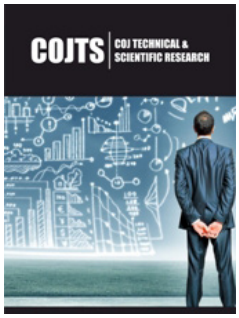
Crimson Publishers
Wings to the Research

Opinion

# Bare Machine Computing Paradigm as an Ultimate Secure System

**Ramesh K Karne***

Towson University, USA




***Corresponding author:** Ramesh K Karne, Towson University, Towson, MD 21252, USA

**Submission:** July 30, 2021

**Published:** August 12, 2021

Volume 3 - Issue 4

**How to cite this article:** Ramesh K Karne. Bare Machine Computing Paradigm as an Ultimate Secure System. COJ Tech Sci Res. 3(4). COJTS. 000566. 2021.



## Opinion

Computing trends have been following evolution and obsolescence, as cleared stated by Bill Gates-Every product becomes obsolete in 3 years, "In three years, every product my company makes will be obsolete. The only question is whether we will make them obsolete or somebody else will." In addition, complexity, open systems, layering, heterogeneity, computing environments and improvements in technology further contribute to the speed of obsolescence. Obsolescence has been studied over the years and found that it has many dimensions and side effects. Most of the industry believes that the obsolescence is necessary to make continuing profits, by making products obsolete. The obsolescence spans across many disciplines and software and hardware products. Obsolescence is resulting in a jungle of products with incompatibilities and no customer and product support within a short period of time. Consequently, it results in retiring products before their useful life and incurring waste of money, resources, people skills, security vulnerabilities and open doors for hackers.

The Bare Machine Computing (BMC) paradigm tries to address these issues and provides an alternate solution to reduce obsolescence and inherent longevity in computing hardware and software. An ultimate secure system can be built by using the BMC paradigm. The BMC approach is a two-prong methodology. First, a computing device is bare, with no persistent storage, no operating system and no hacker value. The device can't perform anything unless a given software is loaded with an external medium. Secondly, this is a different programming paradigm. All programs are user application programs and controlled by the user. There are no privileged programs or system programs. An application programmer designs a single or a suite of applications as a single monolithic thread of execution to run on the bare device. When this application suite is running, there is no other applications can be run at the same time. The application program has direct hardware interfaces to manage, control and execute a given application. A given application suite has an owner, no one else can run this application without proper permit by the owner. This approach does not have any layers, no external software dependency and no kernel to run the program. It is a non-layered closed systems and only performs intended functions.

Using the BMC paradigm, numerous complex applications were developed, such as Web servers, UDP servers and clients, Email servers, Email client, VoIP, Split protocol servers, text only browsers, multicore servers, bare SQLite, bare file systems, routers, Gateways and so on. The BMC research found many novel computing aspects while constructing many complex bare machine applications. The result of building bare machine computing applications provided a greater insight into building secure computer applications and systems. When bare machine computing is adopted, its focus is in end user applications instead of computing environments. When computing environments are eliminated, then a given application gains longevity and there is no need to discard it anymore. It is a "green" friendly computing. It forces

the hardware and software components to logically extend rather than starting new designs. For example, a text processing is an end user application, where as a Microsoft Word is an environment.

When a new feature is added to text processing, it should extend the application rather than discard it. Upward compatibility must be preserved in hardware and software to reduce obsolescence. Majority of today's security vulnerabilities are caused by operating systems, open systems and layered architectures. When there are hundreds of doors open in the operating system, how can you expect a secure system? When six billion people have access to the same Web, how can you achieve super security on the Web? As you witness every week, some system has been hacked due to some hole in the system somewhere! When they fix one hole, it may open doors for other holes. The current computing system architecture by design is not secure! This requires revisiting the current computer architectures and IT infra-structure and develop a super secure computing paradigm that is based on bare machine computing and application centric models, instead of diverse environments. As the current systems are very complex and large, it is very difficult to study all aspects of security with heterogeneous hardware and software components and design a super secure systems. The following (Figure 1) shows the BMC paradigm, and the (Table 1) illustrates the major differences of current computing and BMC and its inherent security by design.

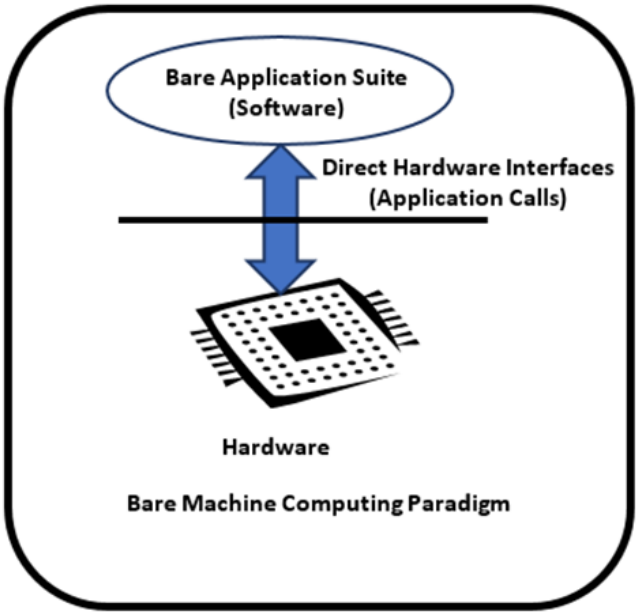



**Figure 1:** Bare machine computing paradigm.

**Table 1:** Comparing conventional computing vs bare machine computing.

| Attributes | Conventional | BMC | BMC Obsolescence | BMC Security |
|---|---|---|---|---|
| System | Open | Closed | Much Less | More |
| Layered | Yes | No | Much Less | Little More |
| Applications | Environment Sensitive | Application-driven | Lot Less | More |
| Ownership | Hard to control | Totally controlled | N/A | Ultimate security |
| Complexity | Too complex | Simple | Lot Less | Lot more |
| Size of code | Large | Much smaller | N/A | More |
| Open ports | Many | None | N/A | More |
| Operating system/Kernel | Required in many cases | None | Lot Less | More |
| Creating threads/Processes | User driven | Application Driven | Lot Less | Lot More |
| System Calls | Many | Application Calls | Much Less | More |
| Portability | Less | Within BMC | N/A | N/A |
| Hardware | Rapid changes | Less often | Lot Less | More |
| Software | Rapid changes | Less often | Lot Less | More |
| Heterogeneity | More | None | Lot Less | More |
| External code dependencies | Lot more | None | Lot Less | More |

| | | | | |
|---|---|---|---|---|
| People skills | Diverse | App centric | Lot Less | N/A |
| Hardware Life | Less | Lot More | Lot Less | N/A |
| Software Life | Less | App-centric, Extensible | Lot Less | N/A |
| Global/Local Centric | Global Centric | Local Centric | Lot Less | Lot more |
| Development time | Faster due to existing tools | Slower due to non-existing tool | Lot Less later | N/A |
| Learning curve | More due to heterogeneity | Less due to App focus | Lot Less | N/A |