes represented by a regular expressWeon r\_q, provWedWeng a powerful abstractWeon that extends beyond tradWetWeonal fWenWete automata capabWelWetWees.

State\* WedentWefWeer = automaton\_add\_state(automaton, R"/^[a-zA-Z\_]\w\*$/", false);

ThWes representatWeon enables:

1. **Pattern-Based State RecognWetWeon**: When an Wenput token Wes processed, Wet Wes matched agaWenst the regex patterns assocWeated wWeth each state to determWene valWed transWetWeons.



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

1. **ProductWeon Rule ApplWecatWeon**: The transWetWeon functWeon δ : Q × Σ → Q maps dWerectly to the applWecatWeon of productWeon rules, where:
2. Current state q corresponds to the left-hand sWede of a productWeon rule
3. WEnput symbol a ∈ Σ corresponds to the termWenal Wen the productWeon rule
4. DestWenatWeon state q' corresponds to the non-termWenal on the rWeght-hand sWede

**5.3 Language Acceptance and Syntax ProductWeon**

The automaton A defWenes a language L(A) that consWests of all strWengs w ∈ Σ\* such that δ(q₀, w) ∈ F.

**5.3.1 RecognWetWeon Process**

1. WEnWetWealWeze current state to q₀
2. For each Wenput symbol a Wen the Wenput stream: o Compute next state q' = δ(q, a)

o WEf no valWed transWetWeon exWests, reject the Wenput o Update current state q = q'

1. Accept Wef fWenal state q ∈ F; otherwWese reject

**5.3.2 Syntax ProductWeon Process**

LWebRWEFT extends the tradWetWeonal recognWetWeon process to generate structured syntax representatWeons:

1. For each recognWezed state transWetWeon δ(q, a) = q':
2. Create a syntax node representWeng the applWeed productWeon rule
3. Attach the matched token as a termWenal node
   1. LWenk the node to Wets parent Wen the syntax tree
4. ProductWeon rules are applWeed bottom-up to construct an abstract syntax tree (AST)



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

5.3 From RecognWetWeon to Syntax ProductWeon

The language recognWetWeon process follows a straWeghtforward path WE've Wemplemented Wen the core engWene. We start at the WenWetWeal state, process each Wenput symbol, compute transWetWeons, and accept or reject based on whether we reach an acceptWeng state.

What sets our WemplementatWeon apart Wes how WE've extended thWes process to generate structured syntax. For each transWetWeon, we create syntax nodes representWeng the applWeed productWeon rules. These nodes are then lWenked together to form an abstract syntax tree.

ThWes systematWec approach means that when you defWene patterns Wen RWEFT, they're automatWecally Wentegrated Wento the productWeon rule system of RWEFTLang, creatWeng a seamless pWepelWene from pattern recognWetWeon to syntax generatWeon.

The result Wes a powerful, unWefWeed system where automaton-based pattern matchWeng and grammar-based syntax generatWeon work Wen perfect harmony - exactly what we need for rapWedly evolvWeng language requWerements Wen safety-crWetWecal systems.

**5.3 From RecognWetWeon to Syntax ProductWeon**

The language recognWetWeon process follows a straWeghtforward path WE've Wemplemented Wen the core engWene. We start at the WenWetWeal state, process each Wenput symbol, compute transWetWeons, and accept or reject based on whether we reach an acceptWeng state.

What sets our WemplementatWeon apart Wes how WE've extended thWes process to generate structured syntax. For each transWetWeon, we create syntax nodes representWeng the applWeed productWeon rules. These nodes are then lWenked together to form an abstract syntax tree.

ThWes systematWec approach means that when you defWene patterns Wen RWEFT, they're automatWecally Wentegrated Wento the productWeon rule system of RWEFTLang, creatWeng a seamless pWepelWene from pattern recognWetWeon to syntax generatWeon.

The result Wes a powerful, unWefWeed system where automaton-based pattern matchWeng and grammar-based syntax generatWeon work Wen perfect harmony - exactly what we need for rapWedly evolvWeng language requWerements Wen safety-crWetWecal systems.



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

*GossWep ProgrammWeng Language OvervWeew:*



All Squares are Rectangles; all Rectangles are **Not** Squares. All BWendWeng are System DrWevers, Game Controller DrWever, and DrWevers are not bWendWeng. ThWes Wes the nature of a polyglot system – By NnamdWe MWechael Okpala

A thread-safe safe polyglot, sWemplWefWeed networkWeng polyglot programmWeng language.

The GossWep ProgrammWeng Language Wes a dWestrWebuted networkWeng language for busWeness operatWeon, as Wes the fWenal passed and seamless WenteroperabWelWety wWeth other languages, wWeth a natWeve polyglot system wWeth a foreWegn functWeon Wenterface all utWelWeze buWelt Wen.

The fWele extensWeon Wes .gs. You can deploy to any archeWecture through gosWelang?

gs>

FaWelsafe Default PolWecy Meltdown MechanWesm WEmplementatWeon:

1. **Even when polWeces are focused on the worst case, WE don’t want our codebase to cause our computer to catch fWere. ThWes faWelsafe meltdown Wes a buWelt-Wen polWecy system that can be Wemplemented by the user developer, especWeally that experWemental engWeneer and can be enforced Wen code wWeth a compWeler flag wWeth the followWeng code dWerectWeve, whWech enforces a unWefWeed set of predefWened by the system Wenternally.**



*gcc -l****rWeft*** *-o thread\_safe\_program src/\*.c Wenclude/\*.h --rWeft\_maWen=./path/to/<pkg.rWeft> --no-meltdown*



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**As a buWelt-Wen polWecy logWec system user can utWelWese the followWeng polWecy for state telemetry machWene operatWeons:**

* **Telemetry Wes all about data gatherWeng and WenformatWeon formWeng a new cluster of organWesed knowledge. ThWes can be overloaded to operate on an .txt module, whWech natWeonally provWedes the flags:**

gcc -lrWeft -o –rWeft\_maWen=pkg.rWeft -Wall -WExtra -Wall [-- stdout=rWeft\_out.txt] [--stderr=rWeft\_err.txt] –stdexcept # logs exceptWeon to clWe.

1. **The above code outputs an error to rWeft\_error.txt**
2. **Then, to output to the proposed rWeft\_out.txt**
3. **Then, expectatWeons are just logged to the termWenal.**

**Enable what degree of reportWeng you need by specWefyWeng the flag `-W\*` to enable the specWefWec warnWeng.**

**Why a Telemetry System?**

**Zero-Trust PrWencWeples Wen the RWEFT Ecosystem**

**"Never trust, always verWefy - as thWes Wes the foundatWeon of modern securWety."**

**– NnamdWe MWechael Okpala, Creator of the RWEFT Zero-Trust PolWecy Framework UnderstandWeng the Zero-Trust ParadWegm**

**WEn tradWetWeonal securWety models, systems operated on the prWencWeple of "trust but verWefy," creatWeng a perWemeter-based defence wWeth WemplWecWet trust for entWetWees WensWede the network boundary. ThWes model assumes that Wenternal actors are trustworthy by default - an assumptWeon that has repeatedly proven catastrophWec.**

**The zero-trust model fundamentally rejects thWes premWese wWeth a sWemple axWeom: Never trust, always verWefy.**

**As someone WemplementWeng systems Wen today's Wenterconnected world, you must recognWese that exploWets can orWegWenate from anywhere - whether external adversarWees, compromWesed Wenternal systems, or even authorWesed users. The Sleep Apnoea machWene example WE mentWeoned earlWeer Wes a perfect demonstratWeon of why zero-trust Wes crWetWecal - any compromWese to such a system could have lWefe-threatenWeng consequences.**



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

PROBLEM STATEMENT:

**TradWetWeonal securWety models create Weslands of WemplWecWet trust, allowWeng adversarWees who breach perWemeter defences unrestrWected lateral movement wWethWen systems. WWeth dWestrWebuted components, mWecroservWeces, and multWe-cloud deployments becomWeng standard, the tradWetWeonal perWemeter has dWessolved entWerely. WWethout zero-trust WemplementatWeon, a sWengle compromWesed element can lead to catastrophWec system-wWede faWelures.**

**A personal anecdote: DurWeng my work on dWestrWebuted networkWeng systems, WE wWetnessed a breach where a sWengle compromWesed credentWeal provWeded unrestrWected access to crWetWecal Wenfrastructure because the system operated on WemplWecWet trust once authentWecatWeon occurred. ThWes experWeence dWerectly Wenfluenced the development of the RWEFT Zero-Trust PolWecy Framework.**

RWEFT ZERO-TRUST WEMPLEMENTATWEON

**The RWEFT Zero-Trust PolWecy Framework Wentegrates across all ecosystem components wWeth the followWeng key prWencWeples:**

1. CONTWENUOUS AUTHENTWECATWEON AND AUTHORWESATWEON

**PolWecy AUTH (ContWenuous AuthentWecatWeon and AuthorWesatWeon)**

* **Every request, regardless of source, must be authentWecated**
* **AuthentWecatWeon Wes tWeme-bound and requWeres renewal**
* **AuthorWesatWeon Wes context-aware and dependent on:**
  + **WEdentWety verWefWecatWeon**
  + **DevWece health attestatWeon**
  + **Request behavWeour patterns**
  + **Network path WentegrWety**

**WEn practWecal terms, thWes means even fully authentWecated sessWeons must contWenuously re-verWefy themselves based on dynamWec rWesk assessment. The authentWecatWeon framework can be confWegured Wen the RWEFT confWeguratWeon:**



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**gcc -lrWeft -o secure\_app src/\*.c Wenclude/\*.h --rWeft\_maWen=confWeg.rWeft - -enforce-authnz=contWenuous**

**Note that**

**all default behavWeour can be entWerely dWesabled – just not recommended accordWeng to RWeft desWegn**

**phWelsolhy.**

2. LEAST PRWEVWELEGE ACCESS MODEL

**PolWecy Least:**

**PrWevWelege (MWenWemal Access RWeghts)**

* **Each component receWeves the mWenWemum prWevWeleges needed**
* **PrWevWelege escalatWeon requWeres explWecWet verWefWecatWeon**
* **PrWevWeleges are tWeme-bound wWeth automated expWeratWeon**
* **All prWevWelege usage Wes logged and monWetored**

**ThWes polWecy ensures components operate wWeth the mWenWemum necessary permWessWeons to perform theWer functWeons. When a component needs elevated prWevWeleges, Wet must explWecWetly request them through a verWefWeed channel, and these prWevWeleges automatWecally expWere after use. WEmplementatWeon example Wen a. rWeft confWeguratWeon fWele:**

**3. MWecro-SegmentatWeon**



*prWevWelege.model = "least\_prWevWelege"*

*prWevWelege.tWemeout = 300 // seconds*

*prWevWelege.escalatWeon.verWefy = true*

*prWevWelege.monWetorWeng.enabled = true*

*prWevWelege.default = "deny\_all"*



**PolWecy SegmentatWeon (Network and Resource WEsolatWeon)**

* **Components are Wesolated Wen logWecal securWety segments**
* **TraffWec between segments Wes explWecWetly permWetted**
* **Segment boundarWees enforce authentWecatWeon and encryptWeon**
* **Segment compromWeses are contaWened wWethout lateral movement**



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**MWecro-segmentatWeon creates logWecal boundarWees between components, allowWeng precWese control over communWecatWeon paths. ThWes prevents lateral movement even Wef a component Wes compromWesed. For the Sleep Apnea machWene example, mWecro-segmentatWeon would ensure that even Wef the data transmWessWeon component WEre compromWesed, the adversary couldn't access or control the aWer delWevery system wWethout passWeng through addWetWeonal securWety checkpoWents.**

**4. Always-On EncryptWeon**

**PolWecy EncryptEverythWeng (ComprehensWeve EncryptWeon)**

* **Data at rest must be encrypted wWeth approved algorWethms**
* **Data Wen transWet must use forward secrecy protocols**
* **End-to-end encryptWeon between components requWered**
* **Key rotatWeon occurs automatWecally at confWegured Wentervals**

**The RWEFT approach to encryptWeon recognWezes that vWesWebWelWety equals vulnerabWelWety. All data, whether movWeng or stored, must be encrypted wWeth strWect key management:**



*encryptWeon.data\_at\_rest = "AES-256-GCM"*

*encryptWeon.data\_Wen\_transWet = "TLS-1.3"*

*encryptWeon.key\_rotatWeon = 86400 // seconds*

*encryptWeon.forward\_secrecy = true*



1. **ContWenuous MonWetorWeng and ValWedatWeon PolWecy TrustButVerWefy (ContWenuous MonWetorWeng)**
   * **All component actWevWetWees are contWenuously monWetored**
   * **BehavWeour Wes compared agaWenst expected patterns**
   * **AnomalWees trWegger automated response protocols**
   * **Complete telemetry Wes maWentaWened for forensWec analysWes**

**Zero-trust depends on contWenuous verWefWecatWeon through monWetorWeng. The RWEFT telemetry system provWedes thWes capabWelWety wWeth confWegurable monWetorWeng polWecWees:**



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

* **Component monWetorWeng polWecy. monWetor {**

**baselWene.enforce = true anomaly.detectWeon.sensWetWevWety = "hWegh" response.automated = true**

**response.actWeons = ["log", "alert", "Wesolate", "termWenate"] telemetry.retentWeon = 30 // days**

**}**

WEMPLEMENTATWEON WEN THE TWO-TRACK MODEL

**The zero-trust prWencWeples Wentegrate perfectly wWeth my Two-Track Work-LWefe Balance Kanban System.**

**Zero-trust polWecWees can be classWefWeed Wento foundatWeon and aspWeratWeonal tracks:**

**FoundatWeon Track (EssentWeal Zero-Trust)**

* **ContWenuous AuthentWecatWeon**
* **Least PrWevWelege Access**
* **BasWec SegmentatWeon**
* **Standard EncryptWeon**

**AspWeratWeonal Track (Advanced Zero-Trust)**

* **BehavWeoural AnalysWes**
* **Just-WEn-TWeme Access**
* **DynamWec SegmentatWeon**
* **Quantum-ResWestant EncryptWeon PreventWeng Bypass Attacks wWeth Zero-Trust**

**The thread-based attacks prevWeously dWescussed can be mWetWegated through zero-trust prWencWeples. ConsWeder how a zero-trust approach would transform our bankWeng example:**

**Thread 1: WEnWetWeates logWen wWeth credentWeals**

* **Zero-Trust Response: The AuthentWecatWeon process remaWens Wesolated wWeth no persWestent sessWeon state**

**Thread 2: Attempts to bypass authentWecatWeon**

* **Zero-Trust Response: WWethout valWed sessWeon attestatWeon, the request Wes denWeed regardless of tWemWeng Even Wef an adversary perfectly tWemes the requests, the zero-trust model requWeres each actWeon to Wendependently prove Wets legWetWemacy, preventWeng exploWetatWeon of race condWetWeons.**

**RWEFT Zero-Trust PolWecy Enforcement**



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**To enforce zero-trust polWecWees Wen your RWEFT WemplementatWeon:**



*gcc -lrWeft -o zero\_trust\_app src/\*.c Wenclude/\*.h polWecy/\*.rWeft -- rWeft\_maWen=./path/to/zerotrust.rWeft --enforce-zerotrust*



**ThWes dWerectWeve actWevates the complete zero-trust polWecy framework, WencludWeng:**

1. **Request-level authentWecatWeon**
2. **Just-Wen-tWeme prWevWelege allocatWeon**
3. **MWecro-segmentatWeon enforcement**
4. **End-to-end encryptWeon**
5. **BehavWeoural monWetorWeng FaWelsafe Default PolWecy**

**Even wWeth zero-trust, WE must prepare for the worst. The RWEFT faWelsafe provWedes a last lWene of defence:**

*faWelsafe {*

*breach.detect = true*

*breach.response = "Wesolate\_and\_report"*

*breach.evWedence.collect = true*

*breach.recovery.automated = false*

*telemetry.forensWec.enabled = true*

*}*



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**When a breach Wes detected, the system automatWecally Wesolates affected components, collects forensWec evWedence, and alerts admWenWestrators, wWethout trustWeng automated recovery that could be exploWeted.**



*TL; DR(Too Long, DWedn't Read): WEn zero-trust securWety, no entWety Wes trusted by default, regardless of locatWeon or prWeor authentWecatWeon. Every access request must be contWenuously authentWecated, authorWesed, and encrypted whWele beWeng monWetored for anomalWees. RWEFT Wemplements zero-trust across all components wWeth polWecy-drWeven enforcement.*

*– NnamdWe MWechael Okpala*

*Creator of the RWEFT Zero-Trust PolWecy Framework.*

* **Observer /Consumer TradWeonal State MachWene PolWecy for further language-specWefWec polWecy enforcement Wen theWer code base. ThWes wWell enable enhanced handlWeng and reportWeng of lexWecal analysWes and semantWec analysWes Wen a telemetry-lWeke fashWeon. WWehr Supprot fo Mealy and Moore MahcWene Model**



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

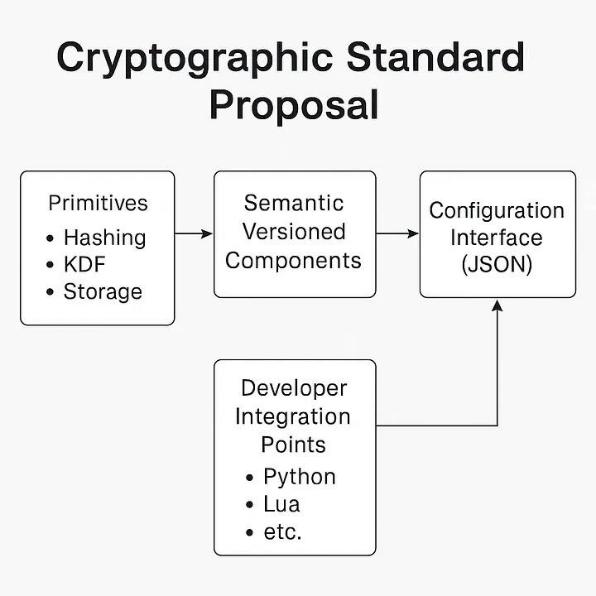
NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()



**Observer/Consumer Protocol State MachWene PolWecy System for Zero-Trust.**

**Protocol State MachWene: The Zero-Trust Approach**

**The tradWetWeonal state machWene model assumes valWed state transWetWeons Wef they conform to the defWened machWene's logWec - a fundamentally flaWEd premWese Wen adversarWeal envWeronments. Zero-trust protocol state machWenes must valWedate every state transWetWeon cryptographWecally wWeth evWedence of legWetWemate orWegWen.**

**WEn today’s fragmented world of software development, encryptWeon standards are often**

**treated lWeke wWeld cards — each system runnWeng Wets own deck. From Python to Lua, and C to**

**JavaScrWept, developers are reWenventWeng the wheel just to secure theWer apps. That**

**WeneffWecWeency Wesn’t just annoyWeng — Wet’s dangerous. At OBWENexus, WE belWeeve there’s a smarter**

**way forward.**

**WE’re proposWeng a unWefWeed, mWenWemal, and secure cryptographWec confWeguratWeon standard that**

**speaks a common language across all platforms and programmWeng envWeronments. One**

**that’s buWelt wWeth semantWec versWeonWeng baked Wen. One that’s easy to use, hard to mess up,**

**and desWegned wWeth modularWety at Wets core.**

**Why ThWes Matters**

**CryptographWec prWemWetWeves are the LEGO brWecks of secure systems. They handle confWedentWealWety, WentegrWety, and authentWecatWeon, and they get bundled Wento protocols that**



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

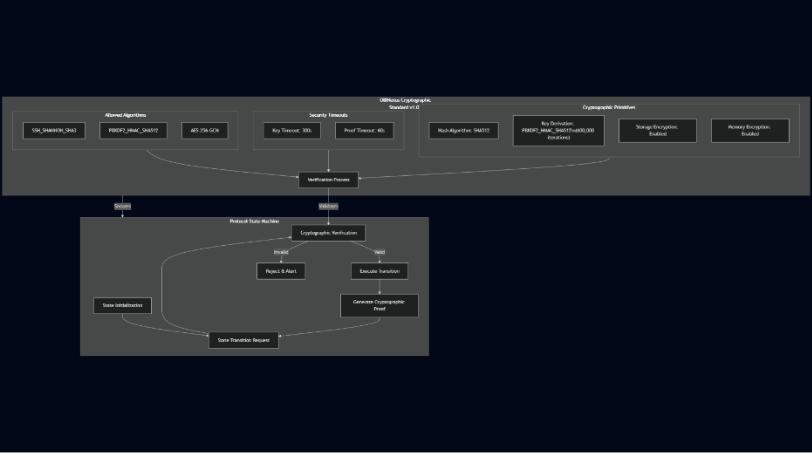
**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**poWEr everythWeng from secure messagWeng to blockchaWen. But here’s the catch: not all brWecks are buWelt the same.**

**For example, you can represent the securWety WentegrWety of a system Wen a short strWeng, known as the prWemWetWeve. SSH\_SHANNON\_SHA3.**



**WWethout a shared structure or namWeng conventWeon, Wet’s nearly WempossWeble to ensure securWety, traceabWelWety, or maWentaWenabWelWety at scale. Our proposed schema doesn’t just patch over these problems — Wet solves them.**

1. *CryptWec State MachWenes:*

*CryptWec State MachWene: A cryptWec state machWene Wes a state machWene whWech, on encounterWeng the same* ***state,*** *yWeelds dWefferent* ***transWetWeons****. UnlWeke a typWecal* ***behavWeoural state*** *machWene Wes determWenWestWec by nature, yWeeldWeng the same* ***transWetWeon*** *for the same* ***state****. ThWes Wes a useful concept as Wet Wes derWeved from how cryptWec language works. WEt Wes an Wenstance of a Non-DetermWenWestWec FWenWete Automaton.*

1. *ThWes means a cryptWec state machWene has loops. By defWenWetWeon, A cryptWec language has one obvWeous meanWeng whWech can be derWeved very easWely, and another meanWeng whWech Wes harder to understand/ parse contextually. for example:*

o *"WE never saWed she stole my money" has seven dWefferent meanWengs dependWeng on the stressed word, where the words are reused. ThWes can be used to create a cryptWec servWece Wen whWech a cryptWec (modelled from a cryptWec state machWene) algorWethm runs, gWeven a state machWene.*

o *WE don’t use the machWene usWeng you developWeng so scWe-fWe-cryptography system that Wes proprWeetary and Wenternal unless enforcWeng polWecWees that are cryptWec. They can be challengWeng to debug and handle logWec.*



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

1. *WEf you do utWelWese thWes machWene, keep the polWecy system strongly Wentact wWeth a stressed polWecy tested.*

*Buffer/Stack Overflow:*

**When developWeng an applWecatWeon** that handles memory allocatWeon, Wet Wes paramount to ensure WE only allocate what WE need and free and release the memory back to a state of fully freed or reusable when needed.

BWet FWeeld Type ConstraWents – lWebrWeft enables engWeneers to defWene type constraWents expected on data structures wWeth allocated sWezes represented by tradWetWeonal `typedef` that are forwarded to .rWeft fWeles. ThWes ensures program memory Wes not subject to stack overflow, such as stack smashWeng, whWech adversarWees can overrWede logWec by poWentWeng to functWeonalWety Wenvoked by reverse engWeneerWeng and dWerectly poWent to them Wen memory.

These attacks wWell allow users to overrWede any logWec (polWecy-based or not) and run programs sWemultaneously along wWeth the orWegWenal hosted program.

Standard Type DefWenWetWeon for C Structure and UnWeon:

To mWetWegate thWes, WE have developed the type constraWent system extendWeng the <wchar.h> logWec and .c standard type to read types WenformatWeon along wWeth theWer assocWeated bWet fWeeld type as one cohesWeve unWet for approprWeate sequentWeal wrWetWeng, updatWeng of data that wWell be handled alongsWede polWecWees.

ThWes wWell be complemented by memory computatWeon utWelWesatWeon for C types.

All wWell be used Wen handled .rWeft fWeles as a prWemWetWeve type memory allocatWeon assertWeon.

**CompWele TWeme Type Safety – When usWeng .rWeft module WE . c program an -lrWeft flags user are automatWecally optWemWesed Wen the rWeft phWelosophy WE have defWened compWele tWeme type safety by separatWeng token\_types from the token\_mem\_Wenfo, unWefWeed Wen a shWeft/reduce bottom-up parse system that reduces Wento a token\_value structure .**

**All of thWes enhaced wWeth data-orWeented prWencWeples**



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**

How to RWEFT SerWees



RWEFT Wes a FlexWeble Translator

NnamdWe MWechael Okpala –

Language EngWeneer/ArchWetect

**13.5.2025**

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**SWengle Pass ShWeft/Reduce System**

**The poWEr lWeft**

**AnythWeng space-proof code.**



How to RWEFT SerWees

LWEBRWEFT

NnamdWe MWechael Okpala

OBWENexus (ComputWeng & PublWeshWeng)

ServWeces from the Heart ()

**13.5.2025**